WebSphere Business Integration Server Express
The Express Route to Business Integration

Integration basics
Overview of WebSphere Business Integration Server Express
Working example for new developers

Bill Moore
Knut Inge Buset
Dennis Higgins
Wagner Palmieri
Mads Pedersen
Joerg Wende
Alison Wong

ibm.com/redbooks
First Edition (February 2005)

This edition applies to V4.3 of WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus for use on Windows 2000.

© Copyright International Business Machines Corporation 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.
Chapter 3. WebSphere Business Integration Server Express

3.1 Key concepts

3.2 Integration solution

3.3 Architecture

3.3.1 Supported technologies

3.3.2 Collaborations

3.3.3 Adapters

3.3.4 Business Objects

3.3.5 Data map services

3.3.6 Toolset Express

3.4 Summary

Chapter 4. Installation and configuration

4.1 System requirements

4.1.1 Requirements for Microsoft Windows 2000

4.1.2 Windows XP

4.1.3 Requirements for OS/400

4.1.4 Requirements for Linux

4.2 Installation from the Launchpad

4.3 Installing prerequisite software

4.3.1 IBM Java Developer Kit

4.3.2 Installing IBM WebSphere MQ

4.3.3 Installing WebSphere Application Server - Express

4.3.4 Installing Microsoft Internet Explorer

4.3.5 Installing Netscape Navigator

4.3.6 Completing the installation of prerequisite software

4.4 Installing IBM DB2 Universal Database

4.4.1 Checking database installation prerequisites

4.4.2 Installing DB2 UDB Express

4.5 Installing WebSphere Business Integration Server Express

Chapter 5. Getting started

5.1 Using the System Manager

5.2 Creating a new instance of the JDBC adapter

5.3 Creating a test project

5.4 Creating a new connector

5.5 Creating Business Objects

5.6 Creating maps
5.7  Updating SAL301RConnector ............................................. 89
5.8  Testing the connector ..................................................... 90
5.9  Helpful resources ......................................................... 95

Part 2.  WebSphere Business Integration Server Express features ................. 97

Chapter 6.  Introduction to Toolset Express .................................. 99
6.1  Introduction ............................................................... 100
6.2  Business Object Designer ............................................... 103
6.3  Map Designer Express ................................................... 107
6.4  Connector Configurator Express ....................................... 119
6.5  Test Connector ............................................................. 126
6.6  Relationship Designer Express ......................................... 130
6.7  Process Designer Express ............................................... 136
6.8  Test Environment .......................................................... 148

Chapter 7.  Adapters .............................................................. 159
7.1  Adapter architecture ....................................................... 160
  7.1.1  Architectural overview .............................................. 160
  7.1.2  Adapter overview ..................................................... 161
  7.1.3  Event notification .................................................... 163
  7.1.4  Subscription handling ............................................... 164
  7.1.5  ASBO instantiation .................................................. 165
  7.1.6  Data mapping ........................................................ 167
  7.1.7  Transport layer ....................................................... 167
7.2  Express adapters .......................................................... 168
  7.2.1  Technology adapters ............................................... 168
  7.2.2  Application adapters ............................................... 170
  7.2.3  Capacity pack adapters ............................................. 170
7.3  How to implement an adapter ............................................ 172
  7.3.1  Installing and uninstalling an adapter ......................... 172
  7.3.2  Setting up event notification ..................................... 175
  7.3.3  Adapter configuration .............................................. 181
  7.3.4  Creating Business Objects ....................................... 188
  7.3.5  Modifying the ASBOs ................................................ 190
  7.3.6  Creating the GBOs .................................................. 192
  7.3.7  Setting up supported Business Objects ....................... 192
  7.3.8  Setting up supported maps ....................................... 193
  7.3.9  Adapter Agent runtime configuration ......................... 194
  7.3.10  Deploying the adapter ........................................... 198
  7.3.11  How to subscribe to events ..................................... 202
7.4  Data handlers ............................................................. 203
  7.4.1  Overview .............................................................. 204
  7.4.2  Available data handlers ........................................... 205
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4.3 Configuring a data handler</td>
<td>206</td>
</tr>
<tr>
<td>7.5 Adapter Development Kit</td>
<td>206</td>
</tr>
<tr>
<td>7.6 ODAs</td>
<td>207</td>
</tr>
<tr>
<td>7.6.1 ODA architectural overview</td>
<td>207</td>
</tr>
<tr>
<td>7.6.2 ODA Development Kit</td>
<td>208</td>
</tr>
<tr>
<td>7.6.1 ODA architectural overview</td>
<td>207</td>
</tr>
<tr>
<td>7.6.2 ODA Development Kit</td>
<td>208</td>
</tr>
<tr>
<td>Chapter 8. Business Objects</td>
<td>209</td>
</tr>
<tr>
<td>8.1 Introduction to Business Objects</td>
<td>210</td>
</tr>
<tr>
<td>8.1.1 Business Object definitions</td>
<td>211</td>
</tr>
<tr>
<td>8.1.2 Business Objects</td>
<td>218</td>
</tr>
<tr>
<td>8.2 Designing Business Object definitions</td>
<td>221</td>
</tr>
<tr>
<td>8.2.1 Using a methodology</td>
<td>221</td>
</tr>
<tr>
<td>8.2.2 Standards</td>
<td>222</td>
</tr>
<tr>
<td>8.3 Developing Business Object definitions</td>
<td>223</td>
</tr>
<tr>
<td>8.3.1 Overview of Business Object Designer Express</td>
<td>223</td>
</tr>
<tr>
<td>8.3.2 Developing an example</td>
<td>225</td>
</tr>
<tr>
<td>8.4 Developing with Object Discovery Agents</td>
<td>228</td>
</tr>
<tr>
<td>8.4.1 Developing an example using JDBCODA</td>
<td>228</td>
</tr>
<tr>
<td>8.4.2 Developing an example using XMLODA</td>
<td>234</td>
</tr>
<tr>
<td>8.5 Deploying Business Object definitions</td>
<td>240</td>
</tr>
<tr>
<td>8.6 Using packaged Business Object definitions</td>
<td>241</td>
</tr>
<tr>
<td>8.7 JDBC application-specific information</td>
<td>244</td>
</tr>
<tr>
<td>Chapter 9. Mapping</td>
<td>247</td>
</tr>
<tr>
<td>9.1 Developing mappings</td>
<td>248</td>
</tr>
<tr>
<td>9.1.1 Developing maps</td>
<td>248</td>
</tr>
<tr>
<td>9.1.2 Defining a simple map</td>
<td>252</td>
</tr>
<tr>
<td>9.1.3 Creating advanced mapping</td>
<td>261</td>
</tr>
<tr>
<td>9.1.4 Using Web services</td>
<td>275</td>
</tr>
<tr>
<td>9.1.5 Relationship mappings</td>
<td>279</td>
</tr>
<tr>
<td>9.1.6 Map documentation</td>
<td>282</td>
</tr>
<tr>
<td>9.2 Relationships</td>
<td>286</td>
</tr>
<tr>
<td>9.2.1 Relationship development</td>
<td>288</td>
</tr>
<tr>
<td>9.2.2 Relationship deployment</td>
<td>297</td>
</tr>
<tr>
<td>9.2.3 Testing relationships</td>
<td>298</td>
</tr>
<tr>
<td>9.2.4 Relationship management</td>
<td>303</td>
</tr>
<tr>
<td>Chapter 10. Collaborations</td>
<td>309</td>
</tr>
<tr>
<td>10.1 Introduction to collaborations</td>
<td>310</td>
</tr>
<tr>
<td>10.1.1 Collaboration components</td>
<td>311</td>
</tr>
<tr>
<td>10.1.2 Collaboration processing</td>
<td>315</td>
</tr>
<tr>
<td>10.1.3 Collaboration types</td>
<td>317</td>
</tr>
<tr>
<td>10.2 Collaborations in WebSphere Business Integration Server Express</td>
<td>318</td>
</tr>
<tr>
<td>10.2.1 Base collaboration templates</td>
<td>318</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>10.2.2 Collaboration capacity packs</td>
<td>321</td>
</tr>
<tr>
<td>10.3 Designing collaborations</td>
<td>323</td>
</tr>
<tr>
<td>10.3.1 Methodology</td>
<td>323</td>
</tr>
<tr>
<td>10.3.2 Collaboration standards</td>
<td>324</td>
</tr>
<tr>
<td>10.4 Developing collaborations</td>
<td>326</td>
</tr>
<tr>
<td>10.4.1 Overview of Process Designer Express</td>
<td>326</td>
</tr>
<tr>
<td>10.4.2 Developing and deploying collaborations</td>
<td>328</td>
</tr>
<tr>
<td>10.4.3 Working with activity diagrams</td>
<td>336</td>
</tr>
<tr>
<td>10.4.4 Logging and exception handling</td>
<td>346</td>
</tr>
<tr>
<td>10.5 Developing an example</td>
<td>348</td>
</tr>
<tr>
<td>10.5.1 Creating the collaboration template</td>
<td>349</td>
</tr>
<tr>
<td>10.5.2 Specifying collaboration template definitions</td>
<td>350</td>
</tr>
<tr>
<td>10.5.3 Specifying messages</td>
<td>350</td>
</tr>
<tr>
<td>10.5.4 Creating a scenario</td>
<td>351</td>
</tr>
<tr>
<td>10.5.5 Editing template definitions</td>
<td>367</td>
</tr>
<tr>
<td>10.5.6 Modifying connectors</td>
<td>367</td>
</tr>
<tr>
<td>10.5.7 Creating the collaboration object</td>
<td>368</td>
</tr>
<tr>
<td>10.5.8 Deploying the collaboration and its dependent objects</td>
<td>369</td>
</tr>
<tr>
<td>10.5.9 Testing the collaboration</td>
<td>373</td>
</tr>
<tr>
<td>Chapter 11. Administration tools</td>
<td>379</td>
</tr>
<tr>
<td>11.1 Administrative tools for the developer</td>
<td>380</td>
</tr>
<tr>
<td>11.2 Starting the tools</td>
<td>380</td>
</tr>
<tr>
<td>11.2.1 Starting Interchange Server Express</td>
<td>380</td>
</tr>
<tr>
<td>11.2.2 Starting System Manager</td>
<td>381</td>
</tr>
<tr>
<td>11.2.3 Starting the Flow manager</td>
<td>382</td>
</tr>
<tr>
<td>11.2.4 Setting up to run the Web-based tools</td>
<td>382</td>
</tr>
<tr>
<td>11.2.5 Starting ICSMonitor</td>
<td>384</td>
</tr>
<tr>
<td>11.2.6 Starting Failed Event Manager</td>
<td>385</td>
</tr>
<tr>
<td>11.3 Logging and tracing</td>
<td>386</td>
</tr>
<tr>
<td>11.3.1 Configuring logging and tracing</td>
<td>386</td>
</tr>
<tr>
<td>11.3.2 Generating sample tracing</td>
<td>392</td>
</tr>
<tr>
<td>11.3.3 Viewing the tracing information</td>
<td>393</td>
</tr>
<tr>
<td>11.3.4 Using the Log Viewer to look at trace</td>
<td>394</td>
</tr>
<tr>
<td>11.4 Debugging collaborations</td>
<td>399</td>
</tr>
<tr>
<td>11.4.1 Debugging a success flow</td>
<td>402</td>
</tr>
<tr>
<td>11.4.2 Debugging a failure flow</td>
<td>405</td>
</tr>
<tr>
<td>11.4.3 Finished debugging</td>
<td>406</td>
</tr>
<tr>
<td>11.5 Managing failed events</td>
<td>406</td>
</tr>
<tr>
<td>11.5.1 Using Flow manager</td>
<td>406</td>
</tr>
<tr>
<td>11.5.2 Using the Failed Event Manager</td>
<td>410</td>
</tr>
<tr>
<td>11.5.3 Failed event actions</td>
<td>412</td>
</tr>
<tr>
<td>11.6 Using statistics for system interrogation</td>
<td>413</td>
</tr>
</tbody>
</table>
Part 3. Developing an integration sample

Chapter 12. Sample requirements, design, and specification
12.1 Introduction to the samples
12.2 Business problem description
12.3 Objectives of the solution
12.4 Process modeling
  12.4.1 Integration solution description
  12.4.2 Business modeling using IBM WebSphere Business Integration Workbench
12.5 System integration design
  12.5.1 Methodology - system integration development using Interchange Server
  12.5.2 Sample design
  12.5.3 Sample - Business Objects
  12.5.4 Sample - collaborations

Chapter 13. Sample development
13.1 The sample problem
13.2 Development begins
13.3 Preparing the development environment
  13.3.1 System Manager
  13.3.2 Database and stored procedures
13.4 Adapter configuration
  13.4.1 Creating new adapter instances
  13.4.2 SAL301RConnector
  13.4.3 SAB404RConnector
  13.4.4 PropertyPublishingConnector
  13.4.5 EMailConnector
13.5 SAB404R to SAL301R: employee synchronization
  13.5.1 Business Objects
  13.5.2 Relationships
  13.5.3 Maps
  13.5.4 Collaboration
  13.5.5 Additional information
13.6 SAL301R to SAB404R: property synchronization
  13.6.1 Business Objects
Appendix B. Additional material

Locating the Web material
Using the Web material
  System requirements for downloading the Web material
  How to use the Web material

Abbreviations and acronyms

Related publications
IBM Redbooks
Online resources
How to get IBM Redbooks
Help from IBM

Index
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 on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or 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:

- @server®
- Redbooks (logo) ™
- developerWorks®
- ibm.com®
- iSeries™
- z/OS®
- AIX®
- AS/400®
- ClearCase®
- CrossWorlds®
- CICS®
- Domino®
- DB2 Universal Database™
- DB2®
- ETE™
- IBM®
- IMS™
- Lotus®
- MQSeries®
- Notes®
- OS/390®
- OS/400®
- Passport Advantage®
- Rational®
- Redbooks™
- S/390®
- SupportPac™
- Tivoli®
- 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.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

This IBM® Redbook will help you to understand and use WebSphere Business Integration Server Express. It includes three main parts:

1. “Introduction to business integration” on page 1

   In this part, we introduce topics in the theory and practice of enterprise application integration (EAI). We provide an overview of WebSphere Business Integration Server Express and how it can be used to develop integration solutions.

2. “WebSphere Business Integration Server Express features” on page 97

   In this part, we provide a detailed examination of the integration features and functions provided by WebSphere Business Integration Server Express.

3. “Developing an integration sample” on page 423

   In this part, we implement a sample business integration scenario based on realistic requirements for small and medium-sized customers. We develop a practical example using WebSphere Application Server - Express and WebSphere Business Integration Server Express Plus.

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, Raleigh Center.

Bill Moore is a WebSphere specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches classes on WebSphere and related topics. Before joining the ITSO, Bill was a Senior AIM Consultant at the IBM Transarc laboratory in Sydney, Australia. He has 19 years of application development experience on a wide range of computing platforms and using many different coding languages. He holds a Master of Arts degree in English from the University of Waikato, in Hamilton, New Zealand. His current areas of expertise include application development tools, object-oriented programming and design, and e-business application development.

Knut Inge Buset is a Senior Consultant at ConCrea in Norway. He has seven years of experience in the IT field. He holds a degree in IT from the Norwegian School of Information Technology. His areas of expertise include design and architecture of collaboration, e-commerce, and integration solutions. In the past,
he has worked as a J2EE architect and developer on large projects in various countries in Europe.

**Dennis Higgins** is a Senior Systems Architect in Australia. He has a degree in Applied Science (Computer Science) from the Royal Melbourne Institute of Technology and over 17 years of experience in the IT field with four years of middleware development experience. He has worked for the Australian and New Zealand Banking Group for 8 years. His areas of expertise include gathering requirements, designing, building and deploying middleware solutions in an enterprise.

**Wagner Palmieri** is co-owner of SST IT solutions, an IBM Business Partner in Brazil. For the last eight years, he has been working with systems implementation and integration. Before joining SST, Wagner worked for several years in large companies as a manufacturing manager. He has a Doctorate in Mechanical Engineering from UNICAMP of Brazil.

**Mads Pedersen** is a Senior Consultant with ConCrea AS in Norway. He has over four years of experience in working with enterprise application integration technology. His areas of expertise include design and development of application integration solutions. He holds a Masters degree in information technology from Queensland University of Technology (Brisbane, Australia) and he is a Sun Certified Java™ Developer.

**Joerg Wende** is a Certified Consulting IT Specialist in Germany working for the IBM Software Group. He has 13 years of experience in the IT field. Before joining the Software Group, Joerg worked as an IBM systems engineer and consultant for iSeries™ and AS/400®. He graduated with an engineering degree in Theoretical Electrical Engineering from the Technische Hochschule Ilmenau. His areas of expertise include messaging and business integration, designing and building applications based on WebSphere MQ, WebSphere Business Integration Message Broker, and WebSphere MQ Workflow.

**Alison Wong** is a senior at the Pennsylvania State University School of Information Sciences and Technology. She will graduate in the Spring of 2005 with a Bachelor of Science in Information Sciences and Technology with a Legal Environment of Business minor from the Mary Jean and Frank P. Smeal College of Business Administration. As president of the student organization Women in Information Sciences and Technology, one of her passions is encouraging other women to explore the IT industry as a career path.
Thanks to the following people for their contributions to this project:

Peter Kovari  
International Technical Support Organization, Raleigh Center

Fred Plassman  
IBM Austin

Yury Kosov  
IBM Burlinghame

Glen Bartels  
Mahesh Dodani  
Mark North  
Mike Orth  
Thomas Pack  
Adrian Schechter  
Rick Salmon  
Chandra Venkatapathy  
IBM Raleigh

Richard G. Brown  
IBM UK Hursley
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
Introduction to business integration

In this part, we introduce topics in the theory and practice of enterprise application integration (EAI). We provide an overview of WebSphere Business Integration Server Express and how it can be used to develop integration solutions.
Chapter 1. Introduction

This chapter provides an overview of our redbook and describes how to read the book depending on your interests.
1.1 Welcome to the redbook

Our primary aim for this redbook is to provide a guide for developers using the WebSphere Business Integration Server Express Plus for the first time to facilitate Enterprise Application Integration (EAI). This redbook contains high-level material on EAI topics, WebSphere Business Integration Server Express Plus and how it fits into EAI solutions, before working through a solution to a sample business problem.

The purpose of this chapter is to introduce the topic of the redbook and provide a quick overview of each of the following chapters with a synopsis of the material available. This chapter can be used as a reference to find material spread throughout the book.

The remainder of the book is divided into three parts. The first part introduces WebSphere Business Integration Server Express Plus. This first part of the book discusses why the product exists, its architecture, installing and configuring the product and a getting started with a simple sample development.

The second part delves deeper into the details of WebSphere Business Integration Server Express Plus. There is a chapter discussing the tools that make up the suite available in the product and then each of the components that complete the picture is discussed in depth, using examples of design and development where possible.

In the third part of this redbook, we describe our sample solution from requirements through development. Deployment and administration topics are also discussed.

1.2 Target audience

The main focus of this redbook is developers working in the integration area for business partners, for IBM, or working for small to medium business enterprises (SMB). Solutions needed in this segment often are a good fit for WebSphere Business Integration Server Express Plus.

Material provided in this redbook may help business partners and other developers understand issues related to scoping business solutions.

Much of the material covered is also applicable to enterprise solutions built with the enterprise level WebSphere Business Integration Server product.
This redbook may be of interest to project managers looking to manage the life cycle of implementing integration solutions using WebSphere Business Integration Server Express Plus.

Some material in this book crosses into the systems implementation area and will be of interest to people responsible for the production installation and management of WebSphere Business Integration Server Express Plus solutions.

1.3 Background material on EAI

In Chapter 2, “Integration concepts and solutions” on page 19, we will discuss at a high level the challenges facing businesses using the common scenario of off-the-shelf and custom legacy applications and databases while trying to take advantage of newer services provided by outside agencies through various portals. We discuss the motives behind the drive for integration.

An important aspect of understanding what an integration solution should be like is to form a high-level architectural view of the current systems and to understand the options available. In 2.2, “Integration architectures” on page 22, we discuss design criteria important for an integration project. The section then compares the common integration topology options against these criteria in 2.2.1, “Integration topologies” on page 23.

Patterns are a hot topic in discussions in all IT-related fields. Integration design and build professionals often take advantage of the benefits of using patterns. 2.2.2, “Integration patterns” on page 29 provides a number of useful links relevant to pattern topics and integration before providing a summary of the use of patterns in relation to integration solutions.

This section also provides links to a discussion about Service-Oriented Architecture (SOA) for the reader to gain an understanding of SOA in terms of integration solutions.

How the capabilities of WebSphere Business Integration Server Express Plus fit into the design criteria in this chapter is discussed in 2.3, “Runtime mapping” on page 32. The topology and patterns options applicable to an integration solutions using WebSphere Business Integration are summarized.

1.4 Product overview

The key concepts and architecture of WebSphere Business Integration Server Express Plus are described in Chapter 3, “WebSphere Business Integration Server Express” on page 43.
An overview of the different WebSphere components that make up WebSphere Business Integration Server Express Plus and the different configurations of product level and available add-ons can be found in 3.1, “Key concepts” on page 44. This section discusses the different offerings and how they fit into partners and customers of differing sizes.

3.2, “Integration solution” on page 47 describes at a high level the steps involved in the life cycle of a EAI solution. The focus is on big picture processes and concepts. This section outlines where the WebSphere Business Integration Workbench tool might be used to help with modelling the processes involved in the business being targeted for integration.

A description of the architecture of the WebSphere Business Integration Server Express Plus product can be found in 3.3, “Architecture” on page 48. This section also contains a description of the components and tools packaged or provided as part of the offering:

- 3.3.1, “Supported technologies” on page 49 provides a description of the background components and tools. Descriptions of the technologies and roles of:
  - WebSphere Business Integration software
  - Database Management System
  - IBM Java Developer Kit
  - WebSphere MQ
  - Object Request Broker

- Overviews of the technologies specific to WebSphere Business Integration, Collaborations, Adaptors, Business Objects and Maps can also be found in Chapter 3.

- Tools provided as part of the WebSphere Business Integration Server Express Plus offering are summarized in 3.3.6, “Toolset Express” on page 55.

### 1.5 Installation and configuration

Chapter 4, “Installation and configuration” on page 59 discusses the system requirements for the supported platforms to use WebSphere Business Integration Server Express Plus either for development or implementation of components.

The WebSphere Business Integration Server Express Plus product comes with a Launchpad to coordinate the installation of all of the main components required to install the product. The Launchpad guides the installation through pre-requisite products:
1.6 Getting started

In this chapter, we step through some introductory topics for developers getting started with the WebSphere Business Integration Server Express Plus product.

In the first section, we walk through interacting with the Eclipse-based System Manager tool. See 5.1, “Using the System Manager” on page 76 for information on setting up the main developer System Manager perspective. The initial section also describes setting up the CVS perspective.

One common task that developers will need to become proficient at to use WebSphere Business Integration Server Express Plus is creating instances of connectors. In section 5.2, “Creating a new instance of the JDBC adapter” on page 76, the discussion centers on the configuration steps necessary to create the new instance, including the execution of DDL for the underlying tables and the WebSphere MQ configuration necessary for the connector to run using JMS as a transport.

Creating a project is one of the first tasks that a developer will need to perform. In section 5.3, “Creating a test project” on page 80, a step-by-step description is provided on how to set up a test server, create an integration component library (ICL) and a user project.

The next step is typically setting up connectors. In this section, 5.4, “Creating a new connector” on page 81, the connector instance configured earlier in the chapter is used in creating and configuring a connector within System Manager.

Business Objects are a core component in a integrated solution using WebSphere Business Integration Server Express Plus. In section 5.5, “Creating Business Objects” on page 84 we provide an introduction to working with Business Objects.
The final major building block in creating integration solutions with WebSphere Business Integration Server Express Plus is maps. Section 5.6, “Creating maps” on page 87 gives some initial insights into working with maps.

Having completed some of the pieces of an integration solution, we show how to use Test Connectors to do some incremental unit testing of the configured components. Section 5.8, “Testing the connector” on page 90 shows how to set up the Test Connector to see events occurring when changes to the underlying database are made.

The final section in this chapter 5.9, “Helpful resources” on page 95 provides a number of links of interest to developers looking at building integration solutions.

### 1.7 WebSphere Business Integration toolset

Chapter 6, “Introduction to Toolset Express” on page 99 gets into the detail of the tools provided with WebSphere Business Integration Server Express Plus. The chapter gives a general overview and then delves in some detail with the specific tools. Provided is a detailed discussion of where the tool fits in the process and what the tool options are. The tools discussed are:

- **Business Object Designer** on 6.2, “Business Object Designer” on page 103, discusses the general process of building Business Objects either directly or via the Object Discovery Agent, or ODA. This section describes defining the details of fields and the configuration of the verbs that the object supports. Information about configuring the ODA agents is provided. Chapter 8, “Business Objects” on page 209 has more details about Business Objects and has examples of using this tool.

- **Map Designer Express** on 6.3, “Map Designer Express” on page 107 introduces the tools for defining maps. Maps are the facility to transform information from application-specific format to generic format. The chapter describes how the tool can be used to do the mapping, showing examples of direct field mapping as well as splitting and joining fields. This section also describes the general process of testing. More details of mapping including examples can be found in Chapter 9, “Mapping” on page 247.

- **Connector Configurator Express** on 6.4, “Connector Configurator Express” on page 119 describes in general the connector configuration tool. Connectors are the technology that provide the bridge between the application and the Interchange Server Express platform. This section overviews configuration of generic settings and connector-specific settings. More details of connectors and the adaptor framework can be found in Chapter 7, “Adapters” on page 159 which describes in detail the technology of connectors, the role of
application-specific connectors and the configuration of specific examples including e-mail, JMS and Web services.

- The section 6.5, “Test Connector” on page 126 describes the general use of the Test Connector. The Test Connector is a valuable tool that allows the testing of elements in the integration solution without the need to have the application available or fully configured. This is especially valuable where application adapters are used as the developer often does not have access to the full product being integrated. You will find many references to the Test Connector throughout the book. Your first experience with this technical component may well have been in performing the installation verification as described in the first steps document accessed by clicking

  start → Programs → IBM WebSphere Business Integration Express → First Steps.

- One of the challenges in providing integrated solutions is the different views of data held between different systems. The Relationship Designer Express tool described in 6.6, “Relationship Designer Express” on page 130 gives an explanation of the tool and how it is used to manage participants and attributes to provide the technology that manages disparate data views.

- Process Designer Express is the tool for managing Collaborations, the component that holds business logic. 6.7, “Process Designer Express” on page 136 showcases the workings of the Process Designer Express tool. Showing aspects of Collaborations and how scenarios are created and updated and other aspects such as Ports are defined.

- The Eclipse-based System Manager provides an integrated test environment. How to set up and use this test environment is discussed in 6.8, “Test Environment” on page 148.

### 1.8 Adaptors

Chapter 7, “Adapters” on page 159 describes Adapters used by WebSphere Business Integration Server Express Plus to interface to the Interchange Server Express. As discussed in this chapter, the primary feature of the adapter is a connector that allows WebSphere Business Integration Server Express to exchange Business Objects with applications that send or receive data. These connectors allow business applications to send and receive business data and events.

Connectors consist of an application-specific component and the connector framework. The chapter discusses the architecture of connectors and delves into details of installation and configuration. The chapter describes technology adaptors and application adaptors and what is provided in the capacity packs. The role of the event and archive tables is examined.
It provides in 7.3, “How to implement an adapter” on page 172 a detailed worked example using the JDBC connector. This section describes the end-to-end process of using the JDBC connector from installation, including configuration such as setting up subscriptions, through to deployment. Testing of the adapter is documented in Chapter 13, “Sample development” on page 469.

Chapter 7 includes information about Data handlers in 7.4, “Data handlers” on page 203, the adapter development kit in 7.5, “Adapter Development Kit” on page 206 and the role of adapters in the usage and implementation of ODAs in 7.6, “ODAs” on page 207.

1.9 Business Objects

Section 8.1, “Introduction to Business Objects” on page 210 introduces the topic of Business Objects. This chapter discusses Application Specific Business Objects (ASBO); Generic Business Objects (GBO) and the roles and differences of each of these. The chapter also provides a description on Business Object definitions and what the differences are.

Section 8.1.1, “Business Object definitions” on page 211 gets into the high level detail of Business Objects, and contrasts with an example the difference between ASBOs and GBOs. The structure of the Business Object is described, showing the role of the Business Object header, attributes and verbs. A description is provided for the attributes name, type, key, foreign key, required attribute, cardinality, maximum length, default value, application-specific information and the of the ObjectEventId. The role of Business Object verbs such as create are described.

Section 8.1.2, “Business Objects” on page 218 goes into more detail about GBO and ASBO and discusses flat Business Objects and hierarchical Business Objects. This section gives a Business Object overview view of the life cycle of objects and of the Business Object flow through the WebSphere Business Integration Server Express.

In section 8.2, “Designing Business Object definitions” on page 221 we discuss the requirements of a design method and the kinds of documentation artifacts you should consider using when undertaking an integration project. The chapter provides a guide to the topics that should be addressed when producing your design documentation. It discusses the steps to be taken during the design phase of the project. This section would be of interest to project managers and designers and may also aid in providing a key into how to scope for costing an integration project using WebSphere Business Integration Server Express Plus.
Under section 8.2.2, “Standards” on page 222 we give some good practice suggestions for the naming and structuring of Business Objects.

Heading 8.3, “Developing Business Object definitions” on page 223 is of interest to developers as it gets into heavy detail in the process of creating Business Objects. This section details using the Business Object Designer Express tool to develop a Business Object from scratch.

Developing Business Objects is continued in 8.4, “Developing with Object Discovery Agents” on page 228. In this section there is detail about creating a Business Object with a worked example using the JDBCODA and XMLODA, which will probably be commonly used techniques for creating Business Objects.

The chapter continues with a description of deploying in 8.5, “Deploying Business Object definitions” on page 240 and an overview of the packaged Business Object definitions provided as part of the expansion packs. More details of these pre-packaged objects is provided in 10.2.2, “Collaboration capacity packs” on page 321.

The Business Objects available by default when WebSphere Business Integration Server Express Plus is installed are probed in 8.6, “Using packaged Business Object definitions” on page 241 with an summarized explanation of the objects, cross-referenced with the child objects and the collaborations that use them. The Customer Business Object hierarchy is explored in more detail.

In 8.7, “JDBC application-specific information” on page 244, we discuss the details of Business Objects, relating to the JDBC ODA:

▶ TN - table name
▶ SCN - status column
▶ CN - column name
▶ FK - parent/child relationship
▶ CONTAINMENT - parent/child relationship
▶ KEEP_RELATIONSHIP - parent/child relationship

1.10 Mapping and relationships

In Chapter 9, “Mapping” on page 247 we discuss the role of mapping and relationships in WebSphere Business Integration Server Express Plus.

Mapping is the process of transforming data from one format to another. Mapping is central to the process of transferring information between different applications, and for providing business processes that are independent of specific applications.
In 9.1.1, “Developing maps” on page 248, there is a description of the process of developing maps using Map Designer Express. The section describes the process of mapping individual attributes of the source Business Object to the target Business Object using standard transformations such as Set Value, Move, Join and Split, along with sub-maps. This section also discusses cross-references and Custom transformations using the Activity Editor. Also in this section, we provide a description of interacting with the tool with hints about how to develop the specifications for maps prior to development.

Section 9.1.2, “Defining a simple map” on page 252 walks through an example of using the Map Designer Express wizard to create a map. This section also discusses testing the map using the example built in this chapter.

In 9.1.3, “Creating advanced mapping” on page 261 we work with a more complex example to show how attributes can be manipulated to meet business requirements. Complex mapping is often required where ASBO needs for one application are different from that of a different application.

An ever increasingly relevant topic is the integration with Web services; 9.1.4, “Using Web services” on page 275 describes the process of integrating WebSphere Business Integration Maps with Web services.

Maintaining relationships between key values of the participating Business Objects or setting attribute content based on a lookup table is one of the key requirements of integration projects. Section 9.1.5, “Relationship mappings” on page 279 describes static and dynamic relationships you can define using the Map Designer Express tool. This section provides an example of setting up a static lookup relationship.

More information about relationships is provided in 9.2, “Relationships” on page 286 where the discussion explores the challenges of having different representations of the same business data in different related objects. The meaning of participants is explained and the differences and roles lookup, identity and non-identity relationships are examined. A more in depth sample is developed in 9.2.1, “Relationship development” on page 288, where the roles of database tables, stored procedures and other details are shown.

In 9.1.6, “Map documentation” on page 282 we describe the process of creating documentation from the Map Designer Express tool. This technique can provide a valuable cross-checking method, or tool to generate review material.
1.11 Collaborations

In Chapter 10, “Collaborations” on page 309 we discuss collaborations and provide a high level overview before delving deeper into more details and working some examples.

The Collaborations chapter discusses the role of Business Objects with Collaborations and describes with business scenarios, the kinds roles that collaborations play.

In 10.1.1, “Collaboration components” on page 311 Collaboration templates, Collaboration objects, Collaboration groups, Ports, Scenarios and subdiagrams are introduced and described in high-level terms. The description of Ports provides an explanation of how Business Object types and verbs interact with collaborations and shows how these can be configured with the product tools to do port binding.

Different mechanisms by which collaborations get triggered and interact with applications are discussed in 10.1.2, “Collaboration processing” on page 315. The different mechanisms for triggering a collaboration are: a connector in a publish-and-subscribe interaction, an access request and a service call. In this section, we also provide an overview of long-lived business processes.

In 10.2, “Collaborations in WebSphere Business Integration Server Express” on page 318 we describe collaborations in WebSphere Business Integration Server Express Plus and the Collaboration capacity packs. We discuss the types of collaborations: Synchronization Collaborations, Wrapper Collaborations and Process collaborations.

Section 10.2.1, “Base collaboration templates” on page 318 examines in more detail the collaborations provided in WebSphere Business Integration Server Express Plus. The Foundation Collaboration and Wrapper Foundation capabilities are explored before describing how to use an existing collaboration template to create your own templates.

With WebSphere Business Integration Server Express Plus, there is the option to purchase pre-packaged additional collaborations. These are called Capacity Packs and an introduction to the collaborations available in these Packs is described in 10.2.2, “Collaboration capacity packs” on page 321. The discussion includes instructions on how to install a Capacity Pack.

We provide some considerations on designing collaborations in the section 10.3, “Designing collaborations” on page 323, touching on topics of methodology and standards.
The rest of the chapter is devoted to aspects of developing collaborations, with 10.4.1, “Overview of Process Designer Express” on page 326 providing an overview of how to use the Process Designer Express tool. In 10.5, “Developing an example” on page 348 an example is provided that steps through:

- Creating collaboration templates.
- Specifying template definitions.
- Specifying messages.
- Creating a scenario.
- Editing template definitions.
- Compiling templates.
- Modifying connectors.
- Creating the collaboration object.
- Deploying the collaboration and its dependent objects.
- Testing the collaboration by showing how to set up Test Connectors to perform the tests.

### 1.12 Administration tools

In Chapter 11, “Administration tools” on page 379 we discuss the tools available in WebSphere Business Integration Server Express Plus to help developers and others interact with the various components on the Server.

In section 11.1, “Administrative tools for the developer” on page 380, there is an overview of the tools available categorized by tool type.

Section 11.2, “Starting the tools” on page 380 provides a single place where there is a description on how to start all the administration tools. Included in this section are the tasks necessary to get access to the Web-based tools. The Web based tools require some additional configuration prior to being used and gain you access to the Interchange Server via the WebSphere Interchange Server System Monitor.

The discussion centers on how to configure the tools to provide logging and tracing information in section 11.4, “Debugging collaborations” on page 399, and how to use the tools to view the output of the logging and tracing. This section uses the system verification sample to describe how logging is configured and managed, as well as the sorts of data provided by logging.

In section 11.5, “Managing failed events” on page 406, the debugging tools for collaborations are illustrated in an example and the kinds of interactions possible are described.

In section 11.6, “Using statistics for system interrogation” on page 413 we use the two administrator tools to manage failed events. We use the installation
verification scenario again to demonstrate how these tools can be used by causing failure events and then resolving them.

In section 11.7, “Starting and stopping components” on page 420 we use both WebSphere Interchange Server System Monitor and System Manager to manage the state of the various system components.

1.13 Sample requirements, design, and specification

In 12.1, “Introduction to the samples” on page 426 we discuss the motives behind the desire to integrate systems. We discuss factors that are important in any integration solution. Using business examples, we look at the kinds of scenarios that can lead to the need for integration.

The discussion then proceeds to show where WebSphere Business Integration Server Express Plus provides value for performing systems integration in relation to the SMB market.

We organize the process into the following phases:

► Business problem description

In 12.2, “Business problem description” on page 427 the existing SAL301RRealty Web application and Agency Legacy system are described with the use of each explained. The business processes involving external agencies, such as the placing of advertisements in newspapers is also examined. The largely manual as-is business processing model is described.

► Objectives of the solution

This section, 12.3, “Objectives of the solution” on page 429, extracts the fundamental business requirements to be addressed by the integration solution. In a real project this list would be prioritized and delivered in a number of phases. The sample produced in Chapter 13, “Sample development” on page 469 implements only a small number of these requirements.

► Process modelling

In the next section, 12.4.1, “Integration solution description” on page 430, we work through the architectural design process, examining the business processes and activities that make good candidates for integration. The section discusses future processes which come out of the design scenario.

The use of the WebSphere Business Integration Workbench tool included with WebSphere Business Integration Server Express Plus is discussed in 12.4.2, “Business modeling using IBM WebSphere Business Integration
WebSphere Business Integration Workbench is used to produce the business processing models in section 12.4.3, “Business process model” on page 438. There are models for each of the major business processes identified in the earlier design phases.

In the final section of the requirements, the design of the solution is discussed. In section 12.5, “System integration design” on page 448, it is decided to use the WebSphere Business Integration Server Express Plus product to implement the solution and the reasons why the product is a good fit for the technical requirements.

There is a section summarizing the design and build methodology, 12.5.1, “Methodology - system integration development using Interchange Server” on page 450, that discusses the order and steps undertaken with integration style projects using WebSphere Business Integration Server Express Plus.

In section 12.5.2, “Sample design” on page 451, we position WebSphere Business Integration Server Express Plus in the design and choose the technology adapters for integrating the systems into the overall design, and define the collaborations necessary. This section includes Figure 12-18 on page 452 that brings together all the components of the system into one diagram.

The design then explores the Business Objects and mapping details required in 12.5.3, “Sample - Business Objects” on page 454 and the collaboration details required in 12.5.4, “Sample - collaborations” on page 464.

1.14 Sample development

In this chapter we discuss using WebSphere Business Integration Server Express Plus for development. We provide examples from the development effort and the types of things we did to learn about the product and produce the final example.

It was a common observation that there were tasks to be performed that would take ten minutes the second time around after the ten hours spent doing it the first time. The product is conceptually easy to understand from a high level, but can be daunting when trying to produce a functioning system. In this chapter we hope to provide some insights that may reduce the steepness of the learning curve.

Chapter 13, “Sample development” on page 469 begins by reiterating the business problem in developer terms then steps through the development effort:
The first tasks are to create the projects and set up the Interchange Server instance. This is described in 13.3, “Preparing the development environment” on page 471.

The adapters needed for the solution to be configured are as shown in 13.4, “Adapter configuration” on page 477.

Section 13.5, “SAB404R to SAL301R: employee synchronization” on page 499 describes the development of the employee synchronization from the SAB404R to SAL301R systems.

The synchronization between properties in SAL301R and inventory in SAB404R including the integration with Web Services and e-mail development is worked through in 13.6, “SAL301R to SAB404R: property synchronization” on page 548.

There is additional miscellaneous material discovered in the process of learning about the capabilities of the product and developing the solution we thought might be of interest to developers. This material is covered in:

- Section 13.8, “Additional material, tips and tricks” on page 592 contains miscellaneous material that may be of interest.

### 1.15 Installing the sample system and other details

The material contained in Appendix A, “Deploying the redbook sample code” on page 603 describes all the steps necessary to install the sample application up to the point where it can be tested with all components complete. This is, in effect, the answers in the back of the book to the problem discussed in the requirements and then coded in the sample development chapter.

Appendix A, “Deploying the redbook sample code” on page 603 defines the prerequisite steps prior to installing the sample application, then goes into detail describing getting the sample solution into your development environment.

See Appendix B, “Additional material” on page 643 for instructions on obtaining our sample code.
Integration concepts and solutions

This chapter provides a high-level overview of the problems and challenges inherent in building EAI solutions. We discuss best practices and provide advice for all levels of an EAI solution including design, implementation and deployment. We discuss types of application integration, such as data level (information) integration, application logic integration and process integration, and consider different EAI architectures such as brokers or point-to-point integration.
2.1 Overview

Integration is not a new concept or requirement, since most enterprises have to use more than one application to support their daily business and therefore need to exchange data between applications or between different steps of the same application. Reasons for integration include the following:

► Files with many different characteristics have been used in the past and are still used.

► Many enterprises have selected or developed most of their applications to run on a single application server, like IBM System 390, iSeries or other third-party systems. Transaction systems like IMS™ or CICS® and databases were widely used to minimize the integration efforts between different applications running on the same system.

► Even with the high capabilities of a single application server, business requirements for integration across systems remained high. Decentralized departmental solutions, distributed warehouse management systems, and supplier integrations based on EDI are examples of the needs that drove integration requirements beyond the confines of the single server solution.

► Many application development or purchasing decisions were made from a business point of view, without strong focus on integration with existing systems. In many cases, applications needed a special runtime platform different from the ones currently in use.

► Further integration requirements developed as a consequence of mergers and acquisitions, because finding the same application in both of the merging partners was not very likely.

It is common to find application and systems interconnected as shown in Figure 2-1 on page 21.
Great efforts have been made in the area of Enterprise Resource Planning (ERP) suites to develop systems that fit all requirements of a modern business. But none of these suites has been able so far to address all business problems. On the other hand, application vendors have been able to provide applications with a high focus on special business needs, such as Customer Relationship Management (CRM) or Supply Change Management (SCM) to name some of the most common ones. Therefore, it is not unusual to find many different applications within a single enterprise, as shown in Figure 2-2 on page 22.
The business drivers behind the current focus on integration include the following:

- Changing business needs requiring a much faster response:
  - Driven by the market, by customers, and certainly by competitors.
- Need for improving operational efficiency:
  - To raise customer satisfaction.
  - To reduce operational costs.
  - To increase revenue.
- Increase in revenue generation and improved growth.
- Long-term cost savings.
- Further expansion by mergers and acquisitions.
- Government regulations
  - Sarbans-Oxley, Basel II.
  - RFID, EDI.

IT addresses these business drivers using different initiatives like e-business, e-commerce, supply chain integration and straight through processing.

### 2.2 Integration architectures

In this section, we deconstruct typical integration scenarios so that you can assess the building blocks needed to define a solution. This assessment is independent from any vendor-specific implementations. It is intended to give you an idea of what kind of technology might be appropriate for a given use case.
We also explain how WebSphere Business Integration Server Express helps you address these integration issues. More details can be found in 2.3, “Runtime mapping” on page 32.

2.2.1 Integration topologies

First, we will take a closer look at some basic integration topologies:

- Point-to-point topology (see “Point-to-point” on page 24).
- Hub and spoke topology (see “Hub and spoke” on page 25).
- Bus topology (see “Bus topology” on page 26).

To get a better understanding of advantages and disadvantages of each topology, we introduce some vital design criteria. As shown in Table 2-1, there are a number of questions you have to ask your client or yourself, if you are going to analyze and design an integration solution.

Table 2-1 Design criteria

<table>
<thead>
<tr>
<th>Design criteria</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of applications</td>
<td>▶ How many applications are part of the integration scenario?</td>
</tr>
<tr>
<td></td>
<td>▶ What is their degree of interconnections?</td>
</tr>
<tr>
<td>Routing rules</td>
<td>▶ Static or dynamic routing: are there fixed rules or is there a dynamic selection of the destination applications?</td>
</tr>
<tr>
<td></td>
<td>▶ What is the desired administration model, central or distributed?</td>
</tr>
<tr>
<td>Data transformation</td>
<td>▶ Do you have to transform the data to comply with each application and where should the transformation take place?</td>
</tr>
<tr>
<td></td>
<td>▶ Is the use of a standard form required or desired?</td>
</tr>
<tr>
<td></td>
<td>▶ What is the desired administration model for transformation rules, central or distributed?</td>
</tr>
<tr>
<td>Process logic</td>
<td>▶ Is there any logic besides the core application business?</td>
</tr>
<tr>
<td>Scalability</td>
<td>▶ What are the expected numbers for throughput and latency? Should the system be designed for average or peak numbers?</td>
</tr>
<tr>
<td>Operational considerations</td>
<td>▶ How can you operate the infrastructure?</td>
</tr>
<tr>
<td>Manageability</td>
<td>▶ Do you need a central administration?</td>
</tr>
</tbody>
</table>
Important: All of the following topologies are basic patterns. In practice, you will find several variations and mixed implementations.

Point-to-point
Point-to-point integration, as shown in Figure 2-3, is the most used and the oldest topology. In all probability, most of you have already had to develop an interface between application X and application Y.

If you apply our design criteria to this topology, you find that:

- The number of applications should be equal to or less than three, otherwise the number of interfaces increases dramatically. If you have $N$ applications and all are connected with each other, you have to develop and maintain $N \times (N-1)$ adapters. In many cases, you have to develop multiple interfaces for the same application.

- All transformations are done by the application or the adapter. Only in raw circumstances do you have any two applications speaking the same data format.

- Routing is done explicitly by the adapter.

- All process logic is embedded, more often than not hard coded, in adapter logic.

- Scalability is limited.

- Management is distributed.

- All applications and adapters must be protocol and format aware. If there is one change, all participants have to change at the same time.

- Thick adapters have to run a good deal of code.

- It is suitable for a more or less static environment.
You will see, later on, how to overcome some of the limitations of the basic point-to-point topology.

**Hub and spoke**

In the hub and spoke topology, a central hub, sometimes called the broker or integration server, provides a central instance for transforming and routing the data between the participating applications (see Figure 2-4). Each application must only connect to the hub, simplifying transport and providing room for standardization. Removing logic (transformation, routing and process) from the adapter not only reduces the effort of building the adapter, but also reduces the burden on the application server.

![Hub and spoke topology](image)

The hub and spoke topology fits with our design criteria so that the following may apply:

- The number of applications can now be much larger than three.

  *Note:* Even if you have a broker infrastructure in place, you can use it for point-to-point integration as well.

- The number of interfaces is equal to the number of applications.
- Transformation is done by the hub and centrally managed.
- Routing is done by the hub.
- Process logic may be delegated to the hub.
- Scalability is good; many hub implementations allow vertical and horizontal scaling.
- The management of routing, process logic and transformation is centralized, and therefore applications are decoupled in these terms.
Adapters can be simpler, connection-oriented programs. They can be based on a standard, reusable framework.

This is the best setup for dynamic, frequently changed environments.

There are some important design considerations for the hub layout:

- Have redundancy for high availability
- Provide multiple instances for scalability
- A combination of both

**Important:** Integration servers may provide additional capabilities, like data enrichment, automatic cross-referencing between different participants, or transaction control and compensation.

**Note:** If you do not introduce a generic data model for the hub, all transformations are still point-to-point. A generic data model not only provides an additional abstraction between the source and destination applications, it is also the fundamental part in building generic process models.

**Bus topology**

A bus topology is built, if all applications are connected to a distributed infrastructure, based on a common transport protocol and a normalized or intermediate data format (see Figure 2-5).

The characteristics of a bus-based design are as follows:

- The number of applications is larger than three with a high degree of interconnections.
- Transformations to normalized data format are done by the adapter or application.
The routing is implicit, based on adapter/application criteria. The sending application has to address the receiver(s); the bus will find the target applications based on this logical or physical address.

Process logic is done by the adapter logic.

Scalability is good.

There is no centralized management.

All applications are decoupled.

All applications/adapters must be bus protocol aware.

The physical bus layout is important for scalability.

Theoretically, every application can see every message, therefore the security design can be critical.

**Mixed topologies**

You can design solutions by combining the basic topologies so that you have thin adapters, converting application data in a intermediate representation, and you may then choose to:

- Address the target application, if there is no need for transformation and process logic.
- Address the hub, for process logic, transformation and routing.

See Figure 2-6 on page 28 for an example of this combined topology.
The concept of the Enterprise Service Bus (ESB) is a further development of this basic scenario, leveraging advanced technologies that use:

- Open data formats and format descriptions based on XML and XML schemas (XSD).
- Common open transport technologies, based on advanced Web service transports.
- Common interface descriptions, based on the Web Services Description Language (WSDL).

**Tip:** To review the current state of the discussion of ESB related technology, check the following Web sites and documents:

- IBM developerWorks®
- Understand Enterprise Service Bus scenarios and solutions in Service-Oriented Architecture
- The IBM Redbook *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346
**Coupling techniques**

When business integration solutions are being considered, a frequent topic of discussion is whether to select synchronous or asynchronous coupling techniques. Table 2-2 looks at the respective advantages and disadvantages of these techniques against the background of their use in integration projects.

<table>
<thead>
<tr>
<th>Synchronous coupling</th>
<th>Asynchronous coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>– Theoretically requires less concept work.</td>
<td>– Loose coupling between applications.</td>
</tr>
<tr>
<td>– Theoretically works with distributed two-phase-commit.</td>
<td>– Very good scalability, because every layer scales independently from others.</td>
</tr>
<tr>
<td>– Allows a single runtime environment.</td>
<td>– Request- and reply-routing and transformation can run on independent systems.</td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>– All applications tightly linked together.</td>
<td>– No single point of failure on the critical path.</td>
</tr>
<tr>
<td>– Hard to scale.</td>
<td>– Implicit high availability when implemented with software and hardware-clustering.</td>
</tr>
<tr>
<td>– Hard to operate.</td>
<td>– Systems management software available.</td>
</tr>
<tr>
<td>– No systems management tools available for some kinds of implementation.</td>
<td>– Offers additional QoS when assured application access is needed.</td>
</tr>
<tr>
<td>– No assured delivery, if one participant does not conform to Open/XA.</td>
<td></td>
</tr>
</tbody>
</table>

The real question to answer is: do you need to have a tight or a loose coupling between your applications? There is no easy answer.

### 2.2.2 Integration patterns

Deconstructing integration aspects into patterns is a very useful and proven approach that allows you to take advantage of basic structures, representing best practices and lessons learned from many integration projects. For an overview of patterns methodology, we suggest you read the following Redbooks:
IBM Patterns for e-business distinguishes between business patterns, describing the business purpose of a solution, and integration patterns, enabling the interaction between the business patterns. The patterns with strong integration aspects are shown in Figure 2-7.

- Access Integration: The Access Integration pattern describes those recurring designs that enable access to one or more business patterns. In particular, this pattern enables access from multiple channels (devices) and integrates the common services required to support a consistent user interface.

- Extended Enterprise: The Extended Enterprise business pattern (also known as Business-to-Business or B2B) addresses the interactions and collaborations between business processes in separate enterprises. This pattern can be observed in solutions that implement programmatic interfaces to connect inter-enterprise applications.

- Process Integration: The Process Integration pattern enables companies to connect people, processes, and applications across and beyond their enterprise. These solutions make it possible to leverage existing IT
investments while providing the flexibility to adapt quickly to changing business conditions and emerging technologies.

**Note:** Process Integration is not explicitly named in IBM Patterns for e-business as a basic pattern on its own, but, for the purposes of our redbook, it makes sense to distinguish between broker-oriented integration and process-oriented integration as separate layers.

- **Application Integration:** The Application Integration pattern brings together multiple applications and information sources without the user directly invoking them. This pattern is most effectively applied when developmental efforts require the seamless execution of multiple applications and access to their respective data in order to automate a complex, new business function.

**Note:** Instead of using the Process Integration pattern, you may also distinguish between:

- Process-focused application integration: the integration of the functional flow of processing between the applications.
- Data-focused application integration: the logical integration of the information used by applications.

- **Information Integration:** The Information Integration pattern brings together multiple data sources so users can extract useful information from large volumes of data, text, images, and so on, or where integration is carried out directly between data sources, for example using replication technologies.

Now we need consider how to use these patterns to construct an integration platform. Each pattern is composed of common services which are selectable by the integration developer. For example, User-to-Business services give users personalized access to the content and transactions of the enterprise, Business-to-Business services allows enterprises to expand their value chain, and internal process services enable the enterprise to automate their business processes. Figure 2-8 on page 32 shows a layered view of these services.
As you can see, the services provided by process and application integration are the key services for an integration platform. In 2.3, “Runtime mapping” on page 32, we discuss how to map these services to runtime and products.

### 2.3 Runtime mapping

Now we look at how to map integration services to runtime server instances. To simplify our discussion, we do not consider any decomposition into physical nodes and network structures. Figure 2-9 shows a runtime mapping for an integration solution.

The last step is to correlate real products with the runtime server instances. The product mapping shown in Figure 2-10 on page 33 uses WebSphere Express products. We do not dictate a particular runtime platform (the Express family runs on a variety of platforms), but most customers will have a preferred one.
The products in the mapping are:

- **WebSphere Business Integration Server Express**
  
  For a product overview, see “WebSphere Business Integration Server Express” on page 34.

- **WebSphere Portal Server Express**
  
  For a product overview, see “WebSphere Portal - Express” on page 34.

- **WebSphere Business Integration Connect - Express**
  
  For a product overview, see “WebSphere Business Integration Connect - Express” on page 35.

- **WebSphere Application Server - Express**
  
  For a product overview, see “IBM WebSphere Application Server - Express V5.1” on page 36.

- **WebSphere Commerce - Express**
  
  For a product overview, see “IBM WebSphere Commerce - Express” on page 37.

- **WebSphere MQ Express**
  
  For a product overview, see “WebSphere MQ Express” on page 38.

You can find more detailed information about the WebSphere Express family on the Internet at:

**WebSphere Business Integration Server Express**

For detailed information on WebSphere Business Integration Server Express, see:

http://ibm.com/software/integration/wbiserverexpress/

WebSphere Business Integration Server Express helps medium businesses integrate their applications quickly and easily to better serve customers, partners and suppliers, because it:

- Can be rapidly deployed.
- Scales as your business grows.
- Delivers lower total cost of ownership.
- Is easy to install and use.
- Includes a rich set of tools that allows you to easily map business processes to underlying IT applications with little or no programming.
- Delivers a comprehensive set of features including a wide range of prepackaged adapters that allows you to connect with diverse back-end systems and applications and customizable process templates.

WebSphere Business Integration Server Express supports open standards including Web services. This on demand offering runs on Windows®, Linux® and OS/400®.

**WebSphere Portal - Express**

For detailed information on WebSphere Portal - Express, see:

http://ibm.com/software/genservers/portalexpress/

WebSphere Portal - Express and WebSphere Portal - Express Plus have been designed for the needs of small and medium-sized businesses, as well as departments within larger companies. With WebSphere Portal - Express, you can deploy portals for your employees, trading partners, and customers on which the content, applications, portal page organization, and graphic appearance have been customized for each user's needs and preferred work patterns. This means critical business information and applications will be much easier to locate and use. This can help employees become more productive, improve service to trading partners, and attract new customers to your site. WebSphere Portal - Express Plus has all the features of WebSphere Portal - Express and adds features that help teams of individuals work together more effectively on projects. Also, WebSphere Portal - Express and WebSphere Portal - Express Plus are available with a choice of user-based licensing or processor-based licensing. User-based licensing and processor-based licensing can be used on an extranet, intranet or Internet portal. You can choose the licensing option that best fits your budget.
WebSphere Portal - Express provides a single unified workspace in which your employees, trading partners, and customers can easily find and use the applications and information that they need. Its features include:

- A portal user interface that can be customized to match each portal user's specific needs.
- A portal framework which allows you to integrate your applications and information sources into the portal.
- Easy to use interfaces for performing portal administration tasks such as controlling portal membership and assigning portal user privileges.
- Portlets for accessing applications such as e-mail, calendars, and syndicated news.
- Authenticated single sign-on, which allows users to log on to multiple portal applications without the need to sign on to each application individually.
- Integrated document management.

WebSphere® Portal - Express Plus includes all the functions of WebSphere Portal - Express and adds:

- Team workspaces that create online environments which help team members work together more effectively.
- Instant messaging for real time collaboration with your customers, trading partners, and teammates.
- A team calendar to keep track of a team's shared calendar events in the team workspace.
- Management of projects, assignment of tasks, and tracking of milestones for specific projects or initiatives.
- Display of the online status for portal user names that appear in portlets. To initiate a chat, just right-click the name of a user who is online and choose Chat.

**WebSphere Business Integration Connect - Express**

For detailed information on WebSphere Business Integration Connect - Express, see:  


WebSphere Business Integration Connect - Express provides a solution to an enterprise's business-to-business(B2B) needs, comprising software that can be extended by optional Community Integration Services. Trading partner integration is critical to enhancing the value of an enterprise's internal integration, and will involve connecting to any number of partners using a variety of different technologies and standards. WebSphere Business Integration Connect provides
a flexible and scalable architecture supporting multi-protocol connectivity, including solutions for AS2 and RosettaNet with sophisticated browser-based tools to provide a rapid partner on-boarding capability as well as complete visibility to all aspects of the operational requirements of Community Integration.

WebSphere Business Integration Connect - Express is a light-weight, easy to use, cost-effective B2B connectivity tool, leveraging AS2 and HTTP standards for transmitting documents securely over the Internet.

**IBM WebSphere Application Server - Express V5.1**

For detailed information on WebSphere Application Server - Express, see: http://ibm.com/software/webservers/appserv/express/

IBM WebSphere Application Server - Express V5.1 offers fast development, deployment, and management of dynamic Web experiences that engage customers and empower employees.

With smaller budgets and limited IT resources, small- and medium-sized companies face special challenges getting started in e-business. Designed to meet the unique needs of these companies, IBM WebSphere Application Server - Express V5.1 combines the power and function of an industry leading application server with a smaller, up-front investment, making WebSphere Application Server - Express V5.1 a low-risk, affordable entry to e-business.

Supporting the latest specifications for JavaServer Pages, Java Servlets and Web services, WebSphere Application Server - Express can help you build dynamic Web sites that improve customer service and employee efficiency and which run on popular servers, including iSeries and popular operating systems such as Linux, UNIX® (AIX®, Solaris, HP-UX) and Windows. You can view information in databases, perform simple updates, and create and use Web services.

IBM WebSphere Application Server - Express V5.1 supports the following key capabilities and benefits:

- **Ready to go**
  - Provides the tools necessary to create and run a simple, dynamic Web site in a tightly integrated and affordable package that includes a Web application server, a development environment based on IBM WebSphere Studio, and wizards and samples that can help reduce complexity and deployment costs.
  - Includes a set of integrated applications for iSeries customers that can boost employee and business professional productivity.
Easy to use

- Delivers an out-of-the-box solution for building static and dynamic Web sites, new simplified database access and rich user interface controls for rapid Web application development.
- Includes a quick, wizard-driven installation, only five clicks to install.
- Provides one-click application assembly and deployment that can help improve productivity.
- Includes a browser-based interface that can help minimize administration requirements.

Migration ready

Protects your investment as you grow to other IBM WebSphere Application Server V5.0 and IBM WebSphere Studio V5.0 configurations, when you require more advanced development and deployment capabilities.

**IBM WebSphere Commerce - Express**

For detailed information on WebSphere Commerce - Express, see:

http://ibm.com/software/genservers/commerce/express/

IBM WebSphere Commerce - Express is an easily installed, affordable, yet complete solution designed to help growing mid-market companies do business on the Web. Jump-start your online presence with the basics, then grow with the industry leader as your needs change and grow. WebSphere Commerce - Express can support both business-to-business (B2B) and business-to-consumer (B2C) needs and is:

- Affordable but complete, with all-in-one licensing that gives small and medium business (SMB) customers everything they need for production, staging and site development at one attractive price.
- Quickly and easily installed, allowing customers or partners to create and start customizing a store site right away with its nimble, fast-loading development tools.
- Ready to go with the capabilities required to get started out-of-the-box, while providing many advanced features to grow into over time.
- Easy to administer, with highly usable tools that let mid-market customers manage their sites effectively without special technical skills.
- Built on open standards, freeing companies from proprietary platform restrictions.

WebSphere Commerce - Express is designed to support the key platforms required by most mid-market customers, which is a subset of the platforms supported by the overall WebSphere Commerce family. Supported platforms
include: Microsoft® Windows 2000 or 2003, Linux-Intel (SUSE LINUX and RedHat) and IBM @server® iSeries.

**WebSphere MQ Express**

For detailed information about WebSphere MQ Express, see:

http://ibm.com/software/integration/wmq/express/

With WebSphere MQ Express, business information can be moved safely and securely to wherever you may need it within or outside your business without writing complex networking code. WebSphere MQ Express not only speeds up development time for integration projects and new application development, but it can also make your business information more widely available within your company and greatly reduce the time to integrate applications in the future.

Designed to meet the needs of the mid-market, WebSphere MQ Express is easy to install and simple to use. You can benefit from its powerful application connectivity capabilities without specialized training or knowledge.

WebSphere MQ Express includes easy-to-use features.

- **Rapid and simple installation**
  
  Installed with a few clicks within ten minutes, WebSphere MQ Express can be up and working in just two hours. A wizard guides you through installation and initial setup so you will have a WebSphere MQ Express configuration that is ready to work within a very short time after installation.

- **Out-of-the-box productivity**
  
  The product includes a tour which introduces the basic concepts of application connectivity. Coupled with structured tutorials and interactive help, you will be guided through application integration tasks.

  A file transfer program is provided to show just how easy WebSphere MQ Express is to install and use and demonstrates some of the important built-in capabilities, such as security and assured delivery. You can move files easily and safely between systems and prove some of the basic values of WebSphere MQ Express.

  A postcard application helps you verify the smooth installation of WebSphere MQ Express. This allows you to send and receive some simple test messages, so you can be confident in minutes that your installation has been successful and that you are ready to work with real business data.

  When your product is up and running, you need information to move on and do more and more tasks. With WebSphere MQ Express, you can look at the product's integrated Information Center at the touch of a button. There is also a wealth of printed and electronic information available from the Web and
through other sources. More formal education is available from IBM Learning Services and other vendors.

Start realizing the benefits today, get faster access to your data and become more responsive to customer needs. Be confident that the information you share with your customers and partners is accurate, timely, and safe because of WebSphere MQ reliability.

### 2.4 Reviewing your design

The following good design principles help you to do integration and to build a good service-oriented architecture. Some principles to keep in mind include:

#### 2.4.1 Encapsulation

All components within a system interact with the system only through a clearly defined interface, either by design in the case of new components or by wrapping in the case of legacy components. Use a component only through its interface. The component interacts with the system through that interface. In short, the system should not be able to see past the interface and through the encapsulation into how the component works. Wherever possible, choose the size of a component so the functionality exposed through its interface is simple and coherent. Strongly different pieces of functionality ideally need to be in different components, although clearly this may not be possible when encapsulating an entire legacy application as a single component.

#### 2.4.2 Isolation

Different components in a system must be isolated from each other as far as possible. This means that, in general, looser, more indirect coupling is preferred over tighter, more direct coupling so that the effect of change in a system is restricted as far as possible to the site of that change. Isolation minimizes interdependency and also leads, generally, to systems which are easier to analyze and understand. In a world of change, it must be possible to carry out such analysis quickly and effectively. A variety of techniques can be used to achieve isolation. One example of isolation is communication through an event-driven model as opposed to through a direct call-level interface. The event-driven model avoids one component making a call directly on another, thereby providing each component with a degree of isolation from changes in the other.
2.4.3 Abstraction

Express functionality and structure as simply and as generally as possible. Without accepting this basic axiom, it is hard to manage the appeal of short-term unstructured approaches which almost invariably lead to long-term problems. An example of some abstractions are Generic Business Objects and generic business processes.

2.4.4 Reuse

Reuse all assets wherever practical. Construct all new assets with reuse in mind. In this sense, it is important to make assets easily understandable and reliable, and therefore simpler and cleaner, in order to enhance their reusability. It is also necessary to be able to find assets in order to reuse them.

2.4.5 Agility

Change is inevitable, whether in the purpose for which a system was intended, within the environment in which it is deployed or within the components from which it is constructed.

Anticipate change in all aspects of policy, strategy, architecture, design, implementation, deployment and configuration. Inflexible and unadaptable systems become legacies which constrain the possibilities for change. In this way, even systems which are faithful implementations of accurate situation analyzes become legacies because the analysis from which they are born does not look to the future. Engineering for agility involves identifying where choices with unknown outcomes may lie, rather than simply trying to guess the outcome of future choices based only on current information.

2.4.6 Maintainability

Engineer systems for maintainability. In many ways, maintainability is promoted by each of the other basic principles. Maintainability involves the anticipation of failure and error on the part of all elements involved in creating, using and maintaining a system. In particular, it involves playing to traditional engineering concerns such as traceability, observability and controllability, making systems testable as well as maintainable. While it is tempting to look at systems optimistically, the reality is that errors are the norm, particularly so in distributed systems. Even if the quality of all components in a system can be assured, the actual behavior of these components is typically dependent on other factors which are often external to the system. Whether these are hardware failures, communication failures or infrastructure failures, systems can not usefully be considered in isolation of the environment which supports their operation. All
components must therefore anticipate errors in their operation, whatever the origin of those errors. Maintainability is strongly related to issues of change management.

### 2.4.7 Design by contract

Well-designed software components reveal their functionality, or the service they provide, through a clearly defined interface. The definition of this interface represents a contract between anything using the service and the component which provides that service. Design by contract emphasizes the primacy of this contract. All components and the interfaces by which they are to be used need to be included in a contract drawn up at design time describing the component exactly. During design, strictly adhere to that contract, neither adding to, modifying nor opting out of any of it. Design by contract is a critical link between the methodology of analysis and that of construction. It is also a key aspect of successful distributed development activity, whether distributed geographically or distributed over time. Design by contract allows a complex task to be safely and reliably decomposed successively into a hierarchy of independent sub-tasks. Such a decomposition is what makes it possible for different parts of a task to be handled in different places or at different times. It also enables the whole to function as intended once assembly from its constituent pieces is complete.

### 2.4.8 Unique and explicit expression

Data and functionality need only to be created once and then shared by all who need to use them. If there is something to say, say it once and say it clearly. For example, code business rules clearly in one place rather than everywhere they are used. Express them explicitly and not simply implicitly in the way code operates.

Unique expression does not necessarily mean that only one copy of a particular piece of data or code exists in a system. Instead, maintain such copies as are necessary automatically by some mechanism such as a cache which guarantees their consistency and coherency. In this sense data and code are being logically expressed exactly once, even though they might be physically replicated as many times as necessary. Consistency and coherency are major issues in engineering typical software systems. The principle of unique and explicit expression goes a long way towards making change management effective and reliable. The principle also helps to divide certain classes of complex problems so as to increase opportunities for reuse. Consider the case of a parser, for example: coding the parsing algorithm separately from the grammar (the data which controls the parser's operation) allows both the parser to be reused verbatim with a different grammar, and the grammar to be reused verbatim with a
different parser. Coding the two together in a single entity without this separation compromises the opportunities for reuse of both assets.

2.4.9 Functionality as a service

Meaningful functionality contributing to a larger whole should not be viewed as part of that whole. Instead, look at it as a separate service which is subscribed to or otherwise used by that whole. In this way, functionality which is meaningful in the context of one task can be reused unchanged in the context of any other. It also provides a strategy for decomposing a complex whole into pieces which can be tackled independently, whether or not the pieces will ever be reused.

Presenting complex functionality as a collection of services promotes agility by decomposing a solution into pieces which are easy to reassemble in a different way. This avoids creating pieces which are reasonable to assemble in only one way.

2.4.10 Minimum diversity

While there is usually more than one way to perform a task, there is great virtue in performing it, as far as is reasonable, in the same way that other tasks have been performed in the past. This promotes reuse with understanding and familiarity while avoiding unmanageable proliferation of different standards. In effect, minimum diversity means converging on a standard architecture for as many solutions as feasible. Architecture developed this way will be general enough to express the natural variation between the solutions but specific enough to avoid differences which arise for the sake of differences.
WebSphere Business Integration Server Express

This chapter provides an overview of WebSphere Business Integration Server Express, including the architecture and key concepts. The chapter is divided into the following sections:

- 3.1, “Key concepts” on page 44.
- 3.2, “Integration solution” on page 47.
- 3.4, “Summary” on page 57.
3.1 Key concepts

WebSphere Business Integration Server Express is based on the same technology as WebSphere Business Integration Server. WebSphere Business Integration Server is IBM’s enterprise business integration solution for process integration, workforce management, and enterprise application connectivity. WebSphere Business Integration Server includes a number of tools and components:

- **WebSphere Business Integration Toolset**
  Tools used to manage, design, test and deploy business process solutions.

- **WebSphere Interchange Server**
  Coordinates business process activities across multiple applications.

- **WebSphere MQ Workflow**
  Coordinates long-lived activities that span multiple systems and workforce groups. Automates and manages task list assignments and workflows.

- **WebSphere Business Integration Message Broker**
  Transforms and enriches information between applications that use different message structures and formats.

- **WebSphere Business Integration Adapters**
  Connect a wide range of packaged applications, technologies and protocols to a central hub, allowing easy creation of integrated processes that exchange information between ERP, HR, CRM and supply chain systems.

WebSphere Business Integration Server Express is an offering designed to provide a comprehensive integration solution for small to medium business (SMB) customers. The key technology in WebSphere Business Integration Server Express is based on that in WebSphere Interchange Server. WebSphere Business Integration Server Express comes in two offerings:

1. **WebSphere Business Integration Server Express**
   This is the base product offering, which is a package including Interchange Server Express, Toolset Express, WebSphere MQ, IBM DB2® UDB Express, WebSphere Application Server - Express and a choice of Adapters and Foundation Collaborations.

2. **The WebSphere Business Integration Server Express Plus** includes all the same offerings as WebSphere Business Integration Server Express with the addition of the WebSphere Business Integration Workbench. Toolset Express also includes the Process Designer Express. Optional packages for WebSphere Business Integration Server Express Plus include an Adapter...
capacity pack and a Collaboration capacity pack that enable more scalability for a SMB.

We will examine the solution components in more detail later in this chapter, but both offerings provide solutions that are ready to run out-of-the-box by the solution provider. However, the WebSphere Business Integration Server Express base product is partner-focused, whereas WebSphere Business Integration Server Express Plus with the additional capacity packs is focused on customer and partner needs. This is shown in Figure 3-1 on page 46. You can see that, as more adapters and capacity packs are added on to the base solution, more capabilities become available to support the needs of a medium-sized business.

WebSphere Business Integration Server Express utilizes WebSphere Business Integration adapters to connect back-end and business applications. In order to provide a solution, WebSphere Business Integration Server Express coordinates business activities by connecting to multiple applications to send or receive business data. The business data is coordinated by using information flows to transform, distribute, and enrich the data messages. Visual tooling enables creating, testing, deploying and managing of the industry solution. Development is accelerated by using existing business logic provided by WebSphere Business Integration collaborations.
WebSphere Business Integration Server Express does not include WebSphere Business Integration Message Broker or WebSphere MQ Workflow. The practical differences between WebSphere Business Integration Server Express and WebSphere Business Integration Server are that Express:

- Is restricted to run only on a single server.
- Uses Interchange Server Express to power message brokering.
- Is a simple workflow with support for human interfaces without staff, organizations or work list support.
- Provides standard performance and availability with no clustering support.
- Supports only IBM DB2, IBM DB2 UDB Express, and MS SQL Server for the server repository.
- Does not support external manageability.
- Does not include benchmarking or performance analysis support.
- Is restricted to a maximum of two processors.
3.2 Integration solution

WebSphere Business Integration Server Express offers a process integration solution that can be developed in a five-step process. These steps integrate back-end applications by facilitating automation of integration with partners as well as with front-end applications such as e-business applications. WebSphere Business Integration Server Express accomplishes this in a non-intrusive manner to leverage the existing infrastructure using the set of pre-packaged content adapters and process templates. Figure 3-2 shows the steps to successfully create an integration solution using a five-step process.

The integration steps are:

1. Analyze the current business processes and determine where there are problems.

2. Map and customize solution components using the Process Designer, Business Object Designer and the Map Designer. This allows you to integrate business processes, Business Objects, and transactional data flows across applications to create more efficient interactions of resources, and to increase the manageability of resources. This integration step uses the Toolset Express to integrate internal systems, customers and business partners.

3. Deploy the process integration solution to a runtime environment using adapters and maps to enable communication between applications.
4. Monitor and manage the processes using probes for local management and a Web-based dashboard that allows anytime, anywhere administration. Probes are user-defined “save-points” in the business process that provide input to business performance measures.

5. Simulate and validate the process using the WebSphere Business Integration Workbench Modeler Entry Edition (available with WebSphere Business Integration Server Express Plus). Determine the process outcomes and feed this information back to the beginning of the lifecycle for continued improvement of the solution.

3.3 Architecture

Figure 3-3 on page 49 depicts the end-to-end solution provided by WebSphere Business Integration Server Express from an architectural standpoint. WebSphere Business Integration Server Express acts as the middleware between applications and reduces manual procedures by automating the update, creation, and deletion of the information that is transferred between various company applications or partner applications.

For example, if two applications need to communicate with each other using the server, then the Toolset Express is used to create the necessary Business Objects (BOs), maps, collaborations, and connectors to enable this communication. The completed communication uses an Application Specific Business Object (ASBO) that is passed to the adapter and a map is used to convert the ASBO to a Generic Business Object (GBO). What is done with the GBO is determined by a collaboration that is triggered by the combination of the GBO and a subscribed verb (for example, create, update, delete). This determines what process will be completed. Once the process is determined, the GBO is delivered to the proper application and mapped to that application’s ASBO. Then the application uses the verb on the ASBO to determine what will be done with the ASBO, whether it is a create, update, delete, or other business functionality. A process can use various application and technology adapters and can be triggered by various means such as from an e-mail sent to an external/internal entity, or by an application event such as creating an employee in a payroll system. The type of connector used influences the type of process completed. For instance, the e-mail adapter handles e-mails sent, whereas a JMS adapter may be used to send a long running message to a business partner and require a response.
3.3.1 Supported technologies

This section details the products and technologies that are part of the WebSphere Business Integration Server Express solution.

**WebSphere Business Integration Server Express software**

The WebSphere Business Integration Server Express software package includes:

- IBM WebSphere MQ
- IBM WebSphere Application Server - Express
- IBM DB2 UDB Express
- WebSphere Business Integration Workbench Modeler Entry Edition (only included in WebSphere Business Integration Server Express Plus)
- IBM Java Developer Kit
Database management systems supported
IBM DB2 UDB Express is shipped with the WebSphere Business Integration Server Express package. MS SQL Server 2000 is also supported.

The database management system is used for the WebSphere Business Integration Server Express repository database, which in turn is used to store repository data, work in progress events, configuration data, transactional data, and relationship data (used for cross-referencing Business Objects).

IBM Java Developer Kit
The JDK is prerequisite software for the installation of WebSphere Business Integration Server Express. The JDK contains the Java interpreter, the compiler (javac), the debugger (jdb), and the AppletViewer. There are two principal products in the J2SE platform family: Java 2 Runtime Environment, Standard Edition (JRE) and Java 2 Software Development Kit, Standard Edition (SDK). The JRE provides the Java APIs, Java virtual machine, and other components necessary to run applets and applications written in the Java programming language. It is also the foundation for the technologies in the Java 2 Platform, Enterprise Edition (J2EE) for enterprise software development and deployment.

WebSphere MQ
WebSphere MQ is prerequisite software that provides the service-oriented infrastructure in a common runtime environment. WebSphere MQ accomplishes this by enabling persistent data messaging to deliver events to the Business Integration Server during the subscription phase. Using individual queues for each adapter, each queue moves data between the adapter agent and the adapter controller. WebSphere MQ provides a means for business applications to communicate, as well as means for an administrator to manage data in a controlled manner. WebSphere MQ enables the user to graphically define business processes and their activities to the level of detail needed for automation. Buildtime includes graphical support for declaring and documenting:

- Business rules on process navigation between steps.
- Business rules for role-based work assignment.
- Need-to-know constraints for process interactions, with data flows and data mapping.
- Process interface definitions (data, programs, queues).

WebSphere Business Integration Server Express automatically installs the Object Request Broker and the Interchange Server Express. The Interchange Server Express supports integration processes which include:

- Business processes
- Event sequencing
- Work in progress management
- Transaction management
- Mapping and cross-referencing
- Support for adapter connectivity
- Tools for configuration of components necessary to process integration data

**Object Request Broker**
Object Request Broker uses the CORBA product to transport Business Objects to the adapter controller. CORBA stands for Common Object Request Broker Architecture. Object Request Broker performs two tasks, administrative communication and BO transportation. The administrative communication tools configure information from the Business Integration Server to clients and adapter agents. BO transportation controls the BOs transportation between the adapter agent and adapter controller.

### 3.3.2 Collaborations

Please refer to Chapter 10, “Collaborations” on page 309 for more details about this topic. Collaborations are application integration modules that are deployed to Interchange Server Express and contain logic that describes the business processes. A few of the tasks collaborations assist with include:

- Validation by checking and retrieving related BOs.
- Synchronization by updating target systems with BOs.
- Filtering by including or excluding BOs based on business data.
- Data enrichment through retrieving and mapping related BOs.
- State management by managing transactions, reconciling related transactions and processes as well as compensating transactions.
- Auditing by capturing data to the repository database.
- Error handling by automating fault processing to maintain data integrity.
- E-mail or other notification to the administrator when there is a failure.

Collaboration templates are packaged as part of the product. These templates can be altered to fit the needs of a particular business process. Thus the sample templates may be reused and saved using different collaboration names that use the same logic, but to perform different processes. Collaboration templates are used to create collaboration objects that show graphically how a collaboration is completed and the components that are used to complete a process, as shown in Figure 3-4 on page 52.
Figure 3-4 illustrates a collaboration design that shows how customer information could be synchronized in a business. Depending on the process to be completed, the starting point would be either create, retrieve, or update. For example, if a customer calls in to request a service, the employee would first check for the existence of the customer in the system, thus finding out whether to create or retrieve data about the customer. If the customer exists, the employee would then update the customer’s information with the information about the current call.

3.3.3 Adapters

Adapters process data in both directions and handle many types of transactions. An adapter is used to invoke multiple operations (create, update, delete) for Business Objects. Since adapters are driven by the meta-data in Business Objects, no code is required to create the adapter. Adapter Agents are responsible for the communication with applications. Adapter controllers reside on the Interchange Server Express and manage communication with the Adapter Agent. Adapters are provided, out of the box, to work with packaged applications and they retrieve their initial configuration and Business Object definitions from the repository database via the CORBA service. Please refer to Chapter 7, “Adapters” on page 159 for more details concerning adapters.
Types of adapters

IBM WebSphere Business Integration Adapters allow users to quickly and easily create integrated processes that exchange information between: ERP, HR, CRM and supply chain systems. Application Adapters extract data and transaction information from cross-industry and industry-specific packaged applications and connect them to a central hub. Mainframe adapters leverage best-of-breed technology to access applications, programs and data residing on z/OS® and OS/390® systems. Technology Adapters provide connectivity to data, technologies and protocols that enhance integration infrastructure. Adapter Development Tools is an integrated toolkit that provides a framework for development of custom adapters.

Figure 3-5  Web-based monitoring example

If WebSphere Application Server is installed, it may be used for the Web-based monitor. Adapters enable Web services as part of the process flows using integrated tooling, making it easy to expose processes as Web services for integrating with other applications facilitating reuse of processes. Allows you to import business process execution language (BPEL) documents into your processes and export your processes as BPEL documents. This process of integrating a Web application and integrating the processes with a legacy system is demonstrated in Figure 3-5.

3.3.4 Business Objects

Business Objects (BOs) come in two forms: Application Specific Business Object (ASBO) and Generic Business Object (GBO). These two forms allow data to be transported between applications, ensuring data is usable by different business processes and departments within a business. By mapping BOs from application-specific to generic layouts WebSphere Business Integration Server Express provides isolation from participating applications and enables process logic integration. Upgrading or adding participating applications does not affect
the entire integration solution, since it is only necessary to define new ASBOs and map between the ASBO and the GBO. Figure 3-6 on page 55 demonstrates how an ASBO is:

1. Passed from one application to an Adapter Agent.
2. Passed to the adapter controller.
3. Mapped to a GBO.
4. Sent to a collaboration which uses the trigger to determine:
   a. Where the GBO should be sent.
   b. What process should be completed.

See Chapter 8, “Business Objects” on page 209 for more details concerning Business Objects.

Business Objects have the following properties:

- Data type
- Default values
- Attributes
- Verbs/actions
- Hierarchical structure
- Repeating structure

- Application Specific Business Objects

For details about how application-specific information can be retrieved using Object Discovery Agents (ODA) to aid in the creation of Business Object instances, see 8.1, “Introduction to Business Objects” on page 210.

- Generic Business Objects

GBOs are data models that are supported by the WebSphere Business Integration Server Express infrastructure and tooling; these objects are associated with specific integration processes. Refer to Chapter 8, “Business Objects” on page 209 for more details.
3.3.5 Data map services

In Figure 3-6, which shows where maps exist within an integration solution, we can see that the map is the key to coding and decoding BOs so that they can be used throughout multiple business applications. GBOs are mapped to ASBOs. Business Objects can be mapped in a number of different ways and the mapping tools support features such as:

- Multi-mode mapping using cardinalities, 1-1, 1-M, M-1, M-M
- Polymorphic mapping (content based),
- Complex structures
- Callable from multiple contexts
- Standalone testing

3.3.6 Toolset Express

IBM WebSphere Business Integration Toolset is a set of easy-to-use tools providing administrative and development support for system management, application connectivity and business process modeling. The Toolset facilitates managing the lifecycle of business processes including rules-driven transformation, relationship mapping and data transformation. Toolset users can use business probes to monitor process flows, use a Web-based dashboard with customizable views to monitor business processes and business data, and use an integrated graphical debugger to test end-to-end integration.
Administration tools
The Administration tools provided by Toolset Express include:

- Flow manager
- Log Viewer
- Relationship Manager
- System Manager
- System Monitor

Flow manager is used for graphically querying unresolved flows. The Log Viewer traces information for the business processes as they travel through processing points. The Relationship Manager is a tool that enables the administrator to monitor and manage the relationships between Business Objects to synchronize data across many applications. The System Manager provides an interface to monitor, analyze and control the entire integration solution. The System Monitor is a central graphical and browser-based tool to monitor and control the key components within the server environment. Refer to Chapter 11, “Administration tools” on page 379 for more details about how to use the development tools provided by Toolset Express.

Design and development tools
The design and development tools provided by Toolset Express include:

- Activity Editor
- Business Object Designer
- Collaboration Debugger
- Process Designer
- Relationship Designer
- Test Environment

The Activity Editor enables flow modeling using simple drag and drop creation of logic flows. The Business Object Designer is a graphical tool used to generate and maintain BOs. The Collaboration Debugger establishes break points and can step through collaboration processes to examine Business Objects and event data. The Map Designer transforms ASBOs to and from the GBO format. A Process Designer is a tool for users to sketch and refine the logical flow of business processes graphically. The Relationship Designer allows a user to define the relationships between application objects that are necessary for synchronizing data across multiple applications. A Test Environment supports both stand-alone unit testing and integrated testing environments. Refer to Chapter 6, “Introduction to Toolset Express” on page 99 for more details about how to use the development tools provided by Toolset Express.
3.4 Summary

WebSphere Business Integration Server Express contains Interchange Server Express, Toolset Express, WebSphere MQ, IBM DB2 UDB Express, WebSphere Application Server - Express, and a choice of adapters and foundation collaborations. WebSphere Business Integration Server Express is used for business process integration solutions and addresses these in a non-intrusive manner by leveraging the existing infrastructure and using the set of pre-packaged adapters and process templates. This is done by coordinating business activities and connecting multiple applications to send/receive business data using adapters and collaborations. The business data is coordinated by using information flows to transform, distribute, and enrich the data messages. Visual tooling enables creating, testing, deploying and managing of the industry solutions and development is accelerated by using WebSphere Business Integration collaborations. WebSphere Business Integration Server Express integrates back-end applications and facilitates automation with partners as well as with in-house front-end applications such as e-business applications.
Chapter 4. Installation and configuration

This chapter provides the installation and configuration details of WebSphere Business Integration Server Express and the prerequisite software. The chapter is divided into the following sections:

- 4.1, “System requirements” on page 60.
- 4.2, “Installation from the Launchpad” on page 63.
- 4.4, “Installing IBM DB2 Universal Database” on page 71.
- 4.5, “Installing WebSphere Business Integration Server Express” on page 73.
4.1 System requirements

We performed our redbook testing on a Windows 2000 test system. The software and hardware requirements for installing WebSphere Business Integration Server Express on Windows 2000 are described below. WebSphere Business Integration Server Express is also supported on OS/400 and Linux, so we have additionally detailed the requirements for those platforms, but we did not test on these systems when writing this redbook.

WebSphere Business Integration Toolset Express may be installed on Windows XP, so we also detail the requirements for this option.

4.1.1 Requirements for Microsoft Windows 2000

This section provides the software and hardware requirements for the installation of WebSphere Business Integration Server Express on Windows 2000.

Software requirements
The following software is required for installation using Microsoft Windows 2000:

- Windows 2000 Professional Server or Advanced Server with Service Pack 3
- One of the following:
  - DB2® Universal Database Express Edition V8.1 (included as part of IBM WebSphere® Business Integration Server Express V4.3)
  - Microsoft SQL Server 2000
- One of the following:
  - IBM WebSphere Application Server - Express, V5.02 or later (included)
  - Tomcat, V4.1.24 or V4.1.27
- One of the following:
  - Microsoft Internet Explorer 6 with Service pack 1
  - Netscape Navigator 4.74 or later
- A C or C++ compiler that supports DB2 to compile DB2 stored procedures
- Adobe Acrobat 4.0.5 or later
- SMTP-based e-mail system

Hardware requirements
The following hardware is required when performing the installation using Microsoft Windows 2000:

- 1GHz processor or dual processors, each with a minimum speed of 667MHz
4.1.2 Windows XP

This section provides the software and hardware requirements for the installation of WebSphere Business Integration Server Express on Windows XP.

Software requirements
Installation with Microsoft Windows XP requires the following software:

- One of the following operating systems:
  - Windows 2000 Professional Server
  - Windows 2000 Professional Advanced Server with Service Pack 3
  - Windows XP Professional with Service Pack 1A

- On of the following:
  - IBM Rational® ClearCase® LT 4.2
  - Concurrent Version System (CVS) 1.11

- One of the following:
  - Microsoft Internet Explorer 6 with Service Pack 1 or later
  - Netscape Navigator 4.75, or later

- Adobe Acrobat 4.05 or later

Hardware requirements
Installation using Microsoft Windows XP requires the following hardware:

- 1GHz processor
- 512MB RAM
- 10GB disk space

4.1.3 Requirements for OS/400

This section provides the software and hardware requirements for the installation of WebSphere Business Integration Server Express on OS/400.

Software requirements
Installation on an OS/400 system requires the following software:

- OS/400, version 5.2 or 5.3 operating system (5722-SS1) which includes:
  - IBM DB2 Universal Database™ for iSeries
  - IBM Developer Kit for Java™ (5722-JV1) which includes:
    - Option 5 (JDK 1.3)
Other software products included as distribution media:

- Crypto Access Provider for IBM @server iSeries (5722-AC3)
- IBM HTTP Server Powered by Apache (5722-DG1)
- TCP/IP Connectivity Utilities for IBM @server iSeries (5722-TC1)
- IBM Toolbox for Java (5722-JC1)

One of the following:

- IBM WebSphere Application Server - Express, V5.1 for iSeries (5722-E51), included
- IBM WebSphere Application Server, 5.0 for IBM @server iSeries (5733-WS5)
- IBM WebSphere Application Server, V5.1 for IBM @server iSeries (5733-W51)

SMTP based e-mail system

One of the following:

- Microsoft Internet Explorer 6 with Service pack 1
- Netscape Navigator 4.74 or later

Adobe Acrobat 4.05 or later

Hardware requirements
The following hardware is required when installing to an AS/400 system:

- IBM AS/400® server with a minimum commercial processing workload of 450
- 1GB RAM
- 40GB disk space

4.1.4 Requirements for Linux
This section provides the software and hardware requirements for the installation of WebSphere Business Integration Server Express on Linux.

Software requirements
The following software is required when installing this product on a Linux-based system:

- One of the following operating systems:
  - Red Hat Enterprise Linux Advanced Server for Intel® 3.0
– SuSE Linux Enterprise Server 8.0
▶ IBM DB2 Universal Database Express Edition, V8.1 (included)
▶ One of the following:
  – IBM WebSphere Application Server - Express, V5.1 (included)
  – IBM WebSphere Application Server - Express, V5.0.2
  – Tomcat, V4.1.24 or 4.1.27
▶ SMTP-based e-mail system
▶ A C or C++ compiler that supports DB2 to compile DB2 stored procedures.
▶ Netscape Navigator, V4.78
▶ Adobe Acrobat 4.05 or later

**Hardware requirements**
The following hardware is required when installing to a Linux system:
▶ 1GHz processor or dual processors, each with a minimum speed of 667MHz
▶ 512MB RAM
▶ 40GB disk space

**4.2 Installation from the Launchpad**
This section provides an overview of the steps necessary to install WebSphere Business Integration Server Express on the Windows platform

**Note:** The installation instructions throughout this document assume installation from product CDs. If you plan to download the program from Passport Advantage®, refer to your Passport Advantage information for those downloading instructions.

Before beginning installation:
1. Remove previous versions of all software and configuration information.
2. Check that your system meets the hardware requirements listed in the “System requirements” on page 60.
3. Check the following Web site for any available Fix Packs for your product:
   http://www.ibm.com/software/integration/websphere/support/
4. Ensure that you have Windows administrator privileges and use a user ID of less than 20 characters
If you have Norton AntiVirus running on your machine, turn it off and restart your machine. From the desktop, do the following:

2. Right-click Norton AntiVirus Client.
3. Select Stop.

**Note:** If these requirements are not met, you will see error messages describing the problems and the Launchpad program will terminate.

We use the Launchpad to install WebSphere Business Integration Server Express as this ensures a smooth and silent install.

1. To start the Launchpad, insert the *WebSphere Business Integration Server Express or Express Plus CD* in your computer. You see the Launchpad Welcome screen. To navigate the Welcome screen, use the buttons on the left side. When you are finished and want to exit the Launchpad, click the Exit button.

   Figure 4-1 shows the Launchpad Welcome window for the WebSphere Business Integration Server Express Plus product.
To check which software is already installed on your system, click **Software Prerequisites**. The Launchpad checks for the following programs:

- IBM Java(TM) Development Kit 1.3.1_05
- IBM WebSphere MQ 5.3.0.2 CSD5 and higher CSD levels
- IBM WebSphere Application Server - Express v5.1
- Microsoft Internet Explorer 6 Service Pack 1 or later
- Netscape Navigator 4.7x

The installation status of each is displayed on the Software Prerequisites screen. The status values are Not Installed, Optional, or OK.

### 4.3 Installing prerequisite software

This section describes how to use the Launchpad to directly install missing prerequisite software.
4.3.1 IBM Java Developer Kit

**Note:** If you do not plan to perform collaboration and mapping development, you do not need this prerequisite. To skip this install, you can select **I do not plan to compile collaborations or maps** on the launchpad and the prerequisite status changes to OK.

![Software Prerequisites](image)

**Figure 4-2** Software Prerequisite listing with the IBM Java Developer Kit installed.

Figure 4-2 shows that the IBM Java Developer Kit has been installed. If the IBM Java Developer Kit is not installed, do the following:

1. From the Launchpad Software Prerequisites screen, click the arrow to the left of **IBM Java Development Kit 1.3.1_05** to expand the entry.

2. Click the **Install** button to start the silent installation of IBM Java Development Kit 1.3.1_05.

3. After IBM Java Development Kit 1.3.1_05 silent installation, verify that its status in the Launchpad changed from Not Installed to OK.
To install IBM WebSphere MQ, do the following:

1. Figure 4-3 displays a menu that includes the Install button. Click **Install**.

2. Do one of the following:
   - If you want to install the Interchange Server Express component, ensure the box beside the field labeled I intend to only install the Tools component or an Adapter is *not* selected. The Launchpad automatically selects the option Install IBM WebSphere MQ 5.3.0.2 CSD5 Server and Client (101 MB).
   - If you want to install only the Toolset Express or an adapter, ensure that the box beside the field labeled I intend to only install the Tools component or an Adapter *is* selected. The Launchpad automatically selects the option Install IBM WebSphere MQ 5.3.0.2 CSD5 Client Only (16 MB).
3. Select **Install** to start the silent installation of **IBM WebSphere MQ 5.3.0.2 CSD5**.

4. After the IBM WebSphere MQ 5.3.0.2 CSD5 silent installation completes, verify that its status in the Launchpad changed from **Not Installed** to **OK**.

5. If the Launchpad finds a pre-existing installation of WebSphere MQ 5.3.0.2 without CSD5 applied, it offers to automatically patch the software. In this case, expand the selection IBM WebSphere MQ 5.3.0.2 CSD5 on the Launchpad Software Prerequisites screen and select **Apply CSD05**. Silent installation of the patch executes and the software status in the Launchpad changes to **OK**.

**Important:** After you install WebSphere Business Integration Server Express Plus, you will need to add a listener to the WebSphere MQ service. Instructions are provided in “Adding a listener to the WebSphere MQ service” on page 68.

---

**Adding a listener to the WebSphere MQ service**

Installing WebSphere Business Integration Server Express or Express Plus created a queue manager; you must add the WebSphere MQ listener as a Windows service and configure it for automatic startup as follows:

1. Select **Start → Programs → IBM WebSphere MQ → WebSphere MQ Explorer** to launch the WebSphere MQ Explorer.

2. In the left pane of the WebSphere MQ Explorer:
   
   a. Select the **Queue Managers** folder and then the name of your queue manager. The Queue Manager Status in the right pane should show the queue manager status as **Running**.
   
   a. If it is not **Running**, right-click the queue manager name, then select **Start** from the drop-down list. The Queue Manager Status should change to **Running**.

3. Open WebSphere MQ Services by right-clicking the queue manager name in the right pane, then selecting **All Tasks → Services** from the drop-down list.

4. In the Console Root tree of the MQServices window, expand WebSphere MQ Services, then select the queue manager. There should be three entries for your particular queue listed in the right pane: Queue Manager, Command Server, and Channel Initiator. (If you do not see a Channel Initiator, it will appear after you restart your system.)

5. For the three entries in the right pane: Queue Manager, Command Server, Channel Initiator, set the Startup mode to **Automatic** by right-clicking each, then selecting **All Tasks → Automatic**.

---

68   WebSphere Business Integration Server Express The Express Route to Business Integration
6. Add a listener entry by right-clicking the queue manager name in the Console Root tree and then selecting New → Listener from the drop-down list.

7. In the Parameters tab of the Create Listener Service dialog box, select TCP as the Protocol and 1414 as the port number, then select OK. You must use a unique port number for each listener. Assign the port a number other than the default of 1414, then select OK.

8. Configure the listener for automatic startup by right-clicking it and then selecting All Tasks → Automatic.

9. Close IBM WebSphere MQ Explorer. Select Yes when prompted to save your changes.

10. Restart your machine.

11. Verify that Channel Initiator has been started by following these steps:
   a. Select Start → Programs → IBM WebSphere MQ → WebSphere MQ Services. Alternatively, right-click the WebSphere MQ icon on your system tray, then select WebSphere MQ Services.
   b. In the MQServices window, verify that Channel Initiator is running and that it is set for Automatic.
   c. Close the MQServices window.

4.3.3 Installing WebSphere Application Server - Express

This prerequisite is mandatory if you intend to install the Toolset Express components System Monitor and Failed Event Manager both of which require an application server with a servlet engine. If you have already installed WebSphere Application Server versions 5.0.2 or 5.1, or Tomcat versions 4.1.24 or 4.1.27, this prerequisite is satisfied.

To install IBM WebSphere Application Server - Express, do the following:
1. From the Launchpad Software Prerequisites screen, expand IBM WebSphere Application Server - Express v5.1.
2. Select Install to start the silent installation.
3. After silent installation completes, verify that its status in the Launchpad changed from Optional to OK.

4.3.4 Installing Microsoft Internet Explorer

This prerequisite is required if you want to install the Toolset Express components System Monitor and Failed Event Manager, both of which require a Web browser. This guide does not provide installation instructions for Microsoft Internet Explorer 6 Service Pack 1. For a description of the installation process,
on the Launchpad Software Prerequisites screen, expand the entry Microsoft Internet Explorer 6 Service Pack 1 or later and select the link to the Microsoft Web site.

4.3.5 Installing Netscape Navigator

This prerequisite is required if you intend to install the Toolset Express components System Monitor and Failed Event Manager, both of which require a Web browser. This guide does not provide installation instructions for Netscape Navigator. For a description of the installation process, on the Launchpad Software Prerequisites screen, expand the entry Netscape Navigator 4.7x and select the link to the Netscape Web site.

4.3.6 Completing the installation of prerequisite software

![Launchpad displaying fulfillment of the Software Prerequisites.](image)

After completing the installation of the prerequisite software, the Launchpad looks exactly like Figure 4-4. All the prerequisite software should be installed according to the way the software is listed in the Launchpad.
4.4 Installing IBM DB2 Universal Database

WebSphere Business Integration Server Express requires a database to store repository information used by Interchange Server Express. This section describes how to install and configure IBM DB2 Universal Database V8.1 Express for use with WebSphere Business Integration Server Express

IBM WebSphere Business Integration Server Express and Express Plus support the following databases:

- IBM DB2 Universal Database v8.1 Express
- IBM DB2 Universal Database v8.1 Enterprise
- Microsoft SQL Server 2000

4.4.1 Checking database installation prerequisites

Before checking for installation of or actually installing a database, do the following:

- Ensure that you have administrator privileges to create a new database and add new users.
- Check that you have the minimum requirements for databases, including:
  - WebSphere Business Integration Server Express or Express Plus administrator user account with database and table creation privileges created.
  - Fifty MB of disk space for data files available for the Interchange Server Express repository database.
  - The maxappls and maxagents parameters configured with a minimum of 50 user connections each.
  - The table space for the mapping tables, optional, configured to contain at least 50 MB of data.
  - Maximum application heap size configured to be at least 2048.

**Note:** If you want to use only the Toolset Express component, a database server is not required.

The installation instructions in this section assume that this is the first time you are installing DB2 UDB Express on your machine. If you have previously installed DB2 UDB Express through the Launchpad, and un-installed it following standard DB2 procedures, you must first do the following before you use the Launchpad to reinstall DB2 UDB Express:
1. Manually delete two residual user IDs that the Launchpad automatically created when it performed the first DB2 UDB Express installation. To remove these IDs, do the following:
   a. Select Start → Settings → Control Panel → Users and Passwords.
   b. On the User and Passwords screen, under the User Name column, select the db2admin user ID and the smbadmin user ID.
   c. Click Remove.
   d. Click Apply and OK to exit the User and Passwords screen.

2. Manually delete copies of the files serverexp, OptionFile_DB2.txt, or OptionFile_MSSQL2000.txt that might exist in the directory %TEMP%.

4.4.2 Installing DB2 UDB Express

To install DB2 UDB Express, from the Launchpad Database Setup screen, do the following:

1. Select Install IBM DB2 Universal Database v8.1 Express. The Launchpad will check for an installed database.
   a. If the database is not installed, launch the installation of IBM DB2 Universal Database v8.1 Express.
   b. If IBM DB2 Universal Database v8.1 Express or Enterprise is installed, launch configuration of the existing installation.

   A dialog will notify you when installation and configuration are complete.

After IBM DB2 Universal Database v8.1 is successfully configured, you will see a message that the database configuration is complete. The configuration process does the following:

- Configures a C compiler for DB2.
- Creates a database named SMB_DB.
- Creates a user named db2admin, with a password of smbP4$$word.
- Grants the appropriate authorities to the db2admin user in the SMB_DB table.

There are also three tables that load automatically upon installation. WebSphere Business Integration Server Express relies on persistent data store to keep information such as the data held in these tables. These tables include:

- CWRRepository - holds definitions and configuration data
  - CxReposAttributes, CxReposBusObjSpecs, CxReposBOSpecAttrs, etc.
- CWevents - holds work-in-progress information
  - CxPBusObjMessage, CxPBusObjState, CxWipBlobs, CxWIPObjects
4.5 Installing WebSphere Business Integration Server Express

The installation instructions in this section assume the following:

- WebSphere Business Integration Server Express or Express Plus is not already installed on your machine.
- You are installing software from a CD. If you plan to use the downloaded version from Passport Advantage, refer to your Passport Advantage information for those downloading instructions.
- The machine you are using for the install is running the Microsoft Windows 2000 operating system. Only the Toolset Express component is supported on the Windows XP operating system.
You are installing a WebSphere Business Integration Server Express system. If you are installing a WebSphere Business Integration Server Express Plus system, you might see slightly different windows.

From the Launchpad, do the following:

1. Click the button labeled **Install Product** from the Launchpad. The Install Product window appears.

2. Select **Launch Product Installation**. The Welcome screen appears.

3. From the Welcome window, select **Next**. The Software License Agreement screen appears. Read and accept the terms of the Software License Agreement by clicking the radio button beside the entry *I accept the terms in the license agreement*, then select **Next**. You see the Destination screen.

4. On the Destination screen, accept the default installation location of C:\IBM\WebSphereServer or browse for a different location, then select **Next**.

5. On the Feature screen, select those components you want to install and select **Next**.

6. On the Pre-installation Summary screen, review the features and installation location and select **Next**.

7. The installer verifies that enough disk space exists for the installation:
   - If enough space does not exist, the **Next** button is disabled because installation cannot be completed with the provided disk space. There are several things you can do:
     - Click **Back** and deselect some features.
     - Delete some unneeded space on the specified drive.
     - Change the target location by revisiting the Destination screen.
   - If enough space does exist, installation and configuration begin. You see a number of informational screens during the install. When installation and configuration are complete, you see the Post-installation Summary screen, indicating if the process was successful or if problems were encountered.

8. On the Reboot screen, select whether you want to restart your machine now or later, and select **Finish** to exit the Launchpad.

9. Add a listener to the WebSphere MQ service by following the instructions in the “Adding a listener to the WebSphere MQ service” on page 68.
Getting started

This chapter describes how to get started with WebSphere Business Integration Server Express and illustrates some key product features by building BOs, maps, and adapter samples. This chapter is divided into the following sections:

- 5.1, “Using the System Manager” on page 76.
- 5.2, “Creating a new instance of the JDBC adapter” on page 76.
- 5.3, “Creating a test project” on page 80.
- 5.4, “Creating a new connector” on page 81.
- 5.5, “Creating Business Objects” on page 84.
- 5.6, “Creating maps” on page 87.
- 5.8, “Testing the connector” on page 90.
- 5.9, “Helpful resources” on page 95.

We use the WebSphere Business Integration Server Express InterChange Component Library components and tools to create an example instance that uses a JDBC adapter. We describe the implementation of the test project and build the associated BOs and maps needed to test the JDBC adapter.
5.1 Using the System Manager

The System Manager is the tool used to organize the creation of WebSphere Business Integration Server Express components such as maps, Business Objects, and connectors. The tool provides a graphical view of the components that are part of our integration solution and also allows us to monitor and manage the Interchange Server Express runtime that is the deployment environment for our integration solution. To group a set of related tasks together, the System Manager uses different perspectives. A perspective is a logical grouping of tools and views used to perform related tasks.

1. To choose a perspective, open the System Manager by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → System Manager.

   To open the perspectives that will be used in this tutorial, follow these steps:

2. Click Window → Open Perspective. Several of the most commonly used perspectives are displayed. If you do not see the perspective you want, click Other. This list of perspectives changes depending on the perspectives that you use more frequently.

3. From the Select Perspective dialog box, select System Manager and click OK. This perspective loads automatically. The System Manager view will show a structured outline of the file that is currently open in the editor area. The Navigator view shows your projects, folders, and files in a hierarchical tree in the System Manager perspective. This view will display the User projects and the ICLs.

4. The other perspective used will be the CVS perspective. To open this perspective, repeat steps 1 through 3 and select the CVS Perspective instead.

   This perspective gives you a view of all the components that are in the CVS, which is particularly useful when working on a team and synchronizing maps and BOs with other developers on the team. This tool enables users to share their projects through the Sample overview.

5.2 Creating a new instance of the JDBC adapter

In this section, we create a new instance of the JDBC adapter and configure it to respond to changes in data in the SAL301 application database.

1. To create a new instance of the JDBC connector, begin by copying the connector files. Navigate to C:\IBM\WebSphereServer\connectors\ and select the JDBC folder. Copy and paste this folder in the same location. Rename the copied folder SAL301RConnector.
2. Open the SAL301RConnector folder and rename the following files as shown in Figure 5-1:
   - start_JDBC.bat renamed to start_SAL301R.bat
   - start_JDBC_service.bat renamed to start_SAL301R_service.bat
   - CWJDBC.jar renamed to CWSAL301RConnector.jar

```
[Image of file structure]
Figure 5-1   New SAL301RConnector Instance of the JDBC Connector
```

3. Edit the start_SAL301R.bat file to set up the required JDBC properties for DB2.
   - Set the JDBCDRIVERPATH to the location of the DB2 JDBC driver zip file
     SET JDBCDRIVERPATH=c:\ibm\sqlib\java\db2java.zip
   - Add the DB2 bin directory(c:\ibm\sqlib\bin) to the -Djava.library.path
     -Djava.library.path="%CROSSWORLDS%\bin;%CONNDIR%;%MQ_LIB%;%JRE_EXT_DIRS%;c:\ibm\sqlib\bin %ORB_PROPERTY%

See Figure 5-2 on page 78 for an example.
4. Navigate to the C:\IBM\WebSphereServer\connectors\message directory. Copy JDBCConnector.txt and paste it in the same location. Rename the copy to SAL301RConnector.txt.

5. Navigate to the C:\IBM\WebSphereServer\mqseries\ directory. Copy the Crossworlds_mq.tst file and paste it in the same location. Rename the copy to SAL301R_mq.tst.

6. Open this file in Notepad and edit it so the file appears as shown in Figure 5-3 on page 79.

7. Save the changes to the file.
8. Open a command window and type:

```
    cd C:\IBM\WebSphereServer\mqseries\n```

This changes the drive to where the SAL301R.mq.tst file is to set up the MQ queues.

9. Type:

```
    runmqsc < C:\IBM\WebSphereServer\mqseries\SAL301R.mq.tst
```

Make sure that there is only one space before and after the < symbol. Press Enter. The result reads:

8 MQSC commands read. No commands have a syntax error. All valid MQSS commands were processed.

10. Create and populate the SAL301R database. Open a DB2 command window and change to the directory where you extracted the redbook additional material database files.
**Note:** This will be C:\SAB404R\DDL if you followed the instructions in Appendix A, “Deploying the redbook sample code” on page 603.

Enter:
```
db2 -td@ -vf SAL301R.ddl
```

11. Create the event tables by opening a DB2 command window, change to the directory where you extracted the redbook additional material database files.

Enter:
```
db2 -td@ -vf SAL301R-Events.ddl
```

12. Create a shortcut to start the connector by navigating to **Start → Programs → IBM WebSphere Business Integration Server Express → Adapters → Connectors**.

Right-click JDBC and click **Copy**.

13. Navigate to C:\Documents and Settings\All Users\Start Menu\Programs\IBM WebSphere Business Integration Express\Adapters\Connectors, right-click the folder and select **Paste Shortcut**. Then right-click the JDBC shortcut and select **Properties**.

14. In the Target field, enter:
```
C:\IBM\WebSphereServer\connectors\SAL301RConnector\start_SAL301R.bat
SAL301RConnector WebSphereICS
-cconnectors\SAL301RConnector\SAL301RConnector.cfg
```

15. Change the comment to SAL301R.

16. Rename this shortcut to SAL301R.

### 5.3 Creating a test project

1. Open the System Manager by navigating through **Start → Programs → IBM WebSphere Business Integration Server Express → Toolset → Administrative → System Manager**.

2. The System Manager Perspective opens. In the Interchange Server Component Manager in the lower left, right-click **Interchange Server Instances** and select **Register Server**.

3. In the Server name field, click the **Browse** button and select **WebSphereICS** as the server. The user name is admin. The password is null. Click the radio button to save the user name and password, then click **OK**.

4. A new server instance is listed under Interchange Server Instances. To connect to the server, go to **Start → Programs → WebSphere Business Integration Server Express → Interchange Server Express**. Give the server a minute to start.
5. In the System Manager, right-click the WebSphereICS server and select Connect. Click OK.

6. Create a test ICL by right-clicking Integration Component Libraries and selecting New integration component library.

7. In the Test Project Name field, type TestProjICL and click Finish. Drag and drop maps, BOs, and so on from the WBIExpressLibrary to the TestProjICL. Creating a test ICL keeps the original WBIExpressLibrary that comes with the product free of all test materials.

8. In the System Manager, right-click User Projects and select New ICS Project. Name the new project TESTPROJECT.

9. In the available ICL, click the radio button next to TestProjICL and click Finish.

5.4 Creating a new connector

1. To create a new JDBC connector in the TestProjICL, right-click Connectors and select Create New Connector. There are already some test and default connectors in this library, but we will be using a new instance of the JDBC connector.

2. Select JDBCAdapterTemplate in the Template Name. In the Name field, type SAL301RConnector and click OK. Refer to Figure 5-4 on page 82.
3. In the Standard Properties tab, change:

- Agent Trace Level to 5.
- Controller Trace Level to 5.
- Delivery Transport to JMS.
- JMS.MessageBrokerName to WebSphereICS.queue.manager.
- MessageFileName to SAL301RConnector.txt

Refer to Figure 5-5 on page 83 for an illustration of these changes.
Chapter 5. Getting started

4. Refer to Figure 5-6 on page 84. In the Connector Specific Properties, change:
   - Application Password to smbP4$$word
   - Application User Name to db2admin
   - Archive Table Name to SAL301R_archive_event
   - Event Table Name to SAL301R_events
   - JDBC Driver Class to COM.ibm.db2.jdbc.app.DB2Driver
   - RDBMS Vendor to DB2
   - Database URL to jdbc:db2:SAL301R

5. Save to a file and to the project by navigating to File → Save → To project and File → Save → To file and name the file SAL301RConnector.cfg. Make sure the file is saved in the directory C:\IBM\WebSphereServer\connectors\SAL301RConnector.
5.5 Creating Business Objects

Create a Business Object (BO) through the Business Object Designer. The BOs directly correlate to the tables within the SAL301R database. The child objects and the parent object variables are taken from the SAL301R database variables.

1. Using the System Manager choose **Tools → Business Object Designer**. This will open the designer and a window prompt for the Business Object name, application-specific information and the project in which to create the BO.

2. Create the child objects first by entering this information:
   - **Business Object Name**: SAL301R_PROP_STS
   - **Application Specific Information**: TN=PROP_STS
   - **Create in Project**: TestProjICL
3. Enter the information as shown in Figure 5-7. Save this child object by clicking **File → Save**. The project to which you want to save the BO has already been identified upon creation of the BO.

4. Create the second child BO by clicking **File → New**. A window prompt for the new Business Object will appear; enter the following information as shown in Figure 5-8.
   - Business Object Name: SAL301R_PROP_TYPE
   - Application Specific Information: TN=PROP_TYPE
   - Create in Project: TestProjICL

5. After entering the information, save this child object by clicking **File → Save**.
6. Create the parent BO by clicking **File → New**. A window prompt for the new Business Object will appear. Enter the information as shown in Figure 5-9.

   - Business Object Name: `SAL301R_PROP`
   - Application Specific Information: `TN=PROP`
   - Create in Project: `TestProjICL`

7. After entering the information, save the parent object by clicking **File → Save**.
8. Create the GBO by clicking File → New. A window prompt for the new Business Object will appear; enter the following information. Enter the information as shown in Figure 5-10.

- Business Object Name: Property
- Application Specific Information: GBO
- Create in Project: TestProjICL

9. After entering the information, save the Generic Business Object by clicking File → Save. Creation of the BOs is complete.


### 5.6 Creating maps

Now that you have created the Business Objects, you can create the maps to connect the information that will be transferred from an ASBO to a GBO. In the System Manager, navigate to Tools → Map Designer. The Map Designer Express Tool opens.

1. In the new map creation window, select TestProjICL as the project to contain the map and click Next.

2. Select the SAL301R_PROP as the source BO by selecting the check box next to the BO, then click Next.

3. Select the Property as the destination BO by selecting the check box next to the BO, then click Next.

4. In the Name field, type SAL301R_PROP_to_Property. Select Application Specific to Generic and click Finish.
5. Click the **Table** tab to view and enter the information as it appears in Figure 5-11. This view allows you to use drop-down menus to select Source and Destination attributes, as well as the transformation rules.

<table>
<thead>
<tr>
<th>Table</th>
<th>Diagram</th>
<th>Messages</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Attribute</td>
<td>Destination Attribute</td>
<td>Source Type</td>
<td>Dest. Type</td>
</tr>
<tr>
<td>1</td>
<td>.USRID</td>
<td>.INV_AGENT_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>2</td>
<td>.PROPID</td>
<td>.INV_ID</td>
<td>Integer</td>
</tr>
<tr>
<td>3</td>
<td>.PROPRIOR</td>
<td>.INV_ASK_PRICE</td>
<td>String</td>
</tr>
<tr>
<td>4</td>
<td>.PROFSIDESC</td>
<td>.INV_STATUS</td>
<td>String</td>
</tr>
<tr>
<td>5</td>
<td>.PRODFIELD</td>
<td>.INV_TYPE</td>
<td>Integer</td>
</tr>
<tr>
<td>7</td>
<td>.PROPOSARADDF</td>
<td>.INV_DESC</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 5-11 Table view for mapping SAL310RProp ASBO to Property GBO*

If you use the Design tab view, the mapping appears as it does in Figure 5-12 on page 89. The arrows point to objects transformed from the ASBO to the GBO. This view can be difficult to use as the arrows become more complex.
Figure 5-12  Design view of mapping SAL310RProp ASBO to Property GBO

6. Save the map by clicking File → Save → To project.

7. Click Yes when prompted to compile the map.

5.7 Updating SAL301RConnector

1. Update the SAL301RConnector to include the BOs and the map created. To do this, double-click the connector in the TestProjICL and click the Supported-Business Objects tab.
2. Use the drop-down menus to select the Business Object Name and click the radio buttons to be certain there is Agent Support for these BOs. Refer to Figure 5-13.

3. Save to a file and to the project by clicking:
   - File → Save → To project
   - File → Save → To file

Name the file SAL301RConnector.cfg.

![Connector Configurator Express - ICS - SAL301RConnector* : TestProjICL](image)

**Figure 5-13** Connector associated Business Objects window

### 5.8 Testing the connector

In this section, we test to see if our new connector is working correctly. To do this, we use DB2 tools to make changes in the SAL301 database and then check that our new connector receives events with details of the database changes.
1. To test the connector, deploy the Business Objects, maps, and connector to the WebSphereICS. Deploy the Business Objects first and the connector last by using the drag and drop method, dragging the object from the TestProjICL and dropping it on the WebSphereICS server.

2. Shut down WebSphereICS by right-clicking WebSphereICS and clicking Shutdown → Gracefully.

3. Start the WebSphereICS by clicking Start → Programs → WebSphere Business Integration Server Express → Interchange Server.

4. Start the SAL301R Adapter by clicking Start → Programs → WebSphere Business Integration Server Express → Adapters → SAL301R.

5. Once the WebSphereICS is ready, in the System Manager right-click WebSphereICS and select Connect. In the connection window, click OK.

6. Open the IBM DB2 Control Center by clicking Start → Programs → IBM DB2 → General Administration Tools → Control Center.

7. In the Control Center, launch the Command Center. Refer to Figure 5-14.

8. In the Command Center, click the icon to the right of the Database connection field. This will bring up a listing of the databases on your local machine.
Navigate to the SAL301R database. Select the database and click OK. See Figure 5-15.

9. In the command field, you will see connect to SAL301R. Click the Execute icon on the command center toolbar. Under the command field, look for a connection to the database.

![Figure 5-15 Browse window to find SAL301R database instance](image)

10. To test, we will add a property to the SAL301R database and see if it shows in the Adapter log or the SAL301RConnector. There are two ways to perform this task:

- Using the SQL Assist tool
- Typing the SQL into the command field

11. Using the command field to create a new property, enter the SQL:

```sql
Insert into DB2ADMIN.PROP (PROP_ID, USR_ID, PROP_TYPE_ID, PROP_STATUS_ID, PROP_ADDR_CNTRY_CD, PROP_ADDR_POST_CD, PROP_ADDR_RGN, PROP_ADDR_MUNI, PROP_ADDR_STRT_NAM, PROP_PRICE) VALUES (3, 2, 1, 'US', '16801', 'NC', 'Some Municipality', '14 South Street', 205000)
```

12. Click the Execute icon. If the insert succeeds, check the adapter command window and check that the property event has reached the adapter. Example 5-1 on page 93 shows an example of the messages you should expect to see in the adapter command window. Because no collaborations exist that subscribe to the Business Object and event combination, the
adapter agent will treat the polled event as unsubscribed. If this appears, then the adapter is working properly. If not, re-check the Business Objects, maps and the connector to ensure that all variables match those in the example.

Example 5-1  Adapter polls for events

[Time: 2005/02/08 17:13:15.438] [System: ConnectorAgent] [SS: SAL301RConnectorConnector]
[Thread: appPolling (#1739983400)] [Type: esg: Thread:appPolling:executing poll call]
[Time: 2005/02/08 17:13:15.438] [System: ConnectorAgent] [SS: SAL301RConnectorConnector]
[Thread: appPolling (#1739983400)] [Type: esg: :Polling App Connector.]
[Time: 2005/02/08 17:13:15.438] [System: ConnectorAgent] [SS: SAL301RConnectorConnector]
[Thread: appPolling (#1739983400)] [Type: esg: :[Type: Warning] [MsgID: 33007] [Mesg: Entering pollForEvents] ]
[Time: 2005/02/08 17:13:15.453] [System: ConnectorAgent] [SS: SAL301RConnectorConnector]
[Thread: appPolling (#1739983400)] [Type: Error] [MsgID: 20026] [Mesg: Unsubscribed event from application: Object Name:SAL301R_PROP, Verb: Create.]
[Time: 2005/02/08 17:13:15.469] [System: ConnectorAgent] [SS: SAL301RConnectorConnector]
[Thread: appPolling (#1739983400)] [Type: esg: :[Type: Warning] [MsgID: 33009] [Mesg: Exiting pollForEvents] ]
[Time: 2005/02/08 17:13:15.469] [System: ConnectorAgent] [SS: SAL301RConnectorConnector]
[Thread: appPolling (#1739983400)] [Type: esg: :checkStatus]
13. Using the SQL Assist, select **Insert** as the SQL statement. In the outline, select **INSERT INTO** and select **DB2ADMIN** from the available tables and **PROP** from DB2ADMIN, then click the <-> button to add PROP to the **Selected Table** field.

14. Click **VALUES** in the Outline and enter the values as shown in Figure 5-17 on page 95.

15. Once the values are entered, click the **Run** button. A window will appear that states a row has been added. Click **OK**.

---

**Figure 5-16** Select Insert as the SQL statement and create in PROP
16. In the command window for the SAL301R, you should see that a new event has passed to the connector. This will show that the connector, and Business Objects are working properly. You may close the Control Center and the Command Center after identifying that the new property has passed through the Adapter.

17. Once all the components work properly, you may synchronize the BOs, maps, and so forth with the CVS so other members may view and use these components. To do so, right-click the WBIExpressLibrary ICL and navigate to Team → Synchronize outgoing changes.

5.9 Helpful resources

A few good online resources exist for the toolset, collaborations, DB2 and WebSphere MQ. These resources are available in either PDF form or in other
downloadable formats. The information provided in these sites will provide further troubleshooting assistance and getting started tips.

- **Information Center:**
  

- **Information to download:**
  
  - WebSphere help system
  - Documentation plugins
  - Toolset help pages

- **Available books in PDF format:**
  

- **Collaborations:**
  


- **WebSphere Business Integration online products information site:**
  

- **DB2 online information centre:**
  

- **WebSphere MQ infocenter (download)**
  

- **WebSphere Application Server infocenter online:**
  
In this part, we provide a detailed examination of the integration features and functions provided by WebSphere Business Integration Server Express.
Introduction to Toolset Express

This chapter provides an overview of the development tools provided by Toolset Express as part of WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus. The discussion is divided into the following sections:

- 6.1, “Introduction” on page 100.
- 6.4, “Connector Configurator Express” on page 119.
- 6.5, “Test Connector” on page 126.
- 6.6, “Relationship Designer Express” on page 130.
- 6.8, “Test Environment” on page 148.
6.1 Introduction

WebSphere Business Integration Server Express provides a toolset for development that acts as an easy-to-use environment, allowing a user to develop, configure and deploy Business Object synchronization and process automation logic using a simplified graphical workbench.

These tools, provided with IBM WebSphere Business Integration Server Express, include sets of tools for development and administration.

The System Manager tool allows access and manipulation of the integration components that are used in the business data exchange solution with IBM WebSphere Business Integration Server Express. The System Manager allows for the creation of multiple projects.

In order to allow system monitoring and administration, WebSphere Business Integration Server Express provides Administrative Tools. These tools can be used during development or after implementation, depending on the administrative objective. They are the Flow manager, Log Viewer, Relationship Manager, and System Manager.

WebSphere Business Integration Server Express Toolset provides the tools for the development of integration components. The toolset provided by WebSphere Business Integration Server Express is based on the Eclipse framework and allows a flexible and consistent user interface within the entire platform. The Eclipse platform is an open-source development environment which enables the developer to write plug-ins to the integration solution.

This framework provides an environment that is driven by a team-based visual design approach. The environment provides the option to allow source code management by a third-party program. Consequently, it simplifies business process design by reducing the necessity for specialized programming.

The Web services open standard is supported by WebSphere Business Integration Server Express. Using the development toolset provided by WebSphere Business Integration Server Express, you can enable Web services as part of an integration solution, and include services as part of the process model. In the same way, it is possible to expose processes as Web services to other applications.

The functionality of drag-and-drop gives swiftness to the creation process and enables connectivity to back-end systems.

WebSphere Business Integration Server Express presents its development environment as a graphical workbench, which helps simplify its main purpose and gives more control over the design of business critical processes. This
allows more agility and the ability to change processes on demand to meet customer needs. The WebSphere Business Integration Server Express Toolset provides the integration solution designer with the ability to adjust the integration system to the speed of business changes. The Toolset also provides the customer with a quick response to market demands and costs reduction.

The WebSphere Business Integration Server Express development toolkit offers the design tools needed to develop and implement the integration solutions, including:

- **Business Object Designer Express**: a graphical tool to create and maintain Business Objects required to the solution.
- **Connector Configurator Express**: enables the adapter configuration using information such as port, the Application Specific Business Objects (ASBOs) to be supported, host name.
- **Map Designer Express**: a graphical tool to generate and maintain Business Objects maps. It allows Application Specific Business Objects (ASBOs) to transform to and from the IBM WebSphere Business Integration Generic Business Object (GBO) format.
- **Relationship Designer Express**: defines the relationships between application Business Objects. This definition is necessary for data synchronization across the applications.
- **Test Connector**: provides the test functionality between the agents without the complexity of application connectivity.
- **Process Designer Express**: allows you to sketch and graphically refine the process flow of the business (only available in the WebSphere Business Integration Server Express Plus version).

This chapter presents an overview of the development tools provided by Toolset Express and embedded in the WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus products.

Figure 6-1 on page 102 shows the WebSphere Business Integration Server Express Plus Toolset window, with links to all the development tools previously mentioned.
The advantages of IBM WebSphere Business Integration development tools are as follows:

- Availability of a visual drag and drop flow modeling tool that makes development easier.
- Possibility of creating Java code using visual tools so only reduced Java programming is needed during development.
- Can be launched from Process Designer or the Map Designer, depending on the focus the designer wants to use during development.
- Simplicity when generating transformation rules for mapping.
- Enables actions generated for collaboration.
- Provides a function block library and math, date, and string handling.
- Comes with cross-referencing and logging/error handling.

In the following sections, we present details about each of these tools.
6.2 Business Object Designer

This section presents the basic concepts of Business Object Designer Express. This tool enables the developer to manipulate Business Objects in a very easy and fast manner.

The Business Object Designer tool provides a graphical environment for generating and maintaining Business Objects at the integration development stage. This tool allows the developer to create Generic Business Objects and Application Specific Business Objects.

The Business Object Designer tool allows the developer to create a BO definition manually and using an Object Discovery Agent (ODA). Each ODA generates definitions for a specific application. The Business Object Designer provides simplification of the Business Object creation process, thanks to this feature (ODA), which inspects a DataSource (JDBC, for instance) and automatically generates the Business Object.

These two types of a Business Object creation are:

- Manual: normally used to create a Generic Business Object, a simple Application Specific Business Object, or to modify a Business Object generated by an Object Discovery Agent. Figure 6-2 shows the window showing where a new Business Object is created manually. Through this graphical interface, the user sets the Business Object name, information about the application to which it is related, and provides the project name where the Business Object will be included.

![Figure 6-2 Creating new Business Object](image)

- By ODA (object discovery agent): normally used to create an Application Specific Business Object. The ODA examines specific entities of the application, defines the elements of those objects that correspond to Business Object attributes and the properties of each attribute, and generates
the Business Object definition. Figure 6-3 shows the creation of a Business Object using ODA. Through this graphical interface, the user defines the agent to locate the ODA, agent host, and port.

![Business Object Wizard - Step 1 of 6 - Select Agent](image)

Press "Find Agents" to locate ODAs in your subnet (3.42.171.xxx). You will be given an estimate of the time required to complete the operation and a means to cancel the search.

If you have an ODA that is running on a machine outside subnet 3.42.171.xxx press "Configure Discovery" to include that machine in the search.

Agent's name: To be discovered
Agent's host: 
Port: 

Figure 6-3   New Business Object using ODA

Figure 6-4 on page 105 shows Business Object generation using attributes. The Business Object attribute values are generated by typing each field name and defining all its attributes.

The information contained in the Business Object is represented by an attribute in the Business Object definition. It must be specified for each attribute in the definition of the object.

Within this graphical interface, it is possible to define the field type (key), foreign key, required attribute, cardinality, maximum length, and application-specific information.

The key, foreign key, required attribute, and cardinality columns specify whether the current attribute is the Business Object's primary or foreign key, whether the attribute's value is required, and whether the attribute represents a child Business Object.
The Max length box specifies the maximum number of bytes available for this attribute's value, and the default value is 255.

![Figure 6-4 Business Object design - attributes definition](image)

It is possible to introduce specific information regarding supported verbs related to the Business Object through a graphical interface, as seen in Figure 6-5 on page 106. The developer specifies the verbs that the Business Object supports and these verbs represent the triggering events that the Business Object sends to the integration broker. The Business Object supports these four verbs that appears, by default, in the General tab:

- Create
- Delete
- Retrieve
- Update
The field type definition of the Business Object Designer tool enables the developer to select from a combo box which variable type is required for the specific field. The user does not have to type in the field type for each field. The Business Object may have many attributes that require definitions, as shown in Figure 6-6 on page 107.
Once the Business Object definition is completed, it can be saved by clicking **File → Save**. This can be done using an ICL-based project or using a local project. In this case, it will be necessary to specify a file name and local directory in which to save the definition.

The Business Object Designer tool is a graphical interface that allows the developer to quickly generate the ASBO/GBO that will play a role in the integration system. It gives the development process the speed that the solution integration requires.

### 6.3 Map Designer Express

A Business Object map defines the transformation steps that specify how to match the value for each attribute in a source Business Object to the destination Business Object. Creating a map is the process where the developer specifies
transformation steps for each attribute that must be transformed during the integration solution.

Map Designer Express is a development instrument for Business Objects mapping. It is a graphical tool used to generate and maintain Business Object maps.

Map Designer Express allows the developer to specify transformation steps by moving an attribute from a source to a destination attribute with the same data type, using drag-and-drop. Map Designer Express then generates Java code required to perform this transformation automatically.

Other transformations, such as splitting a source attribute into multiple destination attributes or joining multiple source attributes into a single destination attribute, can be performed using Map Designer Express. The developer defines the delimiter on which to split or join, and the tool generates the necessary Java code.

This tool allows a developer to execute the transformation of Business Objects to and from IBM WebSphere Business Integration’s Generic Business Object (GBO) format. The Business Objects (ASBO- Application Specific Business Object and GBO - Generic Business Object), once created, can be mapped for transformation within the integration solution using the graphical interfaces shown next.

At the start of the mapping process, Map Designer Express can be selected from the development tools menu or by choosing the specific icon in the System Manager tool bar. Map Designer allows a user to create, view, edit, and modify maps in the System Manager, on a project basis, that is, a logical group of entities for management and deployment purposes.

When Map Designer Express establishes a connection to System Manager, it obtains a list of Business Objects that are defined in the current project. If a business project is added or deleted, System Manager updates Map Designer, which updates the list of Business Object definitions.

Using the Map designer tool, a new map can be generated or a pre-existing map can be selected and edited. It is then necessary to define which is the Business Object source. This can be done by setting the option box related to the Business Object, in the Choose source Business Objects window. This window is shown in Figure 6-7 on page 109. This graphical interface provides a view that lists all Business Objects.
The developer can easily choose which Business Object will be the data source. After that, the next mapping step is to define which Business Object will be the data destination. This definition depends on the business process modeling.

This can be done in the same way that the Business Object source was defined, using the combo box that contains a list of Business Objects with their attributes. The developer selects the appropriate destination Business Object from this list. Figure 6-8 on page 110 shows the *Choose destination Business Object* window where the designer can set the Business Object.

---

**Figure 6-7  Source Business Object window**

The developer can easily choose which Business Object will be the data source. After that, the next mapping step is to define which Business Object will be the data destination. This definition depends on the business process modeling.

This can be done in the same way that the Business Object source was defined, using the combo box that contains a list of Business Objects with their attributes. The developer selects the appropriate destination Business Object from this list. Figure 6-8 on page 110 shows the *Choose destination Business Object* window where the designer can set the Business Object.
After defining the Business Object DataSource and destination, in the transformation process, it is necessary to specify the map name for the project. This can be done by simply typing the name of the map in the New Map box. Figure 6-9 on page 111 shows this box.
In this graphical interface, the developer defines which is the transformation direction: Application Specific Business Object to Generic Business Object, or Generic Business Object to Application Specific Business Object.

After defining the map and choosing the Business Object source and destination, it is necessary to define the relation between the Business Objects in terms of data transformation rules. Figure 6-10 on page 112 shows a way in which this mapping can be done.

The Diagram tab of the Map Designer Express allows the developer to define and review transformations using a drag-and-drop tool. The Diagram tab shows the following areas:

- Business Object browser: displays the map and project name, in the top left portion of the window. This browser uses a hierarchical format to list the Business Objects in the project System Manager when Map Designer Express is connected to System Manager.

- Map workspace: displays the information about the map in the workspace. The workspace displays a Business Object view for each source and destination Business Object used in the map. Each Business Object lists the attributes defined in it. It will display some or all of them, depending on what
viewing mode is currently selected. In the case of a destination Business Object or temporary Business Object, the Business Object window also lists the transformation rule and comments associated with the attribute. The map workspace enables the developer to add, delete, or modify transformations. Lines connecting attributes represent the transformation route between these attributes.

On the left side of the map workspace is the source Business Object. On the right side is the target Business Object which will be filled up with data from the source Business Object.

This mapping can be done easily by using arrows to determine which fields will be filled with data from the Business Object. It is not necessary to set all fields, just those which the developer is interested in mapping, considering the business process modeling.

Figure 6-10  Mapping Business Objects
On the right side, where the Business Object is, it is necessary to define which rule will be used in the Business Object transformation. This choice needs to be made for each field that has been marked for mapping.

As mentioned previously, the Map Designer offers the choice of viewing all the attributes of the Business Objects or just those attributes that will be transformed or linked by map rules. Figure 6-11 shows this second option.

![Linked attributes view](Image)

Figure 6-11  Linked attributes view

Figure 6-12 on page 114 shows the option box that can be opened at the rule column. There are different transformation choices. The box defines not only the transformation rule but also the code for this attribute's transformation step. Each field can have a different transformation rule adopted. This functionality gives the integration solution significant flexibility.

The tool provides a box that contains a list of standard transformations:

- None (no transformation)
- Join
Move
Split
Set Value
Submap
Cross-reference
Custom

Figure 6-12 Transformation rules box

Figure 6-13 on page 115 shows a mapping process executed using the table box. The table displays mapping information in a format that lists all mapping attributes and transformations. The Table tab consists of the attribute transformation table and Business Objects panel. This window allows the user to define the Business Objects mapping through a table format.
Using this graphical interface, the developer can see and check the Business Object source and destination, all fields of the Business Objects, the fields marked for transformation and the transformation rules adopted. With this graphical interface, the developer can check whether the fields set have variable type coherency.

The Map Designer tool provides an interface for testing maps and verifying the results. It is then possible to run tests to verify that the transformations are being implemented properly. This can be done through the test option window. Figure 6-14 on page 116 shows this particular tool.
This graphical interface requires the developer to specify which Business Object calling context will be applied. The options box is shown in Figure 6-15 on page 117.
After defining the calling context, the test scenario can be defined by the developer. In this step, in the Variable value column, the user types values coherent with the variable type specified. Then the user runs the test.

If the test is a success, the field values are repeated in the Values column of the Business Object destination, as shown in Figure 6-16 on page 118, and the test final message is Test run finished.
If the test fails, the values defined in the Business Object source will not be transferred to the Business Object destination.

The Map Designer gives the developer the possibility to validate the map. The validation functionality is activated with the Validate Map option in the File menu. Map Designer provides options to perform validations on the map:

- Show warning if verb not mapped
- Show warning if key attribute not mapped
- Show warning if required attribute not mapped
- Show warning if child Business Object not mapped

The debug menu of the Map Designer tool provides access to the debugging functionality. It provides the following options:
Run Test: connects to a server and starts the test run of a map that is opened from a project.

Stop Test Run: stops the test run of the map.

Breakpoints: displays all breakpoints for the map.

Clear All Breakpoints: clears all breakpoints in the map.

Continue: continues execution after it stops at a breakpoint.

Step Over: continues execution after it stops at a breakpoint, but stops execution before the next attribute execution.

Advanced: provides options for connecting to a server for testing a map that resides in the server (Attach) and disconnecting from a server and closing a map (Detach).

Toggle Breakpoint: sets a breakpoint in a map, which pauses execution just before the selected attribute's transformation.

6.4 Connector Configurator Express

In an integration solution, a connector works as a meta-data driver. Meta-data is application-specific data that is stored in the integration system in order to assist the connector in its interaction with the application. In WebSphere Business Integration Server Express, the connector handles each Business Object that it supports, based on meta-data encoded in the Business Object definition.

Business Object meta-data includes the structure of a Business Object, the settings of its attribute properties, and the content of its application that will be integrated.

Connectors consist of two parts:

- The connector framework and the application-specific component. The connector framework works as an intermediary between the integration broker (ICS, for instance) and the application-specific component. The connector framework provides services to both the integration broker and the application-specific component; it receives and sends Business Objects, and handles the exchange of startup and administrative messages.

- The application-specific component contains code developed specifically for a particular application or technology (JDBC, for instance).

The connector allows the integration broker to exchange Business Objects with an application built on any database supported by a driver that follows the connector technology specification. Figure 6-17 on page 120 shows the
technologies that the WebSphere Business Integration Server Express adapter supports.

![Figure 6-17 Adapter's technologies](image)

A connector can be started using the window connector options shown above. The connector connects to the application database, and the configuration of the database URL parameter allows the developer to specify the name of the database server to which the connector will connect. When the connector is started, it starts a connection pool with the database and uses connections from this pool for all processed transactions. When the connector is closed, all connections in the pool are closed as well.

When the connector receives a request to perform a connection, it processes Business Objects recursively, according to the BO hierarchy; this means that each child Business Object is processed until all individual Business Objects are processed. The order in which the connector processes child Business Objects and Business Objects depends on whether the child Business Objects are contained with or without ownership and whether they are contained with single cardinality or multiple cardinality.

WebSphere Business Integration Server Express provides Connector Configurator Express. It is a tool that enables the developer to configure the adapter with information that is needed to provide the connection between Business Objects, databases, and applications.

This tool allows the developer to define information such as the port, the Application Specific Business Objects (ASBOs) to be supported and the host name, for instance.

In the beginning, the developer must create the connector or define a pre-existing one. To create a new connector, the user chooses the option **New Connector** in the File menu. Then the user defines the connector’s name in the appropriate box.
Figure 6-18 on page 121 shows the New Connector generation window.

![New Connector generation window](image)

**Figure 6-18  New connector generation**

Through this graphical interface, the developer can define the integration broker. Using Interchange Server Express, the broker can be set to ICS, then the user can define the template name. This tool offers options, as shown above in Figure 6-18.

The Connector Configurator Express presents two types of configuration properties: standard configuration properties and adapter-specific configuration properties. Properties values of the adapter must be defined before running the adapter.

The developer defines the configuration values of the connector at startup. During a runtime session, it is possible to execute some changes to the values of connector properties. Changes to some connector configuration properties, such as AgentTraceLevel, take effect immediately. Changes to other connector properties require a component restart or a system restart after a change. It is
possible to see which properties can be changed with a dynamic update and which properties will require a component restart by looking at the Update Method column in the Connector Standard Properties window of Connector Configurator Express. Figure 6-19 shows the Standard Properties configuration graphical interface:

![Connector Configurator Express - Standard Properties](image)

In the Standard Properties configuration window, the user must define each connector property and the specific value adopted for the configuration; the update method used is set by default.

Standard configuration properties provide information that all connectors use. When the user sets configuration properties in Connector Configurator Express,
the BrokerType property is set to Interchange Server Express. Properties are shown in the Connector Configurator Express window.

Some of the properties are absolutely relevant, specific to the solution environment, and must be defined to allow the connection for the integration solution. These are:

- AdminInQueue
- AdminOutQueue
- ApplicationName
- BrokerType
- DeliveryQueue
- DeliveryTransport
- FaultQueue
- MessageFileName
- RequestQueue
- ResponseQueue
- SynchronousRequestQueue
- SynchronousResponseQueue

The configuration of the specific properties provides information that the connector needs at runtime. The user must configure the connector in order to enable it in the integration solution. Figure 6-20 on page 124 shows the configuration interface to application-specific properties of the connector.

Some of the properties are absolutely relevant, and must be defined to allow the connection in the integration solution. They are:

- ApplicationPassword
- ArchiveTableName
- DatabaseURL
- RDBMSVendor
- UniqueIDTableName
- EventTableName
- JDBCDriverClass
- ApplicationUserName

The most important factor to emphasize about these properties is that RDBMS defines the database vendor. This is the property that allows a developer to set databases other than the IBM database (DB2). It is a very important characteristic because it gives the user flexibility, considering that a middle market customer may have different database environments.
The Connector Configurator Express provides several levels of message logging and tracing. The connector uses the adapter framework to log errors and trace messages. Error messages are recorded in the log file, and trace messages and their corresponding trace levels (0 to 5) are recorded in a trace file. Figure 6-21 on page 125 shows the window where the user can configure the log and trace file names, as well as the trace level.
This tool provides the user with a box where it is possible to specify which file will receive the error information and trace messages.

The Connector Configurator Express provides an easy way to support Business Objects definitions. The developer can specify which Business Objects will be enrolled in the integration transaction. Figure 6-22 shows the graphical interface where the developer can perform this configuration.

In order to help the user in this task, this product provides a combo box where it is possible to choose which Business Objects will be part of the connector. In this box, the user can see all Business Objects defined and enrolled in this project. Figure 6-23 on page 126 illustrates this.
One of the most important aspects related to connector configuration is defining the associated map. This map will give the logical route to the connector in order to specify the relation between the application, application Business Object, and Generic Business Object. Figure 6-24 shows the graphical interface where it is possible to define the associated maps.

6.5 Test Connector

WebSphere Business Integration Server Express provides a tool, called the Test Connector, which simulates the activities of a connector. This tool allows the user to test the integration components without the complexity of running an actual connector.
To start the Test Connector tool, click **Start** → **Programs** → **IBM WebSphere Business Integration Server Express** → **Toolset Express** → **Development** → **Test Connector**.

Test Connector uses profiles to store the information that it needs to emulate a connector. In such a case, it is necessary to create a profile for the connector that will be tested.

Figure 6-25 shows the window where is possible to create a new Test Connector profile. It defines the connector configuration file, the extension of which is *cfg* (configure). After that, the user must define the connector name and the broker type. Using WebSphere Business Integration Server Express, the broker type will be ICS, the default. And, lastly, the developer must specify the server where the test will run.

![Figure 6-25 Creating a new Test Connector profile](image)

The user can then select which test profile will be used through graphical interface. Figure 6-26 on page 128 shows this graphical interface. It shows all connector profiles created; the user can select one of them.
Once the connector profile selection is complete, it is necessary to define:

- **BO Type**: the tool provides a combo box where the user can select the Business Object.
- **BO Instance**: the user can choose a pre-existing instance or create a new one.
- **Verb**: the user defines which verb will support the connector test.

This configuration is executed through the graphical interface shown in Figure 6-27 on page 129. Then the attributes values can be filled.
Figure 6-27 Test Connector configuration

The test connection is executed through the Connect command found in the File menu, as show in Figure 6-28 on page 130.
Figure 6-28  Connection test

The Test Connector displays messages in the Output pane at the bottom of the window as it attempts to emulate the connector.

When it finishes connecting, it displays a message in the Output pane indicating that it is ready and populates the BOType list in the Supported Business Objects pane.

6.6 Relationship Designer Express

The WebSphere Business Integration Server Express tool set provides a tool called Relationship Designer Express for creating and modifying relationship definitions. A relationship definition specifies the association between the integration solution participants. With this tool, the user can specify the DataSource and other properties associated with each participant of the integration solution. In other words, this tool provides a graphical interface that
allows the developer to define the relationships between application Business Objects and their attributes to simultaneously synchronize data across multiple applications.

It is possible to launch Relationship Designer Express tool from the System Manager or by using a system shortcut: click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Relationship Designer Express.

The System Manager is the only tool that interacts with the server. It imports and exports entities (relationships, maps) between Interchange Server Express and System Manager projects. The Relationship Designer Express tool connects to the System Manager and the user can view, edit and modify the entities of a project basis, that is, a logical grouping of entities for managing and deployment purposes.

The developer must first select which project the relationship will be in. To select a project, the user must perform these two steps: from the File menu, select the Switch to Project option and then, in the Switch to Project submenu, select the name of the project, as shown in Figure 6-29.

![Relationship Designer Express](image)

Figure 6-29   Starting a new relationship design

Relationship Designer Express lists the relationship definitions stored in the current project; it appears on the left side. After project selection, the user can introduce a New Relationship for the project, as shown in Figure 6-30 on page 132.
In a relationship, whether pre-existing or new, the user must add participants. This can be done by clicking **Add Participant Definition**, as shown in Figure 6-31. This step gives the developer flexibility in terms of relationship definition.

A participant definition option is necessary to specify which participant type will be applied in this relationship. This definition is executed through the Participant Type option, as shown in Figure 6-32 on page 133. The Participant Types window shows a list of available data types in the current project that can be associated with a participant. It is then possible for the user to define a new participant, and select which data or Business Object will be involved in this specific relationship.
As shown in Figure 6-33 on page 134, in this relationship definition list, the contents of each relationship definition appear in hierarchical mode. It is possible to expand the relationship name by clicking it to see the list of its participant definitions, participant types, and associated attributes. With this functionality, a new relationship is easy.

If the developer changes the project, adding or deleting a Business Object using Business Object Designer Express, the System Manager notifies Relationship Designer Express, which updates the list of Business Object definitions.
Figure 6-33  Listing all the project's Business Objects

The Relationship Design Express tool provides the ability to define each attribute that will participate in the relationship. When the attributes are shown, the key attribute is easy to see (as in Figure 6-34 on page 135). The developer can then set this attribute as part of the relationship.
The relationship tree is then defined and the user can verify the relation hierarchy as shown in Figure 6-35.
Relationship Designer Express allows the developer to verify the environment configuration by using the Advanced Settings function. For instance, in Figure 6-36, you can see the database URL and data base type.

![Advanced Settings](image)

*Figure 6-36  Advanced settings configuration*

### 6.7 Process Designer Express

WebSphere Business Integration Server Express Plus provides the Process Designer Express tool in order to configure the collaborations in the integration solution. Collaborations describe business processes that run within IBM Interchange Server Express (ICS). These business processes contain the business rules and logic for application integration. A collaboration performs operations on Business Objects that enable integration, such as:

- Obtain and handle one or more attributes in the triggering event
- Send a Business Object as a request to an application in order to create, delete, or update a specified entity
- Send a request to an application in order to retrieve an entity

Process Designer Express tool provides a graphical interface to business process modeling whereby the developer can sketch and refine the logical flow of business processes. It uses UML notation to represent the business process. This tool can automatically generate Java code underneath the graphical
notations and can extend the generated Java code to support complex business process modeling.

The collaboration configuration starts with a collaboration template installation. A collaboration template contains the execution logic of the collaboration, but it is not executable. To execute a collaboration, it must be linked to a collaboration object, created based on this template. The collaboration object can be executed after the developer configures it, defines with which connectors or other collaboration objects it will be bound, and specifies other configuration properties.

A collaboration template is designed using the Process Designer Express tool, which stores the specific information in System Manager. The development of the collaboration template can be executed following these steps:

1. Create the collaboration template
2. Build the parts of the collaboration template
3. Compile the collaboration template

Through a graphical interface, the user can create a new template whereby it is possible to declare variables, write code, and so on. Process Designer Express provides generic collaboration templates to ease the development process. In such a case, the user develops a collaboration template using a graphical interface, but the collaboration is a program, in the form of a Java class.

Developing a collaboration template using Process Designer Express involves specific activity diagrams. It is possible to create activity diagrams, which are graphical and symbolic descriptions of the business process flow. Compiling the template converts the diagrams and their associated code to an executable Java class.

Figure 6-37 on page 138 shows the window wherein a new collaboration template can be created. It is possible to access this graphical interface by clicking **File → New** in the Process Designer Express tool. In the Project field, the developer can choose the project where this template will play a role. In the Template Name field, the user defines its name and may also provide a description.
A collaboration template consists of:

- Definitions, wherein the developer binds Business Objects, ports and supported verbs.
- Scenarios, which specify sets of actions.
- Activity diagrams, which describe these actions wherein code fragments represent individual actions.
- Messages, which hold the text used in logging.

Figure 6-38 shows the collaboration template structure.

The Process Design Express tool provides a template of the collaboration; there are four tabs for configuring the collaboration’s properties. These are:
General

Enables the user to define the following information:
- Template description
- Support for long-lived business processes
- Minimum transaction level; package information
- General property information

Declarations

Enables the user to view system-generated template variables and declare and edit template variables.

Properties

Enables the user to specify the name, type, and value of user-defined collaboration template properties, as well as to define collaboration configuration properties.

Ports and Triggering Events

Enables the user to define the ports and triggering events for the collaboration template.

Figure 6-39 on page 140 shows the properties’ definition graphical interface, as described above. The Ports and Trigger Events tab includes the following boxes:

- Port: this is where the user sets the ports the destination application retrieves, the attributes’ source From, and the attributes’ destination To. It is possible to define more than one Business Object destination just by clicking the Add Port button.

- BO Type: this is where the user defines which Business Object will be used for application data retrieval, which Business Object will be the data source, and which Business Object will be the data destination. The Process Designer Express tool allows to developer choose the Business Objects, providing a combo box for each BO type.

- Create/Delete/Retrieve/Update: these boxes allow the developer to define the supported verb of each Business Object. The user has the option to choose main or retrieve when the action is applicable to the Business Object.

This graphical interface provides the possibility of Business Object validation through the Validate BOs button positioned at the bottom of the window.
At the beginning of collaboration template development, the business logic of the collaboration is divided into one or more scenarios. A scenario specifies how the collaboration responds to a trigger positioned in the logical flow. The scenario is a method which describes the actions that will be taken by the collaboration. Process Design Express allows the user to create one scenario or several different ones.

For each scenario, the developer creates the activity diagrams that graphically describe the process enclosed in the scenario. It is possible, through this diagram, to design the execution flow. Activity diagrams are based on UML. The use of visual programming in diagrams makes it easy to create a scenario and reduces the amount of actual coding. The steps involved in the activity diagram are the individual actions (code fragments). Figure 6-40 on page 141 shows a typical activity diagram where it is possible to see, for instance, a start point, actions, decisions, and transition links.
Each scenario contains one or more top-level diagrams as the entry point of the scenario during execution and contains the subdiagrams that describe the logic flow. Subdiagrams can divide the details of scenario logic into multiple levels. Process Designer Express presents the activity diagram like a flow chart, but the activity diagram can create the executable Java code that it represents. Figure 6-41 on page 142 shows a diagram and subdiagrams structure.
The action is the basic unit of the activity diagram and it is represented by a rectangle. An action defines a unit of work in the collaboration and the Process Designer Express generates the Java code related to this action. In such a case, the activity diagram has multiple execution paths, that is, a set of symbols and links between the top Start symbol to the End symbol at the end of the flow. The developer can generate the activity diagrams using these symbols. The symbols are positioned in the Process Design Express tool bar and are shown in Figure 6-42.

**Figure 6-41  Diagram and subdiagrams of a collaboration template’s scenario.**

Process Designer Express allows the developer to customize the code underneath the action. This functionality gives the user great flexibility in using...
and reusing the templates in different projects. Figure 6-43 shows the Action Properties window that the developer can use to modify the action Java code fragment. The user accesses this graphical interface just by clicking the Action symbol in the activity diagram.

![Action Properties window](image)

**Figure 6-43  Action properties editing window**

After specifying the definitions, the user defines the scenarios, designs the activity diagrams, and formats the messages; the collaboration is now configured, but it is not executable. The collaboration template must be compiled. The collaboration compilation process creates three files that the collaboration runtime uses. The files applied in a collaboration are .class, .java and .txt.

In the compile process, the Process Design Express automatically creates the collaboration files in the Integration Component Libraries of the current project, within the System Manager. The developer must compile the collaboration template to the current project before saving. It is then possible to compile by clicking the Compile button in the Process Designer Express tool bar. If the compilation is successful, the user will be informed via a message in the message box. Otherwise, the message will be a compile error description: Compiled failed.

After creating and compiling the collaboration template, which contains the collaboration's execution logic, the developer must perform the following actions in order to execute the collaboration:

1. Create a collaboration object: a collaboration object is an instance of a collaboration template and it can be created using the System Manager.
Figure 6-44 shows the window where it is possible, using System Manager, to create a new collaboration object. This can be done by right-clicking the **Collaboration Objects** project option.

After that, the developer can select the collaboration template that will support the collaboration object. Figure 6-45 on page 145 shows the graphical interface where the developer can do this. The Process Designer Express tool shows the user all collaboration templates that belong to the current project, in the Template Name box. The user chooses the proper template for the collaboration object. In the Collaboration Object Name box, the developer defines its name.
2. Configure the collaboration object: the collaboration object becomes executable after its configuration. At collaboration object configuration, connectors and/or other collaboration objects must be bound, and other configuration properties must be defined. The process of specifying the objects with which a collaboration object interacts is called binding. A collaboration object can be bound to any of the following: a connector, other collaboration objects, or access clients with which a collaboration object interacts. When the developer binds the collaboration object and specifies the values for its configuration properties, the collaboration object becomes executable. Figure 6-46 on page 146 shows the window where the user can define the objects and/or collaboration objects that will be part of the integration solution.

The Port and Business Object Definition rows show the collaboration template configuration properties (Figure 6-39 on page 140). The Type row gives the developer the ability to specify the type of connection that will be applied at the bind process: connector, collaboration or Web service. In the BindWith row, the developer can specify the connector, collaboration or Web
service that will be used. This tool provides a combo box with all connectors of the project.

Figure 6-46  Binding ports to connectors

After that, the developer goes to the Collaboration General Properties window, where it is possible to specify the transaction and tracing levels for the collaboration.

Finally, Process Designer Express provides the collaboration object view in the System Manager. This can be a graphic view or a tree view. Figure 6-47 on page 147 and Figure 6-48 on page 147 show these options.
Figure 6-47  Collaboration object - Graphic

Figure 6-48  Collaboration object - Tree
Summarizing, the stages of collaboration development are as follows:

1. Design the business process that the collaboration will implement.
2. Create the Business Object definitions.
3. Create the collaboration template, with each scenario and its activity diagram.
4. Customize any required code fragments and create the message text.
5. Compile the template.
6. Create and configure the collaboration object from the collaboration template.

### 6.8 Test Environment

IBM’s WebSphere Business Integration Server Express provides the Integrated Test Environment tool. It is a workbench that enables the developer to test the integration system. This test is focused on a collaboration object that manages the integration solution of the business process.

In order to execute the test in the Integrated Test Environment, it is required that the user register the Interchange Server Express as a test server, create a test unit, deploy the components in the interface to the server, start the server, emulate the connectors in the interface, and exchange Business Objects between the connectors. This tool allows the user to perform some of these tasks in multiple ways. For example, it is possible to deploy components to the server before the test unit preparation, or to deploy all of the components for a test unit using the Task Manager view, or to deploy single components by using the Test Unit view. This provides flexibility to the developer when the test environment is being prepared. However, it is suggested that the user deploy the integration solution components to the Interchange Server Express beforehand in order to avoid the need to compile maps and collaboration templates as part of the testing process. Another reason is that when connectors are deployed, the developer must restart the server to start them, and this is not efficient to the testing process.

To start the preparation of the test environment, the user must ensure that all of the components required to test are in an active state; this can be done through the System Monitor or the InterChange Server Component Management view.

The first step is to register the Interchange Server Express as a local test server. This can be done by right-clicking the InterChange Server instance node in the Interchange Server Express Component Management view and choosing Register Server from the context menu. Figure 6-49 on page 149 shows this step. At the Register server window, the user can type the name of the Interchange Server Express instance in the Server name field or just browse for the Interchange Server Express instance on the network, clicking the Browse option. The System Manager finds the active servers on the network and lists
them, and the developer selects the relevant one. After that, the user types the user name in the User name field (by default, the user name is admin), as well as the password (by default, for the user name admin, the password is null).

In order to register Interchange Server Express as a local test server, the developer must enable the **Test Server** check box and type the full path to the Interchange Server Express product directory in the Test Server Installation Path field. This can be done using the **Browse** button. Figure 6-50 on page 150 shows this registration step.

---

**Figure 6-49  Select register server**

In order to register Interchange Server Express as a local test server, the developer must enable the **Test Server** check box and type the full path to the Interchange Server Express product directory in the Test Server Installation Path field. This can be done using the **Browse** button. Figure 6-50 on page 150 shows this registration step.
Figure 6-50  Register test server

In such a case, System Manager registers Interchange Server Express as the test server, connects to it, and displays an entry for it in the Interchange Server Express Component Management view.

The developer can then start the Integrated Test Environment perspective. This perspective can be launched by right-clicking the collaboration object that will be tested. Select the **Debug in Integrated Test Environment** option. Figure 6-51 on page 151 shows this step.
First of all, the user must create a test unit in the Integrated Test Environment. A test unit is a resource created in the workbench that defines a test and it is centered around a collaboration object. The test unit contains the configuration information for a test that will be performed. Figure 6-52 on page 152 shows the window where the user can create a test unit from a collaboration object.
The Select Collaboration box shows the collaboration object that will be tested. The CSM Project box shows the project that is the test base in the System Manager. In the New Integrated Test Environment Test Unit, the user defines the test unit. The ITE Project is selected by default.

At the end of the test unit configuration, the test environment displays the workbench where the test will run. Figure 6-53 on page 153 shows the test workbench.

---

**Figure 6-52  Creating integrated test environment**

<table>
<thead>
<tr>
<th>Selected Collaboration To Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration: InventorySync_SAL301R_to_SAP404R_and_Webservice</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New Integrated Test Environment Test Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Unit: InventorySync_SAL301R_to_SAP404R_and_Webservice</td>
</tr>
</tbody>
</table>

---

**Figure 6-52**
The integrated test environment interface, displayed above, has several major windows, which the developer can work with:

1. Integrated Test Environment Navigator: this view lists all projects and test units created. The user can launch the test via this window simply by selecting it. Figure 6-54 on page 154 shows this window.
2. Client Simulator: this view allows the user to emulate the clients in the integration solution (connector agents and access clients). This view, through a graphical interface, allows you to:

- Connect the clients to the server
- Create and send Business Object requests
- Reply with Business Object responses

**Note:** For more details, see:


3. Task Manager: this view allows the user to execute tasks that must be performed during the test, for instance, Start Server, Deployment, or Start Client Simulators. Figure 6-55 on page 155 shows this graphical interface. Using this interface, the user can set or disable a task, as a part of the test, just by clicking the box for each one.

The test can be run from this view by clicking the arrow (green) button.
4. Integrated Test Environment Console: this view shows informations about the execution and evolution of test tasks. Using this console, the user can check the performance of test through displayed messages and use the information to troubleshoot the test. Figure 6-56 shows this view.

---

Figure 6-55  Task manager window

Figure 6-56  Test environment console
5. Test Unit Editor: this shows a graphical representation of the integration system. It enables the developer to execute certain activities:
   - Components deployment, from the repository using the Test Unit view.
   - Component status management, whereby the user can see if the component is active or not.
   - Business Object data viewing, by clicking the current object.
   - Main collaboration with port information viewing.

Figure 6-57 shows the test unit editor where the integration solution components are represented.

![Test Unit Editor Diagram](image)

Figure 6-57 Test unit editor

6. Outline View: this view shows the components that are enrolled in the integration system and are displayed in the Test Unit editor. It is possible for the developer to verify the component’s properties by selecting the item in this view. The properties will be displayed in the Properties view. This view lets the user verify whether the interface dependencies have been resolved, and whether other required conditions have been satisfied prior to testing.

Figure 6-58 on page 157 shows this view.
Figure 6-58  Test unit outline
Adapters

This chapter provides an overview of the role of adapters in the WebSphere Business Integration Server Express architecture, including technology adapters and application-specific adapters. We describe the adapters that are provided as part of WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus and also provide details of the adapters available in the Adapter capacity packs.

Our examination of how to develop, implement, and deploy adapters is divided into:

- 7.1, “Adapter architecture” on page 160.
- 7.3, “How to implement an adapter” on page 172.
- 7.4, “Data handlers” on page 203.
- 7.6, “ODAs” on page 207.
7.1 Adapter architecture

WebSphere Business Integration Server Express adapters offer a communications interface into or out of the application for which they were developed.

As illustrated in Figure 7-1, they also act as an abstraction layer that gives customers the flexibility to protect themselves from technology change.

![Figure 7-1 Adapters in the WebSphere Business Integration reference architecture](image)

7.1.1 Architectural overview

The WebSphere Business Integration Server Express Plus product architecture has a hub and spoke topology. At the ends of the spokes are the information systems. These are the applications or technologies that own the data that must be accessed or changed. A hub is the broker that has the responsibility for receiving data from one or more systems and, based on the business rules or processes, performing tasks that may include propagating data to other systems. Figure 7-2 on page 161 displays a graphical representation of the architecture.
Figure 7-2  Hub and Spoke topology

Adapters interact with the spokes on behalf of the hub to extract data changes or push data updates to the applications. The adapters also hide application-specific details from the hub so that it sees those information systems as generic data consuming or producing entities, as shown in Figure 7-3.

Figure 7-3  Adapters in the Hub and Spoke topology

7.1.2 Adapter overview

The primary feature of an adapter is a connector that allows WebSphere Interchange Server Express to exchange Business Objects with applications. An adapter consists of two parts: an Adapter Agent and a connector controller. Both are based on the Adapter Framework, as illustrated in Figure 7-4 on page 162.
The Adapter Agent is an application-specific component that contains code tailored to a particular application or technology. The Adapter Agent may also be referred to as the connector agent. In this chapter, we will use the term Adapter Agent.

The connector controller communicates with the integration broker and provides services that are common to all adapters. The Adapter Framework provides a common platform for the Adapter Agent and the connector controller.

When an Enterprise Information System (EIS) needs to exchange data with another application, it invokes the Adapter Agent. The Adapter Agent verifies with the connector controller that it is subscribed to the current event. If it is, the Adapter Agent will create an Application Specific Business Object (ASBO) and send it to the connector controller. The ASBO will be mapped into a Generic Business Object (GBO) that the integration server can process. Upon completion the GBO will be passed back to the connector controller where it will be mapped to an ASBO and sent to the Adapter Agent. Finally, the Adapter Agent can map the Application Specific Business Object to an EIS action. This process is shown in Figure 7-5 on page 163.
In summary, as shown in Figure 7-5, the steps are:

1. Invoke Adapter Agent.
2. Verify that the event is subscribed to.
3. Create an ASBO.
4. Send ASBO to the connector controller.
5. Map ASBO to GBO.
6. Send GBO to the integration broker.
7. GBO is sent.
8. Map GBO to ASBO.
9. Send ASBO to Adapter Agent.
10. Map ASBO to EIS action.

### 7.1.3 Event notification

An event is when something happens in the EIS. It is assumed that all business integration scenarios will start with an event triggering an activity as shown in Figure 7-6.
Some examples of events include:

- Insertion, updating, or deletion of a row in a database table
- Arrival of an e-mail
- Submittal of an order to an order entry system

WebSphere Business Integration adapters use these events to trigger a business process. The process of conveying changed EIS data to the integration broker is called event notification. Most of today's EISs either do not provide an event notification mechanism for external systems (such as an adapter) or they provide only rudimentary support. To enable automated integration, event notification is an absolute necessity.

To an adapter, an EIS event is any operation that affects the data of an EIS entity associated with a Business Object definition. Some EISs can report events; other EISs might silently update their databases when something happens. Where an EIS provides a facility for events, the adapter's event detection mechanism generally leverages it to detect events.

If an EIS does not provide native support for events, the event notification mechanism can utilize the EIS database. In this case, a table is created for the adapters event store, and database triggers are installed on the EIS tables of interest to insert event records into the event table.

The event notification mechanism generally consists of the elements described in Table 7-1.

<table>
<thead>
<tr>
<th>Element</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>An event store located within the EIS</td>
<td>Database table</td>
</tr>
<tr>
<td>A means to populate the event store</td>
<td>Database trigger</td>
</tr>
<tr>
<td>Event detection mechanism</td>
<td>Polling</td>
</tr>
<tr>
<td>Event retrieval mechanism</td>
<td>Database query</td>
</tr>
</tbody>
</table>

### 7.1.4 Subscription handling

Adapters use subscription lists to avoid sending every event to all the connected EISs. A subscription list is a list of Business Objects to which collaborations have subscribed.
When collaborations start, they subscribe to the Business Objects that they are interested in by informing the connector controller of their interest. The connector controller stores this information in a subscription list that contains the name of the subscribing collaboration, the name Business Object and the Business Object verb. The subscription list is used by the connector controller to determine which collaborations have subscribed to this type of Business Object.

As part of the initialization, the connector controller instantiates a subscription manager. The subscription manager tracks all subscribe and unsubscribe messages. It also maintains a list of active Business Object subscriptions. Through the subscription manager, the Adapter Agent can query the connector controller to discover whether any collaborations are interested in a particular kind of Business Object.

### 7.1.5 ASBO instantiation

After detecting an event and verifying that it is subscribed to, the Adapter Agent retrieves data from the DataSource and populates the ASBO as demonstrated in Figure 7-8 on page 166.
The following describes the logic of how this is accomplished.

1. The Adapter Agent associates the application event with a Business Object definition.
   - The Adapter Agent uses the event text to associate the event with a Business Object definition.

2. The Adapter Agent sets the verb and key value attributes in the Business Object.
   - Once the object has been instantiated, the Adapter Agent uses the event text to set the verb (for example, Create, Update) and key values such as the unique event ID.

3. The Adapter Agent retrieves application data and populates the Business Object's attributes.
   - The method by which an Adapter Agent populates an ASBO depends on the current application or technology. Most adapters use a meta-data-driven approach. The adapter can handle Business Objects based on meta-data encoded in the Business Object definition rather than on instructions hard coded in the connector.

4. Once the ASBO is populated, it is sent to the connector controller.
   - The Adapter Agent sends the ASBO to the connector controller using the configured transport layer as either synchronous or asynchronous communications.

5. The event is archived.
   - Some connectors use event archives. An event archive can play a useful backup role. If a collaboration stops running, its subscriptions are cancelled. While the collaboration is inactive, the connector continues to retrieve the associated events but finds no active subscriptions, so it archives the events. When the collaboration restarts, you can examine the event archive and
move archived events back to the event notification mechanism for reprocessing.

### 7.1.6 Data mapping

At initialization, the connector controller downloads its Business Object definitions and configuration properties from the InterChange Server Express repository. Each Business Object definition has an associated map that can transfer data between the ASBO and the GBO, as shown in Figure 7-9.

![Figure 7-9  Business Object mapping](image)

When a connector controller receives an ASBO from the Adapter Agent, it does the following:

1. Reads the name of the associated map from the Business Object definition.
2. Uses map to transfer data to a GBO.

For more detailed information about mapping see Chapter 9, “Mapping” on page 247.

### 7.1.7 Transport layer

The transport layer handles the exchange of information between the Adapter Agent and the connector controller as shown in Figure 7-10.

![Figure 7-10  Transport layer](image)
Synchronous communication is used for:

- Startup messages between the connector controller and the client connector
- Administrative messages such as the state of the adapter.
- Collaboration of initiated requests

To send Business Objects between the Adapter Agent and connector controller, the transport layer can be configured to use either synchronous (Common Object Request Broker - CORBA) or asynchronous (Message-Oriented-Middleware - MOM) messaging.

7.2 Express adapters

WebSphere Business Integration adapters can be categorized as either application adapters or technology adapters.

Application adapters are designed to interface with existing EIS systems, such as Siebel and PeopleSoft, using a specific application programmable interface (API). Technology adapters are designed to interface with existing EIS systems through common technology such as JDBC or Web services.

7.2.1, “Technology adapters” on page 168 and 7.2.2, “Application adapters” on page 170 give a brief description of the technology and application adapters that are available with WebSphere Business Integration Server Express.

7.2.1 Technology adapters

The technology adapters that are available with WebSphere Business Integration Server Express are:

- E-mail adapter
  The e-mail adapter allows the integration broker to communicate with e-mail applications. The adapter exchanges Business Objects, files, and messages. For more information, see the following Web site:
  http://ibm.com/software/integration/wbiadapters/email/

- JDBC adapter
  The JDBC adapter can be used to connect the integration broker to any database that has a Java Database Connectivity (JDBC) 2.0 compliant database driver. The connector executes SQL statements or stored procedures to retrieve or change data in the database/application. For more information, see the following Web site:
- **JMS adapter**
  This adapter allows IBM WebSphere integration brokers to exchange Business Objects with applications that send or receive data in the form of Java Message Service (JMS) messages. Both point-to-point (PTP) messaging and publish-and-subscribe (Pub/Sub) messaging defined by the JMS standard are supported. For more information, see the following Web site:
  

- **WebSphere MQ adapter**
  The WebSphere MQ adapter allows the WebSphere integration broker to exchange Business Objects with applications that send or receive data in the form of WebSphere MQ messages. For more information, see the following Web site:
  
  http://ibm.com/software/integration/wbiadapters/mq/

- **XML adapter**
  The XML adapter enables integration brokers to exchange Business Objects with URLs by using HTTP and HTTPS protocols. The adapter supports XML version 1.0. For more information, see the following Web site:
  
  http://ibm.com/software/integration/wbiadapters/xml/

- **Web services adapter**
  The Web services adapter provides the functionality needed to exchange Business Object information in the body of a Simple Object Access Protocol (SOAP) message. The adapter is based on the SOAP 1.1 specification. For more information, see the following Web site:
  
  http://ibm.com/software/integration/wbiadapters/services/

- **COM adapter**
  The COM adapter supports exchange of Business Objects with applications or COM (Common Object Model) components, running on a COM server. The adapter also provides DCOM (Distributed Component Object Model) support. For more information, see the following Web site:
  

- **SWIFT adapter**
  The SWIFT adapter allows the integration broker to exchange Business Objects with applications that send or receive data in the form of Society for the Worldwide Interbank Financial Telecommunication (SWIFT) messages. For more information, see the following Web site:
  
  http://ibm.com/software/integration/wbiadapters/swift/
7.2.2 Application adapters

The application adapters that are available with WebSphere Business Integration Server Express are:

- **Microsoft Exchange adapter**
  
  The adapter for Microsoft Exchange Server enables the exchange of Business Objects between the integration broker and an Exchange Server. This connector is single threaded. As a result, events are handled one at a time, in the order in which they occur. For more information, see the following Web site:


- **Portal Infranet adapter**
  
  This adapter enables the exchange of Business Objects between the integrator broker and the Portal Infranet. The adapter uses a socket-based application program interface (API) to communicate with the Portal Infranet application. For more information, see the following Web site:


- **Lotus® Domino® adapter**
  
  The Lotus Domino adapter allows a Lotus Domino server to exchange information with an integration broker. The adapter supports Lotus Domino Server version R5 (5.0.3 and above) and R6 (6.0.2 and below). For more information, see the following Web site:


7.2.3 Capacity pack adapters

The Adapter capacity pack provide WebSphere Business Integration Server Express with an additional set of adapters. These adapters are:

- **i2 adapter**
  
  The i2 adapter integrates with i2 application modules through the i2 Common Integration Services (CIS) API. CIS API from i2 is an implementation of the JCA Common Client Interface. The adapter enables integration to any version
6.0.1 SDK CIS-enabled i2 application. For more information, see the following Web site:

http://ibm.com/software/integration/wbiadapters/i2/

▶ JD Edwards OneWorld adapter

This adapter allows the integration broker to exchange data between Business Objects and their corresponding OneWorld objects running on a OneWorld server. For more information, see the following Web site:

http://ibm.com/software/integration/wbiadapters/jdedwards/

▶ MetaSolv adapter

This adapter allows a MetaSolv application to exchange information with an integration broker. The adapter uses meta-objects, Business Object definitions, attribute descriptions, and values in a Business Object instance to generate MetaSolv API calls. For more information, see the following Web site:

http://ibm.com/software/integration/wbiadapters/metasolv/

▶ mySAP.com adapter

The mySAP.com adapter allows the integration broker to communicate with SAP applications. SAP's Remote Function Call (RFC) API allows external programs to call ABAP function modules within an SAP application. The adapter uses that API to communicate with an SAP application. For more information, see the following Web site:

http://ibm.com/software/integration/wbiadapters/mysap/

▶ Oracle applications adapter

This adapter allows the integration broker to exchange Business Objects with an Oracle application using a JDBC driver for Oracle database that follows the JDBC 2.0 or higher specification. The connector is meta-data-driven and uses application-specific information to build and execute SQL statements or stored procedures. For more information, see the following Web site:

http://ibm.com/software/integration/wbiadapters/oracleap/

▶ PeopleSoft adapter

This adapter enables the exchange of Business Objects between an integration broker and PeopleSoft. The connector communicates with PeopleSoft using the PeopleSoft Message Agent API and a set of message definitions for each Business Object. For more information, see the following Web site:

http://ibm.com/software/integration/wbiadapters/psoft/
- Siebel e-business applications adapter
  This adapter allows an integration broker to exchange Business Objects with a Siebel's e-business application. The adapter uses the Java APIs provided by the Siebel Java Data Bean to communicate with the Siebel Object Manager for data exchange. For more information, see the following Web site:
  http://ibm.com/software/integration/wbiadapters/siebelebus/

- Telcordia adapter
  This adapter allows the integration broker to exchange Business Objects with applications that send or receive data in the form of Telcordia messages. The adapter allows bidirectional data exchanges between the Telcordia Service Delivery module and the integration broker. For more information, see the following Web site:

- WebSphere Commerce adapter
  The adapter enables the exchange of Business Objects between an integration broker and a WebSphere Commerce application. The adapter is meta-data-driven and uses an MQ implementation of the JMS API to communicate with the commerce application. For more information, see the following Web site:
  http://ibm.com/software/integration/wbiadapters/wcscomm/

7.3 How to implement an adapter

This section describes how an adapter can be installed, configured, and deployed.

7.3.1 Installing and uninstalling an adapter

Some adapters may have specific installation procedures and properties but most will be the same. This section describes how the JDBC adapter can be installed. More information about adapter installation procedures can be found in the documentation for each adapter and at the WebSphere Business Integration information center located at this Web site:

http://publib.boulder.ibm.com/infocenter/wbihelp/

Installing an adapter on a local environment
The adapter is installed on the same computer as the integration broker.
When the setup programs runs, the launchpad program as shown in Figure 7-11 starts up. Using this program, perform the following steps:

1. On the launchpad, click **Install Product**.
2. Click **Next** on the welcome window as shown in Figure 7-12.

3. Read the agreement as shown in Figure 7-13 on page 174, accept the terms, and click **Next**.
4. Select the adapter to install as illustrated in Figure 7-14 and click Next.

5. Verify the selected features as described in Figure 7-15 and click Next.

6. Figure 7-16 on page 175 illustrates a successful installation. Click Next to continue with the Wizard.
7. Restart the computer when prompted.

**Uninstall the JDBC adapter**

WebSphere Business Integration Server Express provides an uninstall program that can be used to remove adapters as shown in Example 7-1.

**Example 7-1  Uninstall program**

Linux: `<ProductDir>_uninstWBIServerExpPlus\uninstaller.bin`

Windows: `<ProductDir>_uninstWBIServerExpPlus\uninstaller.exe`

From the language selection prompt, choose the desired language from the drop-down menu and click **OK**. From the Welcome window, click **Next**. The summary window lists the components that will be uninstalled and the product directory from which they will be removed. Read the information to verify that it is correct and click **Next**. After the uninstaller finishes successfully, click **Finish**.

**7.3.2 Setting up event notification**

This section describes how an event notification mechanism can be set up. The SAL301R application provides functionality to register properties for sale. We will make the event notification mechanism send an event every time a new property is registered. For more information about SAL301R, see *WebSphere Application Server - Express: A Development Example for New Developers*, SG24-6301.

We will use a JDBC adapter to connect to the SAL301R application and database triggers to implement the event notification mechanism. Each time a new row is added to the SAL301.prop table, an after insert trigger will add a row to the SAL301R.SAL301R_Events table. The JDBC adapter will be configured to poll this table for new events. This process is illustrated in Figure 7-17 on page 176.
The tables and triggers can be installed using the SAL301R_Events.ddl. Open a DB2 command window and type the command in Example 7-2 below.

**Example 7-2   Install tables and triggers using the sample script tile**

C:\SQLLIB\BIN>db2 -td@ -vf SAL301R-Events.ddl

The sample assumes that the ddl file is available in the bin directory.

**Creating the event notification tables manually**
The steps for creating the event notification tables manually are:

1. Open the DB2 command center as shown in Figure 7-18 on page 177 and connect to your database
2. Add the event table to the database.

   Copy and past the event table information in Example 7-3 into the Command Center.

**Example 7-3 Event table**

```sql
CREATE TABLE sal301r_events(
    event_id INTEGER NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY (START WITH 0 ,INCREMENT BY 1, NO CACHE ),
    connector_id VARCHAR(40),
    object_key VARCHAR(80) NOT NULL,
    object_name VARCHAR(40) NOT NULL,
    object_verb VARCHAR(40) NOT NULL,
    event_priority INTEGER NOT NULL,
    event_time TIMESTAMP,
    event_status INTEGER NOT NULL,
    event_comment VARCHAR(100));
```
3. Add the archive table to the database.
   
   Copy and paste the archive table information from Example 7-4 into the Command Center.

**Example 7-4  archive table**

```sql
CREATE TABLE sal301r_archive_events(
  event_id INTEGER NOT NULL PRIMARY KEY,
  connector_id VARCHAR(40),
  object_key VARCHAR(80) NOT NULL,
  object_name VARCHAR(40) NOT NULL,
  object_verb VARCHAR(40) NOT NULL,
  event_priority INTEGER NOT NULL,
  event_time TIMESTAMP,
  archive_time TIMESTAMP,
  event_status INTEGER NOT NULL,
  event_comment VARCHAR(100));
```

**Creating the trigger manually**

To create the trigger manually:

1. Open the trigger wizard in the DB2 Control Center as shown in Figure 7-19.

![Figure 7-19  Open trigger wizard](image)

2. Set the trigger to execute after insert as shown in Figure 7-20 on page 179.
3. Specify the trigger details with the dialog shown in Figure 7-21. Replace the text in the Triggered action field with the SQL statement illustrated in Example 7-5 on page 180.
Example 7-5  Event trigger content

```
BEGIN ATOMIC INSERT INTO sal301r_events (connector_id,oject_key,object_name, object_verb, event_priority,event_status, event_time, event_comment)
VALUES ('',RTRIM(CAST( CAST(n.PROP_ID AS CHAR(5)) AS VARCHAR(80))),'SAL301R_PROP','Create',0,0,CURRENT TIMESTAMP,'PROPERTY CREATE');
END
```

Testing the event notification trigger

To test the trigger by creating a new property, log on to the SAL301R application as a property sales agent. For example: enter ilaria/password as shown in Figure 7-22, and then click the properties link. Then, using the input fields on the form, add a new property.

![User Login](image)

Figure 7-22  Log on the SAL301R application as a property sales agent

If the SAL301R application is not installed, use the DB2 Command Center and execute the SQL statement shown in Example 7-6.

Example 7-6  Add a row to the SAL301R.Prop table

```
INSERT INTO PROP (USR_ID, PROP_TYPE_ID, PROP_STATUS_ID, PROP_ADDR_CNTRY_CD, PROP_ADDR_POST_CD, PROP_ADDR_RGN, PROP_ADDR_MUNI, PROP_ADDR_STRT_NAM, PROP_ADDR_BLDG_NUM, PROP_ADDR_UNIT_NUM, PROP_PRICE, PROP_CRD_ON) VALUES (1, 2, 1, 'US', '92675', 'CA','San Juan Capistrano', '33762 VALLE ROAD', '11', '12', 205000, current timestamp);
```

Verify that an event has been inserted into the event table. From the command center run the SQL statement: select * from sal301r_events. The result set should contain one row.
7.3.3 Adapter configuration

Adapters will have some common and some unique configuration settings. For configuration information specific to your adapter, please refer to the adapter documentation. This section describes how the JDBC adapter can be configured with the event notification mechanism we created in “Creating the trigger manually” on page 178.

Creating a new adapter instance
To create a new adapter instance:

1. Open the System Manager and expand the Integration Component Libraries folder. If the SAB404RRealtyICL folder is available, open it; otherwise, create it by right-clicking Integration Component Libraries.

2. Right-click SAB404RRealtyICL/Connectors as shown in Figure 7-23.

![Create New Connector](image)

Figure 7-23 Create new connector (adapter) instance

3. Select JDBCAdapterTemplate as shown in Figure 7-24 on page 182 and name it SAL301RConnector.
Figure 7-24 Select JDBC as adapter type
Configuring the standard adapter properties
Adapters do not have to use all the standard properties. Figure 7-25 shows the values we will be using.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminInQueue</td>
<td>SAL301RCONNECTOR/ADMININGQUEUE</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>SAL301RCONNECTOR/ADMINOUTQUEUE</td>
</tr>
<tr>
<td>AgentTraceLevel</td>
<td>5</td>
</tr>
<tr>
<td>ApplicationName</td>
<td>SAL301RConnector</td>
</tr>
<tr>
<td>BrokerType</td>
<td>ICS</td>
</tr>
<tr>
<td>CharacterEncoding</td>
<td>ascii7</td>
</tr>
<tr>
<td>ConcurrentEventTriggeredFlows</td>
<td>1</td>
</tr>
<tr>
<td>ContainerManagedEvents</td>
<td></td>
</tr>
<tr>
<td>ControllerStoreAndForwardMode</td>
<td>true</td>
</tr>
<tr>
<td>ControllerTraceLevel</td>
<td>5</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>SAL301RCONNECTOR/DELIVERYQUEUE</td>
</tr>
<tr>
<td>DeliveryTransport</td>
<td>JMS</td>
</tr>
<tr>
<td>DuplicateEventElimination</td>
<td>false</td>
</tr>
<tr>
<td>EnableDidForFlowMonitoring</td>
<td>false</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>SAL301RCONNECTOR/FAULTQUEUE</td>
</tr>
<tr>
<td>jms.FactoryClassName</td>
<td>CzCommon.Messaging.jms.IBMMDSeriesFactory</td>
</tr>
<tr>
<td>jms.MessageBrokerName</td>
<td>WebSphereICS.queue.manager</td>
</tr>
<tr>
<td>jms.NumConcurrentRequests</td>
<td>10</td>
</tr>
<tr>
<td>jms.Password</td>
<td>**********</td>
</tr>
<tr>
<td>jms.UserName</td>
<td></td>
</tr>
<tr>
<td>JvmMaxHeapSize</td>
<td>128m</td>
</tr>
<tr>
<td>JvmMaxNativeStackSize</td>
<td>128k</td>
</tr>
<tr>
<td>JvmMinHeapSize</td>
<td>1m</td>
</tr>
<tr>
<td>Locale</td>
<td>en_US</td>
</tr>
<tr>
<td>LogAtInterchangeEnd</td>
<td>false</td>
</tr>
<tr>
<td>MaxEventCapacity</td>
<td>2147483647</td>
</tr>
<tr>
<td>MessageFileName</td>
<td>SAL301RConnector.txt</td>
</tr>
<tr>
<td>OADAutoOfRestartAgent</td>
<td>false</td>
</tr>
<tr>
<td>OADMaxNumRetry</td>
<td>1000</td>
</tr>
<tr>
<td>OADRetryTimeInterval</td>
<td>10</td>
</tr>
<tr>
<td>PollEndTime</td>
<td>HH:MM</td>
</tr>
<tr>
<td>PollFrequency</td>
<td>10000</td>
</tr>
<tr>
<td>PollStartTime</td>
<td>HH:MM</td>
</tr>
<tr>
<td>RepositoryDirectory</td>
<td>REMOTE</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>SAL301RCONNECTOR/REQUESTQUEUE</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>SAL301RCONNECTOR/RESPONSEQUEUE</td>
</tr>
<tr>
<td>RestartRetryCount</td>
<td>3</td>
</tr>
<tr>
<td>RestartRetryInterval</td>
<td>1</td>
</tr>
<tr>
<td>SynchronousRequestQueue</td>
<td>SAL301RCONNECTOR/SYNCHRONOUSREQUEST</td>
</tr>
<tr>
<td>SynchronousRequestTimeout</td>
<td>0</td>
</tr>
<tr>
<td>SynchronousResponseQueue</td>
<td>SAL301RCONNECTOR/SYNCHRONOUSRESPONSE</td>
</tr>
<tr>
<td>WfFormat</td>
<td>CwBO</td>
</tr>
</tbody>
</table>

Figure 7-25  Standard properties used by the JDBC adapter
In our sample, the properties that have been changed are listed in Table 7-2.

**Table 7-2  Changed standard properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>New Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentTraceLevel</td>
<td>5</td>
</tr>
<tr>
<td>ControllerTraceLevel</td>
<td>5</td>
</tr>
<tr>
<td>MessageFileName</td>
<td>SAL301RConnector.txt (Copy of &lt;ProductDir&gt;\connectors\messages\JDBCConnector.txt)</td>
</tr>
<tr>
<td>DeliveryTransport</td>
<td>JMS</td>
</tr>
<tr>
<td>jms.MessageBrokerName</td>
<td>WebSphereICS.queue.manager</td>
</tr>
<tr>
<td>AdminInQueue</td>
<td>SAL301RCONNECTOR/ADMININQUEUE</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>SAL301RCONNECTOR/ADMINOUTQUEUE</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>SAL301RCONNECTOR/DELIVERYQUEUE</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>SAL301RCONNECTOR/FAULTQUEUE</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>SAL301RCONNECTOR/REQUESTQUEUE</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>SAL301RCONNECTOR/RESPONSEQUEUE</td>
</tr>
<tr>
<td>SynchronousRequestQ</td>
<td>SAL301RCONNECTOR/SYNCHRONOUSREQUESTQUEUE</td>
</tr>
<tr>
<td>SynchronousResponseQ</td>
<td>SAL301RCONNECTOR/SYNCHRONOUSRESPONSEQUEUE</td>
</tr>
</tbody>
</table>
We have configured the transport layer to use asynchronous communication when sending Business Objects between an Adapter Agent and a Connector Controller as shown in Figure 7-26.

![Figure 7-26 Asynchronous messaging using jms](image)

**Note:** The WebSphere MQ Express queues have not been created yet. Please refer to Appendix A, “Deploying the redbook sample code” on page 603 for information about how to create and install the MQ queues.

For more information about standard properties, go to the WebSphere Business Integration information center and refer to Appendix A in the adapter documentation:

Configuring the adapter-specific properties

Figure 7-27 shows the configurations for adapter-specific properties and their default values.

![Configuration Table]

**Figure 7-27  Adapter specific properties with default values**
In our sample, the properties have been changed as shown in Table 7-3 on page 187.

Table 7-3  Changed adapter specific properties

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>New value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ApplicationPassword</td>
<td>smbP4$$word</td>
</tr>
<tr>
<td>3</td>
<td>ArchiveTableName</td>
<td>SAL301R_ARCHIVE_EVENTS</td>
</tr>
<tr>
<td>9</td>
<td>Database URL</td>
<td>jdbc:db2:SAL301R</td>
</tr>
<tr>
<td>13</td>
<td>PingQuery</td>
<td>select 1 from prop</td>
</tr>
<tr>
<td>28</td>
<td>EventTableName</td>
<td>SAL301R_Events</td>
</tr>
<tr>
<td>29</td>
<td>JDBCDriverClass</td>
<td>COM.ibm.db2.jdbc.app.DB2Driver</td>
</tr>
<tr>
<td>30</td>
<td>ApplicationUserName</td>
<td>db2admin</td>
</tr>
</tbody>
</table>

Notice the value on the EventTableName property. It is the name of the database table we created when we implemented the event notification mechanism, as shown in Figure 7-17 on page 176.

**Saving the adapter configuration**

To save the adapter instance to the SAB404RRealtyICL project, use the Save as dialog window shown in Figure 7-28 to select SAB404RRealtyICL.
7.3.4 Creating Business Objects

Before we can continue to configure the adapter, we must create the ASBOs and the GBOs we want the adapter to use as illustrated in Figure 7-29.

![Table, Business Object mapping](image)

**Creating the ASBOs**

We will use the JDBC Object Discovery Agent (ODA) to create the initial version of the ASBOs. For more detailed information about ODAs refer to 7.6, “ODAs” on page 207.

To create the ASBOs:

1. Edit the `<ProductDir>/ODA/JDBC/start_JDBCODA.bat` file to let the runtime component know what database driver to use.

2. Add the DB2 driver to the end of DRIVERPATH variable as shown in Example 7-7, where we have added the path after the ellipses that represent the existing value of the environment variable.

   **Example 7-7  Modify DRIVERPATH variable**
   
   `DRIVERPATH=...C:\SQLLIB\java\db2java.zip`

3. Modify the DRIVERLIB variable to point to the DB2 driver dll as shown in Example 7-8.

   **Example 7-8  Modify DRIVERLIB variable**
   
   `set DRIVERLIB=C:\SQLLIB\BIN`

4. Start the ODA runtime component by double-clicking `start_JDBCODA.bat` in Windows Explorer.
5. Once the ODA runtime component is running, open the Business Object Designer Express tool by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Relationship Designer Express.

6. When the Relationship Designer Express window appears, select File from the menu, then click New Using ODA. When the Select Agent window appears, it will ask you to select an ODA as shown in Figure 7-30.

![Business Object Wizard - Step 1 of 6 - Select Agent]

Figure 7-30 Find the JDBCODA agent

7. Click Next and enter the values shown in Figure 7-31 on page 190 to connect to the database.
8. In the Select Source window, click **PROP**, **PROP_STS** and **PROP_TYPE** as shown in Figure 7-32 and click **Next** twice when the dialog appears.

9. When the window shown in Figure 7-33 is displayed, change the Add Stored Procedure property from Yes to No by using the drop-down list provided. Click **OK**.

10. The last window will ask you to select a project where the ASBO definition will be stored. Select **SAB404RRealtyICL**.

### 7.3.5 Modifying the ASBOs

A property instance in the SAL301R database has information spread over multiple database tables (**PROP**, **PROP_STS**, **PROP_TYPE**). For the representation of a property to be complete, the SAL301R_PROP ASBOs must contain information from all three database tables. We can accomplish this by
modifying SAL301R_PROP and allowing it to treat SAL301R_PROP_STS and SAL301R_PROP_TYPE as child objects as illustrated in Figure 7-34.

Figure 7-34  Application Specific Business Object model

Figure 7-35 illustrates the SAL303R_PROP object definition.
For detailed information about the definition of the ASBOs, refer to 13.6.1, “Business Objects” on page 548 and the provided source files:

- SAL301R_PROP.xsd
- SAL301R_PROP_STS.xsd
- SAL301R_PROP_TYPE.xsd

### 7.3.6 Creating the GBOs

The Business Objects Property and Address shown in Figure 7-36 represent the generic definition of a property in our sample application.

![Property - Address relationship](image)

**Figure 7-36  Property - Address relationship**

For details about the Property and Address definitions, refer to 13.6.1, “Business Objects” on page 548 and the provided source files:

- Property.xsd
- Address.xsd

### 7.3.7 Setting up supported Business Objects

The Connector Configurator Express tool lets us specify which objects the current adapter instance can handle. Figure 7-37 on page 193 shows that the adapter instance is configured to handle the Property GBO and the SAL301R_PROP ASBO. (The Employee and SAL301R_USR objects are part of the Employee synchronization scenario.)
Chapter 7. Adapters

7.3.8 Setting up supported maps

The Agent Support check box has been selected for the ASBOs. Selecting the Agent Support option tells the adapter that the Business Object must be made available to the Adapter Agent.

**Note:** The child objects are implicitly registered.

The Associated Maps page lets us specify which map we want the connector controller to use when it receives a Business Object. Figure 7-38 on page 194 is a high-level overview of the interaction between a connector controller, a map, and a collaboration object.

**Note:** The Associated Map drop-down box will be empty if no maps have been created.
Figure 7-38  Mapping SAL301R_PROP to property

Figure 7-39 shows that the SAL301R adapter is configured to use the SAL301R_PROP_Property map when it receives an ASBO of type SAL301R_PROP.

Figure 7-39 SAL301R adapter. Supported maps

The Explicit Binding option should be used when the Associated Map field contains more then one option.

Chapter 13, “Sample development” on page 469 has detailed instructions on how to create the map “SAL301R_PROP_Property“.

7.3.9 Adapter Agent runtime configuration

The Adapter Agent runs as a standalone program as shown in Figure 7-40 on page 195 and must be started with its start-up script.
Creating a copy of the start-up script for each adapter instance is optional; however, if you choose to copy the script, take the following steps:

1. Create a runtime directory for the adapter instance.

   Copy the original runtime directory and give the directory the same name as your adapter. The SAL301RConnector is a JDBC adapter, so we copied the JDBC runtime directory to SAL301R as shown in Example 7-9.

   **Example 7-9  Runtime directory for the SAL301R adapter**

   C:\IBM\WebSphereServer\connectors\JDBC
   C:\IBM\WebSphereServer\connectors\SAL301R

2. Rename existing files.

   Rename the JAR file that contains the adapter class files. Use the name standard: CW<adapterName>.jar as shown in Example 7-10.

   **Example 7-10  Rename JAR file**

   Before: CWJDBC.jar
   After: CW01R.jar

Creating a shortcut for each start-up script is required. The shortcut should specify which server to use and the configuration file for the adapter instance. In our example, WebSphereICS is the server name and the cfg file contains the adapter configuration.

To create the adapter configuration file, save the adapter instance to a file using the Save As dialog shown in Figure 7-41 on page 196.
Create the windows shortcut that will start the Adapter Agent using the command shown in Example 7-11 to update the target field of the short cut. The final dialog should look like the one shown in Figure 7-42 on page 197.

**Example 7-11**  Shortcut details

```
SAL301RJDBC:
C:\IBM\WebSphereServer\connectors\SAL301R\start_JDBC.bat SAL301R WebSphereICS -cconnectors\SAL301RJ\SAL301RCConnector.cfg
```
Chapter 7. Adapters

197

Figure 7-42 Adapter start-up script shortcut

All the libraries that the adapter requires must be available in the adapter’s classpath. The SAL301 start-up script has to be modified to ensure that it can find the DB2 JDBC driver that we specified in “Configuring the adapter-specific properties” on page 186. Our current sample assumes that each adapter instance has its own copy of the start-up script.

To modify the SAL301 start-up script:

1. Open the start-up script for the SAL301R adapter.
2. Change the JDBCDRIVERPATH variable in the start_JDBC.bat file. See Example 7-12.

Example 7-12 Specify which JDBC driver to use

| Before: REM JDBCDRIVERPATH= |
| After: SET JDBCDRIVERPATH=C:\SQLLIB\java\db2java.zip |
3. In addition to changing the classpath, you must tell the adapter where it can find the DB2 executables. To do this, add the bin directory to the java.library.path variable as shown in Example 7-13.

Example 7-13  Add the path to the DB2 executables to the java.library.path variable

Before:
-Djava.library.path="%CROSSWORLDS%\bin;%CONNDIR%;%MQ_LIB%;%JRE_EXT_DIRS% %ORB_PROPERTY%"

After:
-Djava.library.path="%CROSSWORLDS%\bin;%CONNDIR%;%MQ_LIB%;%JRE_EXT_DIRS%;c:\sqlib\bin %ORB_PROPERTY%"

7.3.10 Deploying the adapter

The adapter instance we have been working with is a member of the SAB404RRealtyICL component library. Before the adapter instance can be deployed, it should be made a member of a user project as follows:

1. Create the SAB404RRealtyICS project by right-clicking InterChange Server Projects as shown in Figure 7-43. Enter SAB404RRealtyICS as the project name and click Finish.

2. Make the SAL301R adapter instance a part of the SAB404RRealtyICS project. Drag the SAB404RRealtyICL/Connectors/SAL301RConnector instance to the SAB404RRealtyICS/Connectors folder. (We only have to perform this step once.) The final result is shown in Figure 7-44 on page 199.
3. Save the configuration to the project and file as shown in Figure 7-45.

**Note:** All deployable artifacts should be a part of the user project. Add all the Business Objects and maps that you need to the SAB404RRealtyICS before continuing.
4. Deploy the adapter.

Drag the configuration instance of the adapter to the runtime instance. The runtime instance must be stopped before you can redeploy the adapter. The stopped connector is illustrated in Figure 7-46.

![WebSphere Business Integration System Manager](image)

**Figure 7-46   Deploy adapter using drag and drop**

**Note:** Remember that the Adapter Agent start-up script requires the adapter configuration file as input parameter. When you save the adapter to the file, use the same name and location as those illustrated in Figure 7-42 on page 197.

**Note:** All dependent artifacts must be available in the server before the adapter can be deployed. Make sure all Business Objects and maps specified in the adapter are deployed before attempting to deploy the adapter instance.
5. Reboot the InterChange server.

Once the adapter is deployed reboot the InterChange server by right-clicking the WebSphereICS instance and selecting **Shutdown → Gracefully** as shown in Figure 7-47. Start the InterChange server again once the shutdown process has completed.

![Figure 7-47 Stop the InterChangeServer](image)

6. Start the connector controller.

Verify that the connector controller is running on the InterChange server. If it is not running, start it by right-clicking the connector instance and clicking **Start SAL301RConnector** as shown in Figure 7-48 on page 202.
7. Start the Adapter Agent.

Use the shortcut that we created (Figure 7-42 on page 197) to start the Adapter Agent. Given that the trace level has been set to 5, the Adapter Agent command window should display a message for each time it polls the event table as shown in Example 7-14.

**Example 7-14  Event polling**

```
[Type: Warning] [MsgID: 33007] [Mesg: Entering pollForEvents] ]
[Type: Warning] [MsgID: 33024] [Mesg: No events to Process..!] ]
```

**Note:** No events will be retrieved from the event table until a collaboration has subscribed to the events being created.

For information about how to test the adapter, see Chapter 13, “Sample development” on page 469.

### 7.3.11 How to subscribe to events

The subscription list is controlled by the connector controller and the Interchange Server. When a collaboration object attaches an adapter to one of its ports as shown in Figure 7-49 on page 203, the subscription list for that adapter is updated.
Figure 7-49  **How collaboration objects subscribe to events**

Figure 7-50 shows a collaboration object that has attached the SAL301R adapter to the From port.

Figure 7-50  **Adapter attached to From port**

The collaboration object is configured to work with a GBO definition called Property. Our adapter must be able to handle the GBOs supported by the collaboration object.

For more information about collaborations, see Chapter 10, “Collaborations” on page 309.

### 7.4 Data handlers

In the context of an adapter, a data handler is a sub-component that is responsible for converting Business Objects to and from a specific serialized data format, as shown in Figure 7-51 on page 204.
Like an adapter, a data handler is meta-data-driven. Meta-data encoded in the Business Object definition allows a data handler to support new or modified Business Objects without changing the data handler code.

### 7.4.1 Overview

The data handler framework consists of an abstract class and a meta-data object. The abstract class specifies the methods that the data handler implementations must support. The meta-data object is used to tell the adapter which data handler implementation to use. The meta-data object also contains configuration information for the specific data handler implementation. This is illustrated in Figure 7-52.
DataHandler class
The com.crossworlds.DataHandlers.DataHandler is the base class for all data handlers. This class contains three types of methods:

- Abstract methods that must be implemented by the extending class
- Public methods that have a provided implementation
- Static methods that are called by the adapter

For more details about the DataHandler class, go to:
http://publib.boulder.ibm.com/infocenter/wbihelp/

Meta-data object
The data-handler meta-object is a hierarchical Business Object that can contain any number of child objects. The top-level meta-object contains MIME-type information and the child meta-objects contain configuration information for a specific data handler.

Note: A data handler is not required to use meta-objects to hold configuration information. However, all IBM-delivered data handlers are designed to use meta-objects for their configuration information.

7.4.2 Available data handlers
WebSphere Business Integration Server Express Plus includes the data handlers listed in Table 7-4.

Table 7-4  Data handlers

<table>
<thead>
<tr>
<th>Data handler</th>
<th>MIME type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FixedWidth Data Handler</td>
<td>text/fixedWidth</td>
</tr>
<tr>
<td>Request-Response</td>
<td>text/requestresponse</td>
</tr>
<tr>
<td>Delimited</td>
<td>text/delimited</td>
</tr>
<tr>
<td>NameValue</td>
<td>text/namevalue</td>
</tr>
<tr>
<td>Binary Host</td>
<td>N/A</td>
</tr>
<tr>
<td>XML</td>
<td>text/xml</td>
</tr>
<tr>
<td>EDI</td>
<td>edi</td>
</tr>
</tbody>
</table>

The data handler guide in the WebSphere Information Center has one chapter for each of the handlers listed in Table 7-4:
http://publib.boulder.ibm.com/infocenter/wbihelp/
7.4.3 Configuring a data handler

By default, adapters use the MO_DataHandler_Default provided by IBM as their top-level meta-objects. The MO_DataHandler_Default object is configured to support the MIME types of all the IBM-delivered data handlers.

If you want your adapter to support other MIME types, you must ensure that the types exist as attributes in the MO_DataHandler_Default object. In the sample shown in Figure 7-53, we have added the MIME type soap_xml. The Type column specifies the child meta-object that contains information needed to initialize the data handler. For more details about child meta-object configuration, refer to the documentation for the specific data handler.

![Figure 7-53  MIME type and associated child meta-object](image)

7.5 Adapter Development Kit

The Adapter Development Kit (ADK) is an integrated development toolkit that provides the framework for developing custom adapters. It provides programming interfaces in both Java and C++, with the following key development capabilities:

- Configuring the application model view to the integration broker
- Mapping the application model to a standardized meta-data exchange interface with the integration broker
- Extracting the network communication protocol with the integration broker while exchanging runtime data
- Performing the above runtime data exchange bidirectionally
- Creating extensible meta-data-driven adapters without the need for recoding
- Providing templates to develop adapters that are meta-data driven

The ADK provides files to assist with adapter development and samples of adapter components, including an ODA, a connector, and a data handler.

The ADK is part of the WebSphere Business Integration Adapters product and it requires its own separate Installer. Therefore, you must have access to the WebSphere Business Integration Adapters product and install the ADK. The ADK also includes the Connector Development Kit (CDK), which provides components for use in the development of a connector.
7.6 ODAs

An ODA is a development tool that can be used to generate Business Object definitions for ASBOs. For information about how to use an ODA, see Chapter 8, “Business Objects” on page 209.

7.6.1 ODA architectural overview

The Business Object Designer tool provides a graphical user interface in the form of a wizard (Business Object Wizard) to interact with the ODA. The ODA has two main components: a runtime component, which provides support for the Business Object Wizard, and the ODA implementation, which contains the functionality for discovering source nodes in the DataSource. This architecture is illustrated in Figure 7-54.

![Object Discovery Agent Architecture Diagram]

Note: The ADK is available only on Windows systems.
7.6.2 ODA Development Kit

If you are developing a custom adapter and you want to use an ODA to create Business Object definitions, you can use the Object Discovery Agent Development Kit (ODK). The development kit contains the following elements:

- **ODK API**
  The API includes framework classes that must be extended and utility classes that will facilitate development. For more information, go to Developing and Object Discovery Agent in the WebSphere Business Integration infocenter: [http://publib.boulder.ibm.com/infocenter/wbihelp/](http://publib.boulder.ibm.com/infocenter/wbihelp/)

- **ODA Runtime**
  This is a set of Java classes that the ODA runtime uses to handle communication between the ODA and Business Object Wizard.

- **ODA samples**
  In WebSphere Business Integration Server Express Plus, the samples will be available in the `<ProductDir>/DevelopmentKits/Odk` directory.
Business Objects

This chapter provides an overview of the role of Business Objects and Business Object definitions in WebSphere Business Integration Server Express. We describe the Business Objects that are provided as part of WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus.

We examine how to develop, implement, and deploy Business Objects.

- 8.4, “Developing with Object Discovery Agents” on page 228.
- 8.7, “JDBC application-specific information” on page 244.
8.1 Introduction to Business Objects

In WebSphere Business Integration, data and processing instructions are packaged and transferred between systems as Business Objects. Business Objects that are transferred between an application and the Interchange Server are called Application Specific Business Objects (ASBO), while Business Objects that are processed within the Interchange Server are called Generic Business Objects (GBO).

This chapter describes the key concepts of Business Objects:

► Business Object definitions

A Business Object definition specifies the data and processing instructions the application contains, for example, Customer, and create, retrieve, update and delete.

WebSphere Business Integration Server Express provides two ways to develop Business Object definitions:

– Manually using the Business Object Designer Express

– Automatically using an API (Object Discovery Agent Development Kit) that lets you create Object Discovery Agents (ODAs)

► Business Objects

A Business Object is an instance of a definition containing actual data and a single processing instruction, for example, John Doe and update.

A Business Object can either be application-specific or generic. Application Specific Business Objects are unique to one application and are mapped to Generic Business Objects so they can be used in processes in the Interchange Server.

Figure 8-1 on page 211 highlights the Business Objects in the reference architecture. They are shown as Generic Business Objects (GBO) and Application Specific Business Objects (ASBO).
8.1.1 Business Object definitions

Business Object definitions are templates that provide adapters with the information they need to manage data in the underlying application schema. Business Object definitions are stored in the WebSphere Business Integration Server Express repository. Business Object definitions are either application-specific or generic, and are created using the Business Object Designer Express. Figure 8-2 on page 212 depicts the difference between an SAL301R-specific user Application Specific Business Object and a Generic Business Object for user.
A Business Object definition consists of:

- A header containing the name and version of the Business Object definition
- Attributes representing data entities, for example CustomerName
- Verbs describing processing instructions, for example create or update

At each level, you can enter application-specific information, which provides meta-data, descriptive data, to the component that processes the Business Object. This information provides the application with instructions to the connector or data handler on how to process the Business Object. The purpose of application-specific information in Business Objects is to allow customization of the application data without modifying the adapter agent code.

A Business Object instance is a Business Object containing actual data and a single processing instruction, for example, John Doe and update.

Figure 8-3 on page 213 depicts the elements of a Business Object definition with example attributes using the SAL301R USR table.
Business Object headers

The Business Object header consists of a Business Object name and, optionally, application-specific information. The Business Object name identifies the Business Object within the WebSphere Business Integration Server Express and must be provided when the Business Object definition is created.

It is common practice to provide a descriptive name to the Generic Business Object, for example User, while an Application Specific Business Object has an application prefix following the name, for example SAL301R_USR.

Application-specific information about the Business Object header is provided when processing instructions are relevant for the entire Business Object hierarchy, for example, the transaction scope or the name of table or form in which the record belongs. For a Business Object specific for JDBC, you must specify the table name the Business Object represents (as shown in Figure 8-3).

For further information about application-specific information, refer to the WebSphere Business Integration Server Express Online Help.
Business Object attributes

Business Object definitions contain attributes representing groups of data, typically a column in a table. Each attribute is defined with attribute properties, such as a unique name and a type. When you create a Business Object instance, the attributes are given attribute values.

The attribute properties provide the components acting on the Business Object with information on the type, size and default value of the attribute.

The following attribute properties are available:

- **Name**
  
  An attribute must have a unique name that describes the data that the attribute contains.
  
  Note that the name must use only characters defined in the *en_US* code set.

- **Type**
  
  Each attribute has a data type that can either be a simple attribute (a Boolean, Integer, Float, Double, String, Date, LongText) or a complex attribute (a child Business Object, or an array of child Business Objects).

  An example of a child Business Object is a user containing an address. An example of an array of child objects is an array of order items contained within an order.

  Attributes representing a child Business Object must specify a cardinality of 1 (one), while attributes representing an array of Business Objects must specify a cardinality of *n*.

  **Note:** All attributes representing child Business Objects have a *ContainedObjectVersion* property which specifies the version number of the child object's Business Object definition, and a *RelationShip* property which specifies the value *Containment*.

- **Key**

  A Business Object must contain at least one key attribute that uniquely identifies the Business Object. A key attribute is often referred to as a primary key.

  **Note:** If a child Business Object is defined as a key attribute, the key is a concatenation of the keys in the child Business Object.

  If an array of child Business Objects is defined as a key attribute, the key is a concatenation of the keys in the first element of the array.
Foreign Key

A Business Object can reference other Business Objects through foreign key attributes. This is used in Application Specific Business Objects to specify that a foreign key attribute contains the primary key of another Business Object.

If the column that the attribute represents is not a foreign key in the table, you can still specify the attribute as a foreign key. This will force the connector to check if the foreign entity exists in the database and only create the relationship if the record exists.

Required Attribute

If an attribute is set to required, the Business Object will not be processed unless the attribute contains a value.

Cardinality

An attribute representing a child Business Object must have a single cardinality (cardinality 1), while an attribute representing an array of child Business Objects must have a multiple cardinality (cardinality \( n \)).

Simple attributes (String, Integer and so on) do not need to have a cardinality specified because a single cardinality is implied.

Note: If a complex attribute with a single cardinality is set to required, a child Business Object must exist. A multiple cardinality indicates that zero to many child Business Objects can exist.

Maximum Length

This property indicates the number of bytes a string-type attribute can contain. The rule is not enforced by the WebSphere Business Integration Server Express, but some connectors or data handlers might use it.

Note: The Maximum Length property is very important when you use a FixedWidth data handler. The FixedWidth data handler converts Business Objects to fixed-width strings and streams, and vice versa.

Default Value

You can specify a default value the attribute will be populated with if it has no value at run time.

This property only applies to Application Specific Business Objects, and requires that the UseDefaults connector configuration property is set to true.
► Application-specific information

An attribute can have application-specific information associated with it. This attribute can be a string of up to 100 characters. This information is used whenever processing instructions are relevant for that attribute, for example, the equivalent field in a form or a column in a table.

► Comments

The comments property should be used to provide documentation about the attribute, such as design decisions.

► ObjectEventId

The ObjectEventId attribute is used by the WebSphere Business Integration Server Express to store a value that uniquely identifies and tracks event flows in the system. The connector framework generates these unique values for each Business Object and in each child Business Object.

**Important:** The ObjectEventId attribute is required, and *must* be the last attribute in the Business Object definition.

Do *not* map the ObjectEventId attribute or have a connector or data handler populate it. The business integration system handles the value of this attribute.

Figure 8-4 on page 217 depicts the Attributes tab of the Business Object Designer Express, and some sample data using the CustomerAddress Business Object in the WebSphere Business Integration Server Express library.
Business Object verbs

A Business Object definition contains a list of verbs or processing instructions the Business Object supports. When a Business Object is created at run time, it contains a single active verb which describes the operation to perform on the Business Object.

The most common verbs are:

- **Create**, which creates a new entity in the application
- **Retrieve**, which retrieves a Business Object based on a unique identifier
- **Update**, which updates one or more fields in the entity
- **Delete**, which deletes an entity from the application

You can also create custom verbs. For example, you can create a verb to check if an entity exists or to retrieve a Business Object based on non-key values.

And example of application-specific information provided with a verb is the name of the function to call in the application API.
8.1.2 Business Objects

Business Objects are instances of Business Object definitions created at runtime. A Business Object contains the actual data and a processing instruction, and is passed between systems.

There are two types of Business Objects:

- **Application Specific Business Object**
  An Application Specific Business Object (ASBO) reflects the entity structure of a specific application and is unique to that application.

- **Generic Business Object**
  A Generic Business Object is a superset of the Application Specific Business Objects. Collaborations act on Generic Business Objects.

Business Objects contain the following information:

- **Attributes containing attribute properties.** One or more of the attributes are key attributes (primary keys) that uniquely identify the Business Object. The ObjectEventId attribute is a unique id used by the Interchange Server to track the Business Object in the system.

- **An active verb defining the processing instruction on the Business Object.** This must be a verb supported by the Business Object definition.

Attributes, attribute properties and verbs are described in more detail in 8.1.1, “Business Object definitions” on page 211.

Figure 8-5 on page 219 shows the MyApp_User Business Object definition and an example of a Business Object instance.
Business Object structure

Business Objects can be either flat or hierarchical. The following sections describe the difference between these structures and an example Business Object definition for each.

**Flat Business Objects**

The simplest form of a Business Object is a *flat* structure. A flat Business Object contains only simple attributes such as String, Integer, Long and so forth. Simple attributes have a single cardinality.

The MyApp_User Business Object in Figure 8-5 is an example of a flat Business Object.

**Hierarchical Business Objects**

Hierarchical Business Objects contain references to one or more related Business Objects (children), encapsulating not only the individual entity but also the relationship between the entities. In turn, child Business Objects can contain references to other child Business Objects, creating a hierarchy of Business Objects.

Figure 8-6 on page 220 shows an example of a hierarchy of Business Objects.
In this section, we describe the flow of Business Objects between two systems. Figure 8-7 shows an overview of the flow of Business Objects through the Interchange Server.
1. Create the Application Specific Business Object.
   When a User record is created in the application database, a record is created in the event table. The connector agent polls the event table and detects the new event. It retrieves the appropriate data from the application and creates an Application Specific Business Object.

2. Transfer the Application Specific Business Object.
   The connector agent transfers the Application Specific Business Object through a message queue to the connector controller in the Interchange Server.

3. Map the Application Specific Business Object to the Generic Business Object.
   To ensure that collaborations are application-independent and to handle differences in data modelling across applications, the Application Specific Business Object is mapped to a Generic Business Object. The Generic Business Object contains its own key, and cross-referencing is used to look up application-specific keys based on this key.
   Maps are discussed in more detail in Chapter 9, “Mapping” on page 247.

4. Send a request to the application
   Before a request is sent to an application, the Generic Business Object is mapped to an Application Specific Business Object. Keys are looked up in the relationship tables.

5. Send a response to the Interchange Server
   The application sends the Application Specific Business Object back to the Interchange Server as a response. Before the Business Object is sent, it is filled with keys if new ones are created (for example, if the verb is create). The keys are then stored in the Interchange Server relationship tables.

### 8.2 Designing Business Object definitions

This section describes how to design Business Objects and the methodology involved in the process.

#### 8.2.1 Using a methodology

Designing Business Objects, as with designing any application, requires a structured way of documenting what to develop. The important thing is not which methodology you use, but that you actually use one.

The components of a good methodology contain at least the following:

- Functional Overview and Scope
Provide a brief description of the business context in which the Business Objects will function, and to which collaborations it relates.

► Generic Business Objects

Describe the Generic Business Objects, whether you design them yourself or IBM provides them.

The design should contain:

– A Business Object hierarchy diagram, showing the parent Business Object, its child Business Objects, and their cardinality or relationship.

– One table for each Business Object specifying each attribute with name, type, if it is a key attribute, and any other relevant information.

► Application Specific Business Objects

Describe the Application Specific Business Objects for the source and target applications. In addition to the same design information as for Generic Business Objects, Application Specific Business Object design should also contain:

– Some form of documentation describing the underlying data structure from which the Business Object will be instantiated. With relationship databases this is normally done through an Entity Relationship Diagram (ERD).

– Any application-specific information on attributes

The next step in the design process is to describe the mappings and relationships in which the Business Objects depend.

Tip: WebSphere Business Integration Server Express contains a large list of packaged Business Object definitions. Check if definitions are sufficient for your project. Optionally, you can extend the definition to meet your requirements.

8.2.2 Standards

Consider using standards when designing and naming Business Objects. These practices will help make the application design consistent and improve readability and maintainability of the integration solution. Some suggestions are as follows:

► Naming conventions

The name of the Generic Business Object should represent the Business Object that is being supported. Use the following naming conventions:

– No spaces or underscores.
Business Object attributes
- Make the first attribute the Business Object id.
- The last attribute must be ObjectEventId.
- If the attribute is a foreign key, make the name the foreign Business Object plus Id, for example, EmployeeId.
- Be consistent when using abbreviations.
- Do not use characters other than letters, and do not use spaces.
- Capitalize the first character of the attribute name and concatenated names, for example, UserName and Password.

8.3 Developing Business Object definitions

This section provides an overview of Business Object Designer Express and how you can use the tool to create Business Object definitions.

8.3.1 Overview of Business Object Designer Express

In this section we describe different ways of opening Business Object Designer Express and a Business Object definition.

Opening Business Object Designer Express

Business Object Designer Express can be opened from the Start menu, the System Manager, or from another tool. To open Business Object Designer Express, do the following:
From the Start menu, do the following:

Select Start → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Business Object Designer Express.

From the System Manager, do the following:

a. From the tools menu, select Tools → Business Object Designer or press Ctrl+4.

b. Expand the Integration Component Library in which you want to create the Business Object definition. Right-click the Business Objects folder and select Create New Business Object.

c. Double-click an existing Business Object, either from the Integration Component Library, or from the User Project. This will open the selected Business Object definition.

From any other tool, do the following:


Opening a Business Object definition

There are three ways in which you can open a Business Object definition:

From the System Manager

Open a Business Object definition by double-clicking the Business Object in the Integration Component Library, or a User Project.

From the left pane

A list of Integration Component Libraries are available in the left pane. Expand a library to see the list of Business Objects within the library. Double-clicking a Business Object will open the Business Object definition in the right pane.

From the file

You can open a Business Object definition from the file by selecting File → Open From File. The file must be in one of the formats listed below.

File formats

You can save Business Object definitions to file in three different formats:

- .xsd: an XML Schema Definition.
  Business Object definitions are saved to this format by default (if you select File → Save).
- .in or .txt: Interchange Server format.
- .xls: a spreadsheet
An entire Integration Component Library can be saved to file in either the Interchange Server format or a spreadsheet. Select File → Copy All To One File, type in the file name, select the file format and the project you wish to export.

**Note:** We were unable to save Business Object definitions to a spreadsheet format. This was because of a bug in the WebSphere Business Integration Server Express version we used.

### 8.3.2 Developing an example

In this chapter we describe how to develop Business Object definitions using Business Object Designer Express.

We will cover the following topics:

- Creating the Integration Component Library
- Creating a Generic Business Object (Customer) with an existing child Business Object (Address)
- Creating an Application Specific Business Object (MyAppCust)

The Business Objects we create in this chapter will be used in the Chapter 9, “Mapping” on page 247 and Chapter 11, “Administration tools” on page 379.

The scenario for the example development is a simple one, where we synchronize users between two imaginary applications, MyApp and SAL301R.

**Important:** The Business Objects and other objects we create in the example development must not be confused with the object in the sample application.

### Creating the Integration Component Library

Business Objects, maps and other objects are created within an Integration Component Library (ICL). ICLs are projects within the System Manager. We will create a new ICL project for the example development:

1. From the System Manager Menu, select File → New → Integration Component Library.
2. Type GettingStartedICL as the project name.
3. Click Finish.

### Enabling the Address Business Object

We will be using the packaged Address Business Object definition as a child object to the Customer. We will not be making any changes to this definition.
1. Expand the **WBIExpressLibrary Integration Component Library**, and the **Business Objects** folder.

2. Expand the **GettingStartedICL Integration Component Library**.

3. Drag the **Address** Business Object from WBIExpressLibrary to the Business Objects folder in GettingStartedICL. This will copy the definition from WBIExpressLibrary and make it available in GettingStartedICL.

**Creating the Customer Business Object**

For example purposes we will be creating the Customer Business Object definition from scratch. In a real project you will usually base this definition on an existing one (for example, the Customer Business Object definition in WBIExpressLibrary).

1. Right-click the **Business Objects folder** in GettingStartedICL and select **Create New Business Object**.

2. Enter the following Business Object details:
   - Business Object Name: Customer
   - Application-specific Information: Leave this blank since we are creating a Generic Business Object
   - Create in project: Select **GettingStartedICL**

3. Do the following for each simple attribute specified in Figure 8-8 on page 227:
   a. Right-click the **ObjectEventId** attribute and select **Insert Above**. This will create a new empty line.
   b. Type the attribute name, **CustomerId**.
   c. Select the attribute type, **Integer**.
   d. Select the Key column if it is a key attribute, as with CustomerId.
   e. Type the maximum length if any. You may have to remove the default 255.
   f. Type in the default value if any.

4. Add the Address child Business Object attribute:
   a. Insert a new line above the ObjectEventId attribute.
   b. Type the attribute name: Address.
   c. Select the **Address** Business Object from the list of types.
   d. Accept the default cardinality of 1, according to our design, a customer can only have one address.

5. Select **File → Save** to save the Customer Business Object definition.

6. Click the **General** tab to see the supported verbs. Accept the defaults Create, Retrieve, Update and Delete.

Figure 8-8 on page 227 depicts the Customer attributes.
Creating the MyAppCust Business Object

The MyAppCust is a Business Object specific for an imaginary MyApp application and its underlying model. It is a flat Business Object, meaning it has no child Business Objects. In Chapter 9, “Mapping” on page 247 we describe how MyAppCust is mapped to the Customer Generic Business Object.

1. Create the Business Object as described in step 1 in “Creating the Customer Business Object” on page 226.

2. Enter the following Business Object details:
   - Business Object Name: MyAppCust
   - Application-specific Information: TN=CUST
   - Create in project: Select GettingStartedICL

3. Add the attributes as specified in Figure 8-9 on page 228.

4. Save the Business Object definition.

Figure 8-9 on page 228 depicts the MyAppCust attributes.
8.4 Developing with Object Discovery Agents

Object Discovery Agents (ODAs) are used to generate Business Object definitions for Application Specific Business Objects. An Object Discovery Agent is an optional component of an adapter. When an adapter is installed, its Object Discovery Agent is automatically installed.

If you are developing your own adapter, you can develop an Object Discovery Agent using the Object Discovery Agent Development Kit.

For more information about Object Discovery Agent refer to Chapter 7, “Adapters” on page 159.

See the Online Help for details on using the Object Discovery Agent Development Kit.

8.4.1 Developing an example using JDBCODA

In this example we will use the JDBC Object Discovery Agent to create Business Object definitions for the tables USR and USR_ADDR in the SAL301R database.

Important: The Business Object we are creating for SAL301R in this chapter are not the same as the Business Object in the sample application.
On details about the specifics of the JDBC Object Discovery Agent, refer to Chapter 7, “Adapters” on page 159.

**Note:** Before you can use the JDBC Object Discovery Agent, you must update the batch file that starts it with the following information:

- **Line setting DRIVERPATH**
  
  Append the database driver path with the location of the specific driver you want to use.
  
  `<db2_install_path>\java\db2java.zip`

- **Line setting DRIVERLIB**
  
  Set the correct path to the specific driver DLL libraries you want to use.
  
  `<db2_install_path>\bin`

1. Select **Start → Programs → WebSphere Business Integration Server Express → Adapters → Object Discovery Agents → JDBC**.

2. From Business Object Designer Express select **File → New Using ODA**.

3. Select the Agent.

   If the JDBC Object Discovery Agent has not been previously located, click **Find Agent**. The agent will be visible in the list of located agents.

   **Note:** The Object Discovery Agent must be running in order to use it. Otherwise you will see this error message:

   "Unknown error. while connection to agent JDBCODA"

Select the **JDBC agent** and click **Next**.
4. Configure the agent.

Configure the JDBC agent for the specific database you want to connect.

a. Specify the following information:
   
   - User Name: `<db_user>`
   - Password: `<db_password>`
   - Database URL: `jdbc:db2:SAL301R`
   - Database Driver: `COM.ibm.db2.jdbc.app.DB2Driver`
   - Default BOPrefix: `SAL301R`

   Leave the rest of the fields with their default values.

b. Enter SAL301R in the Current profile field.

c. Click **Save** to save the profile. This lets you load the profile for later use.

d. Click **Next** to continue.
**Note:** The correct naming standard for the Business Object prefix is SAL301R_ (with an underscore). In this example we omit the underscore to distinguish between the Business Objects we create here and the objects created in the sample application.

5. Select the source.

Select the source tables on which you want to generate Business Objects. In the Name column you can browse the database tree. The top levels are the schema names. You can generate Business Objects based on tables, views and stored procedures.


b. Expand **Tables**.

c. Select **USR** and **USR_ADDR** by pressing **Ctrl** while selecting both tables.

d. Click **Next**.

---

![Figure 8-11 Configuring the JDBC Object Discovery Agent](image)
6. Confirm source nodes for Business Objects.
   a. Confirm that you have selected the USR and USR_ADDR tables.
   b. Click **Next**

7. Generate Business Object definitions.
   At this step, you specify the Business Object properties for the selected tables.
   a. Provide the following information:

---

**Tip:** If you know for which table you want to generate a Business Object definition, click **Use this object instead**. This opens a dialog box where you can enter the path to the specific table. In the case of the USR table, this would be `<db_user>:USR`.

Note that you can’t use this feature when you want to create multiple Business Object definitions.
• Prefix: leave as the default SAL301R
• Verbs: Deselect RetrieveByContent from the list. The only verbs in this example are Create, Retrieve, Update and Delete.
• Add Stored Procedure Attributes: select No.

b. Click OK. Generating Business Object definitions will start.

![Figure 8-13 Business Object properties for selected tables](image)

8. Save Business Object definitions.

At this step you specify where you want to save the generated Business Object definitions. You can save them to a project, a file, or both.

a. Provide the following information:

• Save to project: GettingStartedICL
• Select to automatically open the Business Object definitions, and to stop the Object Discovery Agent.

If you select to save the definitions to file, a dialog box appears where you can specify the filename, location and the file format.

b. Click Finish.

This will create two Business Objects:
Open the Business Object definitions as displayed in Figure 8-14 to explore their attributes.

```
1. Right-click the ObjectEventId column in SAL301RUSR and select Insert Above.
2. Enter the following information:
   - Name: USRADDR
   - Type: SAL301RUSR_ADDR
   - Cardinality: 1
3. Select File → Save to save SAL301RUSR.
```

### 8.4.2 Developing an example using XMLODA

In this example, we will use the XML Object Discovery Agent to create Business Object definitions for the Customer DTD specified in Example 8-1 on page 235. In this DTD file we have specified a Customer element containing one (1) Address element. We have also specified the attributes for the elements.
Example 8-1  Customer DTD

```xml
<!-- Customer declarations -->
<!ELEMENT Customer (Address)>
<!ATTLIST Customer
  CustomerIdCDATA
  FirstNameCDATA
  LastNameCDATA
  Status CDATA>
<!ELEMENT Address EMPTY>
<!ATTLIST Address
  AddressIdCDATA
  AddressLine1CDATA
  AddressLine2CDATA
  ZipCodeCDATA
  City CDATA
  State CDATA
  CountryCDATA
  PhoneNumberCDATA>
```

For details about the specifics of the XML Object Discovery Agent, refer to Chapter 7, “Adapters” on page 159.

1. Select **Start → Programs → WebSphere Business Integration Server Express → Adapters → Object Discovery Agents → XML**.

2. From Business Object Designer Express select **File → New Using ODA**.

3. Select the agent.

    If the XML Object Discovery Agent has not been previously located, click Find Agent. After a while the agent should be visible in the list of located agents.

    **Note:** The Object Discovery Agent *must* be running in order to use it; otherwise you will get an error message stating:

    *Unknown error. while connection to agent XMLODA*

a. Select the **XML agent** and click **Next**.
Configure the agent.

Configure the XML agent for the specific database to which you want to connect.

a. Specify the following information, leaving the rest of the fields with the default values:
   - FileName: the file in which you stored the Example 8-1 on page 235.
   - BOPrefix: XML
b. Enter Customer in the field Current profile.
c. Click Save to save the profile. Doing this lets you load the profile later on.
d. Click Next to continue.
5. Select the source.

Select the source elements upon which Business Objects will be generated. In the Name column you can browse the elements with the top level being the DTD file.

a. Select XML_Customer and click Next. This will generate Business Objects based on all the elements.
6. Confirm the source nodes for Business Objects.
   a. Confirm that you have selected all the elements.
   b. Click Next

7. Generate Business Object definitions.
   In this step you specify the Business Object properties for the selected tables.
   a. Provide the following information:
      • Prefix: leave the default XML
      • Verbs: Ensure that create, retrieve, update and delete are selected.
   b. Click OK. This will start generating the Business Object definitions.

8. Save Business Object definitions.
   At this step you specify where you want to save the generated Business Object definitions. You can save them to a project, a file or both.
   a. Provide the following information:
      • Save to the project: GettingStartedICL
Select to automatically open the Business Object definitions, and to stop the Object Discovery Agent.

b. Click **Finish**. This will create the following Business Objects:

- XML_Customer, which represents the DTD file.
- XML_Customer_Customer, which represents the Customer (ROOT) element.
- XML_Customer_Address, which represents Address element.

**Note:** The XML Object Discovery Agent will automatically add application-specific information to the Business Object attributes:

- **XMLDeclaration attribute**
  The value is set to type=pi.
- **DocType attribute**
  The value is set to type=doctype.
- **ROOT attribute**
  The value is set to elem_name=<root_element_name>.
- **Child Business Objects**
  The value is set to elem_name=<element_name>.
- **Attributes**
  The value is set to attr_name=<attribute_name>.

Open the XML_Customer Business Object definitions in Business Object Designer Express. Expand the child Business Object to view the entire structure as displayed in Figure 8-14 on page 234.

**Important:** The XML Object Discovery Agent does not automatically set the ROOT element as the key. The result of this is XML_Customer does not have any key attributes, and saving it will fail. You are required to set the ROOT element as a key attribute manually immediately after generating the Business Objects.

This appears to be a bug in XML ODA, and applies to generation based on both data definition files, and schema definition files.

1. Set the ROOT element as a key attribute.
2. Save the Business Object definition.
8.5 Deploying Business Object definitions

Business Objects can be deployed to an Interchange Server in two ways:

- Through a user project to the Interchange Server
- Directly to the Interchange Server

**Deploying Business Object definitions through a user project**

An Interchange Component Library (ICL) cannot be deployed to a server. To deploy components within an ICL to a server you can create a user project. User projects contain reference objects (like shortcuts) to one or more ICLs. When you open an object in a user project, you actually open the corresponding object in the ICL. A user project can be deployed in its entirety to an Interchange Server.

Follow these steps to create the GettingStartedICS user project and to deploy it to the server:

1. Start the Interchange Server and connect to it.
2. In the System Manager, select File → New → User Project.
3. Enter GettingStartedICS as the project name.
4. Select the GettingStartedICL project.
5. Click Finish.
6. Right-click the newly created project and select **Deploy user project**.
7. Select the Interchange Server to deploy to.
8. Select to deploy the entire user project and click **Finish**.

You will see a message box stating that the objects have been successfully deployed to the server.

**Deploying objects directly to the Interchange Server**

Even though you cannot deploy an entire ICL to an Interchange Server, you can deploy single objects within an ICL. This is done by dragging the object from the ICL to the server in the Interchange Server Component Management outline. You can select and deploy multiple objects in this manner.

### 8.6 Using packaged Business Object definitions

When you install WebSphere Business Integration Server Express the installation procedure creates an Integration Component Library called WBIExpressLibrary. This ICL contains over 200 Business Object definitions. Most of these Business Objects are used in conjunction with the collaboration templates provided in the Collaboration capacity packs. Even though Collaboration capacity packs are separately licensed, you can use the Business Objects associated with them free of charge.

Table 8-1 lists the Business Objects related to the collaboration templates that are imported in the base installation. See the Online Help for details on other Business Object and how they relate to the collaboration templates in the Collaboration capacity packs.

**Table 8-1  Installed Business Object definitions**

<table>
<thead>
<tr>
<th>Definition</th>
<th>Children</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>ControllerChild</td>
<td>CollaborationFoundation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WrapperFoundation</td>
</tr>
<tr>
<td>Contract</td>
<td>Address</td>
<td>ContractWrapper</td>
</tr>
<tr>
<td></td>
<td>ContractCustomer</td>
<td></td>
</tr>
<tr>
<td>Definition.</td>
<td>Children</td>
<td>Collaboration</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CustomerPartner</td>
<td>CustomerAddress</td>
<td>CustomerPartnerWrapper</td>
</tr>
<tr>
<td></td>
<td>PhoneInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RoleUsage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerInformation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerProfileData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerAccountingData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerBankData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerCreditData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CreditAreaCreditData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerTaxData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerTaxExemptLicenseData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerSalesOrderData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerShippingData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerBillingData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerForeignTradeData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>Customer</td>
<td>Same as CustomerPartner + RelatedCustomerRef</td>
<td>CustomerSynchronization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CustomerWrapper</td>
</tr>
<tr>
<td>EmailNotification</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EmailTemplate</td>
</tr>
<tr>
<td>Item</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ItemWrapper</td>
</tr>
<tr>
<td>Site</td>
<td>CustomerAddress</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhoneInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RoleUsage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerProfileData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerSalesOrderData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerShippingData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerBillingData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustomerForeignTradeData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RelatedCustomerRef</td>
<td></td>
</tr>
<tr>
<td>Employee</td>
<td>EmployeeHR_Info</td>
<td>SystemTest</td>
</tr>
<tr>
<td></td>
<td>EmployeeAddress</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EmployeeJob</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EmployeeHR.Misc</td>
<td></td>
</tr>
</tbody>
</table>
Figure 8-19 shows a hierarchical view of the Customer Business Object definition.

![Figure 8-19](image_url)

**Figure 8-19  Hierarchical view of Customer Business Object definition**
When you create Business Objects using JDBCODA, the agent adds some application-specific information on the Business Object header and its attributes:

- **Business Object header**: TN=<table name>; SCN=
- **Attributes**: CN=<column name>

In this section, we will describe some of the most common application-specific attributes for JDBC-related Business Objects.

**TN - table name**

TN is set on the Business Object header, and specifies the table name to which the Business Object relates. This is automatically set if you create the Business Object using ODA.

- **Format**: TN=<table name>
- **Example**: TN=USR_ADDR

**SCN - status column**

SCN is also set on the Business Object header. When a delete is performed, the adapter can either perform a physical delete of a record, or update a specified status column with a value. SCN specifies the column name and value to update if a delete is performed.

- **Format**: SCN=<status column name>:<status value>
- **Example**: SCN=EMPSTATUS:DELETED

**Note**: If you set ChildUpdatePhyDelete property to true on the JDBC connector, it will perform a physical delete of the children objects, even if a status column is specified in the Business Object header.

---

**8.7 JDBC application-specific information**

When you create Business Objects using JDBCODA, the agent adds some application-specific information on the Business Object header and its attributes:

- **Business Object header**: TN=<table name>; SCN=
- **Attributes**: CN=<column name>

In this section, we will describe some of the most common application-specific attributes for JDBC-related Business Objects.

**TN - table name**

TN is set on the Business Object header, and specifies the table name to which the Business Object relates. This is automatically set if you create the Business Object using ODA.

- **Format**: TN=<table name>
- **Example**: TN=USR_ADDR

**SCN - status column**

SCN is also set on the Business Object header. When a delete is performed, the adapter can either perform a physical delete of a record, or update a specified status column with a value. SCN specifies the column name and value to update if a delete is performed.

- **Format**: SCN=<status column name>:<status value>
- **Example**: SCN=EMPSTATUS:DELETED

**Note**: If you set ChildUpdatePhyDelete property to true on the JDBC connector, it will perform a physical delete of the children objects, even if a status column is specified in the Business Object header.
**CN - column name**

CN specifies the column name that the attribute represents. The values for this attribute are automatically set if you create the Business Object using ODA.

- Format: \texttt{CN=column name}
- Example: \texttt{CN=FIRTST_NAME}

**UID - unique identifiers**

UID specifies how the adapter is to generate unique identifiers (primary keys). Do not specify this property if you do not want unique identifiers to be generated. Usually this property will be set to AUTO, meaning that the unique identifier is generated automatically. See the Online Help for other possible values.

Example: \texttt{UID=AUTO}

**FK - parent/child relationship**

FK specifies the foreign key relationship between parent and child objects. The value of this property depends on whether the parent/child relationship is stored in the parent or the child Business Object.

- If the relationship is stored in the parent, specify the child object name and the attribute name to be used as foreign key.
  
  Format: \texttt{FK=child object name}.\texttt{child object attribute}

  A typical example of this is a customer (CUSTOMER table) that has multiple addresses (ADDRESS table). The ADDRESS table would contain an ADDRESS_ID column as primary key, and a CUSTOMER_ID column that is a foreign key to the CUSTOMER_ID primary key in the CUSTOMER table. In the Business Object you specify the attribute ADDRESS.CUSTOMERID as a foreign key attribute, and add the following meta-data:

  \texttt{FK=CUSTOMERID}

- If the relationship is stored in the child, specify only the attribute name in the parent Business Object.

  Format: \texttt{FK=parent object name}

  An example of this relationship is an employee that is located at an office. Since you want to share the office information between several employees you specify an OFFICE_ID column in the EMPLOYEE table that is a foreign key to the OFFICE_ID primary key in the OFFICE table. In the Business Object, you specify the attribute EMPLOYEE.OFFICEID as a foreign key attribute and add the following meta-data:

  Example: \texttt{FK=OFFICE.OFFICEID}
CONTAINMENT - parent/child relationship

The CONTAINMENT attribute specifies whether or not the parent Business Object owns the child Business Object or not. This is only applicable for single cardinality relationships.

- Specify the value NO_OWNERSHIP if the parent Business Object shares the child Business Object. For example, in the EMPLOYEE/OFFICE case, the office information is shared among several employees. We do not want the office information to be deleted when an employee is deleted. We add the following to the meta-data for the attribute EMPLOYEE.OFFICEID:

  Example: CONTAINMENT=NO_OWNERSHIP

- Specify the value OWNERSHIP if the parent Business Object owns the child Business Object. For example, in the EMPLOYEE/OFFICE case, the office may have an ADDRESS child Business Object (single cardinality) that specifies the location of the office. If we delete the office, we want to delete its address. We add the following to the meta-data for the attribute OFFICE.ADDRESSID:

  Example: CONTAINMENT=OWNERSHIP

KEEP_RELATIONSHIP - parent/child relationship

The KEEP_RELATIONSHIP attribute specifies how the adapter is supposed to handle updates of child Business Objects. This is only applicable for multiple cardinality relationships. The meta-data is set on the child Business Object attribute, for example on EMPLOYEE.ADDRESS.

- Specify the value TRUE to prevent the connector from deleting existing child data that is not represented in the source Business Object:

  Example: KEEP_RELATIONSHIP=TRUE

- Specify the value FALSE to make the connector delete all existing child data before adding the child Business Objects contained in the source Business Object:

  Example: KEEP_RELATIONSHIP=FALSE

Note: Logically, if the cardinality of the child object is N, the parent/child relationship must be stored in the child. If the cardinality is 1, both ways are possible.
Mapping

This chapter provides an overview of the role of mappings and relationships in IBM WebSphere Business Integration Server Express.

This chapter covers the following topics:
9.1 Developing mappings

Data mapping is the process of transforming (or mapping) data from one application-specific format to another. Mapping is central to the process of transferring information between different applications, and to providing collaborations (business processes) that are independent of specific applications. By mapping data between Application Specific Business Objects and Generic Business Objects, WebSphere creates an environment that allows for the use of "best of breed" applications. The WebSphere Business Integration system provides a modular and extensible architecture for easy maintenance of your maps. Map Designer Express is a graphical development tool for creating, testing and modifying maps.

All examples in this chapter are based on a simplified customer synchronization scenario between two applications called MyApp and SAL301R. All related files are contained in the integration component library GettingStartedICL. The Business Object definitions are developed in Chapter 8, “Business Objects” on page 209.

9.1.1 Developing maps

To launch Map Designer Express, do one of the following:

- From System Manager, perform one of these actions:
  a. From the Tools menu, select Map Designer Express.
  b. Click a map folder in a project to enable the Map Designer Express icon in the System Manager toolbar. Then click the Map Designer Express icon.
  c. Right-click the map folder in a project and select Create New Map from the Context menu.
  d. Right-double-click a map to start Map Designer Express with the selected map opened.

- From a development tool, such as Business Object Designer Express, Relationship Designer Express, or Process Designer Express, perform one of these actions:
  a. From the Tools menu, select Map Designer Express.
  b. In the Programs toolbar, click the Map Designer Express button.

- Use a system shortcut: click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Map Designer Express.
A map itself consists of a number of individual transformations from the attribute(s) of the source Business Object to the attribute(s) of the target Business Object.

Following standard transformations, you can specify:

- **Set Value**
  For any attribute required by the target Business Object but not specified by the source Business Object.

- **Move**
  If the source and target attributes are to be identical.

- **Join**
  If the target attribute is a combination of two or more source attributes.

- **Split**
  If the target attribute is only a part of a source Business Objects attribute. You may specify a delimiter and index position within the source attribute.

- **Submap**
  You can call a submap to obtain a value for any destination attribute, but submaps are most commonly used for the following:
    - To specify transformations between child Business Objects.
    - To make a map modular.

- **Cross-reference**
  To maintain the identity relationships of the Business Objects.

- **Custom**
  In a Custom transformation, you use the *Activity Editor* to customize the activity for the transformation graphically or to enter Java code to transform the source attribute to the destination attribute.

To make common tasks easier, use the shortcuts in Table 9-1 on page 250.
The Map Designer Express has four tabs, as shown in Figure 9-1 on page 251:

- **Table**
  
  The Table tab of Map Designer Express displays mapping information in a tabular format that lists all mapping attributes and transformations. You design transformations by selecting attributes and transformation rules from the corresponding drop-down lists. If the map result depends on the execution order of transformation rules, you can change this order by specifying the correct value in the first column.

- **Diagram**
  
  The Diagram tab of Map Designer Express provides a drag-and-drop interface for defining and reviewing the transformations. You view and design maps in the map workspace, which displays on the right side of the window.

- **Messages**
  
  The Messages tab displays the map's messages. A message consists of a message ID and its associated message text. Messages defined for a map can be useful if you want a map to send customized messages to the Interchange Server Express log.

- **Test**
  
  The Test tab provides an interface for testing maps and viewing the results. In this tab, you can run tests to verify that transformations are working properly.

---

**Table 9-1  Shortcuts for common mapping tasks**

<table>
<thead>
<tr>
<th>Key</th>
<th>Mapping Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control+Drag</td>
<td>Move</td>
</tr>
<tr>
<td>Control+Drag (multiple source attributes)</td>
<td>Join</td>
</tr>
<tr>
<td>Control+Drag (source attribute is child Business Object)</td>
<td>Submap</td>
</tr>
<tr>
<td>Alt+Drag</td>
<td>Split</td>
</tr>
<tr>
<td>Drag</td>
<td>Custom</td>
</tr>
</tbody>
</table>
Maps can be designed using either the table view or the diagram view. You can toggle between those by clicking the corresponding tabs of Map Designer Express. The table view is helpful if you have Business Objects with several attributes; the diagram view is more suitable if you favor a drag-and-drop programming style.

All transformation rules will be listed in the Transformation Rules row. Right-clicking Show Source opens the activity editor and shows the generated Java code in read-only mode.
By using the Messages tab, you can create your own set of messages for displaying customized mapping information.

The Test tab provides an interface for testing maps and viewing the results. For more information, see 9.2.2, “Relationship deployment” on page 297.

### 9.1.2 Defining a simple map

It is good practice to describe the layout of a map prior to development, for instance, by using a spreadsheet application. Very often, you need to ask many different groups within a business to get all the information about the semantic background of Application Specific Business Object attributes. The layout will help you to check the completeness of this information.

We will now define a mapping between our Application Specific Business Object `SAL301RUSR_ADDR` and the Generic Business Object `Address` based on Table 9-2.

**Table 9-2  Map definitions for SAL301RUSR_ADDR to Address**

<table>
<thead>
<tr>
<th>SAL301RUSR_ADDR</th>
<th>Address</th>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Verb]</td>
<td>[Verb]</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>USRADDRPOSTCD</td>
<td>PostalCode1</td>
<td>Move</td>
<td>Zip Code</td>
</tr>
<tr>
<td>USRADDRCNTRYCD</td>
<td>Country</td>
<td>Move</td>
<td>Name of country</td>
</tr>
<tr>
<td>USRADDRRGN</td>
<td>State</td>
<td>Move</td>
<td>State or region</td>
</tr>
<tr>
<td>USRADDRNAME</td>
<td>Addressline1</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>USRADDRSTRTRNAME</td>
<td>Addressline2</td>
<td>Move</td>
<td>Street</td>
</tr>
<tr>
<td>USRADDRBLDGNUM</td>
<td>AddressLine3</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>USRADDRUNITNUM</td>
<td>AddressLine4</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>USRADDRPOBOX</td>
<td>POBox</td>
<td>Move</td>
<td>Postbox</td>
</tr>
<tr>
<td>USRID</td>
<td>AddressId</td>
<td>Move</td>
<td>Address key, foreign key</td>
</tr>
</tbody>
</table>
Launch the Map Designer Express in one of the previously described ways. If you start with a new map, you will be guided by a Create New Map wizard. If this wizard does not start automatically, select File → New... or press Ctrl+N.

1. On the first panel of the Create New Map wizard, choose the ICL project in which the map has to be created. Choose `GettingStartedICL` and then click Next. See Figure 9-2.

![Figure 9-2  Create New Map wizard, panel 1](image)

2. Now select your source Business Object. Choose `SAL301RUSR_ADDR`. See Figure 9-3 on page 254.
3. Click **Next** and select your target Business Object, in our case the Generic Business Object named **Address**. See Figure 9-4 on page 255.
4. Finally, name the map `Sub_SAL301R_ADDR_Address` (because we will use this map later as a submap only) and select a mapping direction of **Application Specific to Generic**. When you click the **Finish** button, the wizard completes the initial map definition. See Figure 9-5 on page 256.
Creating map definitions

Now proceed with the mapping: click the source Business Object attribute; press and hold the Control key and drag the cursor to the destination attribute, then release the mouse button first and then the Control key.

The result will look similar to Figure 9-6 on page 257.
Save your changes to the project by pressing Ctrl+S or selecting File → Save → To Project. You will be prompted to compile the map. The result messages will be written to the status area of the Map Designer.

You may also force a compile by pressing F7.

Testing your map

Map Designer Express provides a test facility that allows you to deploy and test a map on a running Interchange Server Express instance.

Select the Test tab to access the test facility and select EVENT_DELIVERY from the Calling Context drop-down menu. Define appropriate values for each of the attribute fields in the source Business Object. You may save the Business Object for later use.
Click the **Run Test** icon from the menu bar and select a target Interchange Server Express instance name from the panel. You may click the **Search** button beside the Server Name field to search for a running server instance.

Provide a user ID and password to access the Interchange Server Express (by default, these values are set to *admin* and *null*).

Make sure that the **Deploy map** and **Deploy dependent** objects check boxes are selected and click **Connect**. Figure 9-8 on page 259 shows the Connect dialog box.
Figure 9-8  Connect to Interchange Server Express

Make sure that the **Deploy map** and **Deploy dependent objects** check boxes are selected and click **Connect**. Figure 9-8 shows the Connect dialog box.

A successful test displays the correct values for the destination attributes as shown in Figure 9-9 on page 260.
You can set a breakpoint to pause map execution just before the transformation of the destination attribute on which the breakpoint is set. The use of breakpoints lets you step through map execution (by pressing F10) and check the sequence and the results of individual operations. You can set as many breakpoints as you need. The test run can either be finished by pressing F8 or stopped by selecting **Debug → Stop Test Run**.

You may use the same test facility to test maps running on the server. Select **Debug → Advanced → Attach** to select a running server instance. In the next dialog, you can select the map you want to debug. You can set breakpoints as before. These breakpoints will pause server execution of maps as long as these maps are used.

You can stop the debugging session by clicking **Debug → Advanced → Detach**.
Creating reverse maps
Typically, maps are used in pairs. In most places where a map is used, a map is also needed in the opposite direction. Using Reverse Map automates the steps required to create a reverse map. The following table shows the standard transformation rules that Map Designer Express currently supports (Current map column) and the transformation rules that Reverse Map currently includes (Reverse map column).

To create a reverse map, select Tools → Reverse Map. If your map uses Custom, Set or Cross-References, no mapping will be created automatically.

Restriction: The generated XML for the reverse maps refers to the wrong Dynamic Loadable Modules (DLM) version, specifying 1.0.0 instead of the needed 2.0.0. At the time we wrote this redbook, there was no direct fix for this problem; to work around it, the user has to manually edit the maps xml file as follows:

Find the line:

```xml
<DLMInfo Name="[map_name]" Version="1.0.0">
```

and manually change it to:

```xml
<DLMInfo Name="[map_name]" Version="2.0.0">
```

9.1.3 Creating advanced mapping
A simple move from source to target attribute will fulfill your mapping requirements only in certain cases. Most of the time, additional customizing is still required.

Joining attributes
Using the join transformation, you can concatenate two or more source attributes into one target attribute.

For instance, the source Business Object might store the area code, telephone number, and extension in separate attributes, while the destination Business Object stores these values together in one attribute.

In addition to joining the attributes, you can reorder them and insert delimiters, parentheses, or other characters. For instance, when joining separate area code and telephone number attributes into a single attribute, you might want to insert parentheses around the area code.
Our simple example has the requirement to join the FirstName and LastName attributes and map them to the Name attribute between Customer and MyAppCust.

Table 9-3  Map definitions for Customer to MyAppCust

<table>
<thead>
<tr>
<th>Customer</th>
<th>MyAppCust</th>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Verb]</td>
<td>[Verb]</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>CustomerId</td>
<td>CustId</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Status</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>CreatedOn</td>
<td>Created</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Address.City</td>
<td>City</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Address.PostalCode1</td>
<td>Zip</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Address.State</td>
<td>State</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Address.AddressLine1</td>
<td>Address</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>FirstName</td>
<td>Name</td>
<td>Join</td>
<td>Join FirstName and LastName with a Blank as delimiter</td>
</tr>
<tr>
<td>LastName</td>
<td>Name</td>
<td>Join</td>
<td>Join phonenumberparts with a minus sign as delimiter</td>
</tr>
<tr>
<td>Address.PhoneCountry</td>
<td>Phone</td>
<td>Join</td>
<td></td>
</tr>
<tr>
<td>Address.PhoneNumber1</td>
<td>Phone</td>
<td>Join</td>
<td></td>
</tr>
<tr>
<td>Address.PhoneExtension</td>
<td>Phone</td>
<td>Join</td>
<td></td>
</tr>
</tbody>
</table>

The steps we followed for this operations are as listed:

1. Start the Map Designer Express.
2. Select the Customer Business Object as the source.
3. Select the MyAppCust Business Object as the target.
4. Name the map: Customer_MyAppCust.
5. Use drag and drop to define all move rules based on the information given in Table 9-3.
6. From the Diagram view, select your source attributes (FirstName and LastName) and while pressing Ctrl, drag them over to the target attribute (Name).
7. Release the **CTRL** key after the drag operation, otherwise the transformation rule will be changed to custom. A join dialog will prompt you to define the order and delimiter of the join transformation. Click **OK**.

8. Repeat steps 6 and 7 for joining source attributes PhoneCountry, PhoneNumber1 and PhoneExtension to the target attribute Phone. Specify a - as the delimiter.

![Figure 9-10  Join attributes dialog](image)

Previously defined join transformations may be altered by double-clicking the **Join** icon in the transformation rule column.

**Splitting attributes**

To split a source attribute into two or more destination attributes, you specify the transformation for each destination attribute separately. This type of
transformation is called a *split transformation*. For instance, to split a source attribute, such as *phone_number*, into three separate destination attributes, such as *area_code*, *tel_number*, and *extension*, you specify the transformations for *area_code*, *tel_number*, and *extension* separately.

To define a split transformation, select your source attribute and, while pressing **Alt**, drag it to the first target attribute; after releasing the mouse button, a split dialog will prompt you for the delimiter and index position within the source attribute. The example view shows you how the split transformation will be executed.

You will need to define the split for the transformation of the *Name* attribute of the MyAppCust Business Object to the attributes *Firstname* and *LastName* of the Customer Business Object.

To ease your development, you can open the existing map *Customer_MyAppCust* and select **Tools → Reverse Map**, then name the new map *MyAppCust_Customer*.

You can review the generated split mapping by double-clicking one of the split rules.
**Submap transformation**

A submap is a map that is called from within another map, called the main map.

**When to use a submap**

Submaps can be used in the following situations:

▶ To modularize a map

Using submaps can improve the modularity of your maps by isolating common transformations that can be reused in more than one map.

▶ To specify transformations between child Business Objects

When the source and destination attributes contain multiple-cardinality child Business Objects, a submap is useful to specify their transformations. Typical examples of multiple-cardinality child Business Objects are the multiple addresses of a customer or the multiple line items in an order.
Using Expression Builder to call a submap

Coding the map call allows you to write more varied operations than Map Designer Express supports graphically. For example, you can use a submap to provide a value for an attribute that does not contain a child Business Object or use a submap that has multiple inputs and outputs. You will find more information on this topic in the book *Map Development Guide*, available at:


Developing a submap is identical to “normal” map development. To select a submap using the Diagram view, press Ctrl, select the source child object, and drag it onto the target child object. A dialog will prompt you for the selection of a submap definition.

We will use a submap for transforming the SAL301R_USR object to Customer.

Table 9-4  Map definitions for SAL301RUSR to Customer

<table>
<thead>
<tr>
<th>SAL301RUSR</th>
<th>Customer</th>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRID</td>
<td>CustomerId</td>
<td>Move</td>
<td>ID of the user</td>
</tr>
<tr>
<td>USRFIRSTNAME</td>
<td>FirstName</td>
<td>Move</td>
<td>User’s first name</td>
</tr>
<tr>
<td>USRLASTNAME</td>
<td>LastName</td>
<td>Move</td>
<td>User’s last name</td>
</tr>
<tr>
<td>USRCDON</td>
<td>CreatedOn</td>
<td>Move</td>
<td>Creation date</td>
</tr>
<tr>
<td>USRUSERNAME</td>
<td>Status</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This is for demonstration purposes only</td>
</tr>
<tr>
<td>[Verb}</td>
<td>[Verb]</td>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>SAL301USR_ADDR R</td>
<td>Address</td>
<td>Submap</td>
<td>The mapping of address information will be a submap</td>
</tr>
</tbody>
</table>

1. Start the Map Designer Express.
2. Select the SAL301RUSR Business Object as the source.
3. Select the Customer Business Object as the target.
4. Name the map: SAL301R_Customer.
5. Use drag and drop to define all move-rules from Table 9-4.
6. Use **CTRL** and drag and drop to move the Business Object SAL301USR_ADDR to Address.

7. Select the **Sub_SAL301RADDR_Address** submap and click **OK**.
   You are now ready to test your map.

---

**Figure 9-12  Select submap**

**Providing conditions when calling the submap**

Often, the map requires some programming logic to determine when to call a submap. In some cases, certain conditions must be true for the submap to be called. This logic goes into the main map, in the attribute that contains the submap's destination Business Object, and before the call to the `runMap()` method. If you enter these conditions in the Conditions area of the Submap dialog, Map Designer Express adds these conditions to the if statement with which it surrounds the `runMap()` call.
Sample conditions could be (according to product documentation):

\[
\text{currentBusObj\_For\_ObjSAL301RUSR\_USRADDR.getInt(“USRID”) = 1}
\]

or

\[
\text{currentBusObj\_For\_ObjSAL301RUSR\_USRADDR.getString(“USRADDRNAME”).equals(“Doe”)}
\]

**Tip:** Keep the following points in mind when entering these conditions:

- You can enter only one condition, although the condition can have multiple clauses, combined with an AND (&&) operator.
- Do not end the line with a semicolon because the condition that you enter is turned into an if clause in the destination attribute's generated code.

You can make use of conditions for creating polymorphic mappings and dynamic routing scenarios; literally, this means that you will call a different submap depending on the attribute’s value of your source Business Object.

**Customizing maps using the Activity Editor**

Using Activity Editor, you can specify the flow of activities for a specific transformation rule graphically, without knowing programming or Java code. For each transformation rule in Map Designer Express, you can develop one activity and its sub-activities. You can develop the associated attribute's transformation code graphically, modify it, and have the tool generate the corresponding Java code.

You launch Activity Editor directly from Map Designer Express (see "Starting Activity Editor") by double-clicking a **Custom** transformation rule. At startup, the Activity Editor communicates with System Manager to discover the set of activities allowed. After you have finished designing the activity for a particular transformation rule, you save the changes in Activity Editor, and they are communicated to Map Designer Express.
The Activity Editor initially starts in Graphical view. By clicking the Java tab of the Activity Workbook window, you can switch to the Java view, showing the automatically generated Java code. By clicking Tools → Edit Code, this Java code may be further customized. Transformation rules containing customized code will always start the Activity Editor in Java view.

The Graphical view has four main windows: the Activity Workbook window, the Library window, the Content window, and the Properties window.

- **Activity Workbook window**
  
  This window is the main activity editing area, and is usually referred to as the editing canvas. It is also known as the activity canvas or graphical canvas. This area is where you drag and drop the function blocks.

- **Library window**
  
  This window contains a tree view of the available function blocks and, optionally, the named groups. The function blocks are arranged in folders according to their purpose, and you can expand them to show the actual function blocks. You can also view the function blocks as icons in the Content window.
In addition, the Library window contains the following folders:

- **System**
  This folder contains system elements that can be added to the editing canvas. System elements include comments, descriptions, labels, to-do tags, and constants.

- **Library**
  This folder enables you to customize the Library window. It contains any user-defined function blocks that have been specified in the Activity Settings view in System Manager.

- **My Collection**
  This folder enables you to create a collection of the components you use most often. You can place regular function blocks in this folder, or you can create your own reusable component group.

- **Variables**
  This folder contains global variables accessible to the current activity. It typically contains the port's Business Object variables, all of the other Business Objects and variables defined in the scenario, and the global variable cwExecCtx, a variable that contains the execution context for the current map. This variable is defined in the code that Map Designer Express generates for every map.

- **Content window**
  This window contains a large icon list of the available function blocks under the currently selected folder in the Library window. You can select a function block to view its description and properties in the Properties window, or drag-and-drop a function block onto the editing canvas to create part of the activity flow.

- **Properties window**
  This window displays the properties of the currently selected function block in a grid like layout. Some properties are editable; others are read-only.

By browsing the library window, you will find many functions to help you to develop custom transformations; for example:

- **String functions**
- **DateTime functions**
- **Mathematical functions**
The number of available functions can be extended by:

- Importing Web Services
- Building function blocks
- Importing external libraries (Java class files)

**Important:** The following examples may be reviewed by opening the `SAL301RUSR_Customer_Using_CustomTransformations` map.

To demonstrate the Activity Editor, we will first change the String content of the source attribute `USRUSERNAME` to upper case and move the result to the source attribute `Status`.

1. Drag `USRUSERNAME` and drop it onto `Status`.
2. The transformation rule is now `Custom` (if not, change the rule by using the drop-down menu).
3. Double-click `Custom`; the Activity Editor opens.
4. In the Library window, click `Library → String-Upper Case` and drag and drop the `Upper Case` function onto the Activity Workbook.

Now you can draw connections between the three elements to realize a data flow between them.

Your results should look similar to Figure 9-14 on page 272.
To change the format of an incoming date attribute, you can use the custom transformation shown in Figure 9-15 on page 273.
Special transformations may require the use of customized Java code. Following the example creates a database lookup using the source attribute and moves the returned value into the target attribute.

To get this example running you need to define a database connection pool named SAMPLEPOOL.

**Example 9-1  Simple Java database lookup**

```java
{
    CwDBConnection connectn = null;
    Vector theRow = null;
    Enumeration theRowEnum = null;
    String theColumn1 = null;
    String var_7 = null;
    String var_9 = null;

    var_7 = ObjMQ_Employee.get("ID") == null ? "" : ObjMQ_Employee.get("ID").toString();

    try {
        // Obtain a connection to the database
```
connectn = getDBConnection("SAMPLEPOOL");
}
catch (CwDBConnectionFactoryException e) {
    System.out.println(e.getMessage());
    throw e;
}

// Test for a resulting single-column, single-row, result set
try {
    Vector argValues = new Vector();
    argValues.add(Integer.valueOf(var_7));
    // Send the SELECT statement to the database
    connectn.executeSQL("select salary from wende.staff where id = ?", argValues);
    // Loop through each row
    while (connectn.hasMoreRows()) {
        // Obtain one row
        theRow = connectn.nextRow();
        int length = 0;
        if ((length = theRow.size()) != 1) {
            logInfo("Expected result set size = 1," + " Actual result state size = "+ length);
        }
        // Get column values as an Enumeration object
        theRowEnum = theRow.elements();
        // Verify that column values exist
        if (theRowEnum.hasMoreElements()) {
            // Get the column value
            var_9 = theRowEnum.nextElement().toString();
            trace("Element returned : " + var_9);
        }
    }
}

// Handle any exceptions thrown by executeSQL()
catch (CwDBSQLException e) {
    System.out.println(e.getMessage());
}

BusObj destBusObj = ObjEmployee;
String destAttr = "Salary_Base";
// Set the destination value only if neither source nor destination are null.
//
if ((var_9 != null) && (destBusObj != null)) {
    if (dataValidationLevel >= 1) {
        if (!ObjEmployee.validData("Salary_Base", var_9)) {
            // Log a warning about this failure.
            String warningMessage =
"Invalid data encountered when attempting to set the value of the "Salary_Base" attribute of BusObj 'ObjEmployee' while running map 'getName() + getName() + '". The invalid value was 'var_9 + var_9 + '"."

// Log a warning about this failure.
logWarning(warningMessage);
if (failOnInvalidData) {
    // Fail the map execution with a warning message.
    throw new MapFailureException(warningMessage);
}
}
destBusObj.setWithCreate(destAttr, var_9);


9.1.4 Using Web services

You can use Web services to enhance your custom transformation rules. For instance, you can use a Web service to get current exchange rates or stock information.

To use a Web service, you have to first register the Web service through the System Manager as follows:

1. Go to the GettingStartedICL project, right-click Web Services and select Register a new web service.
2. Select Enter a URL for WSDL file and click Next.
3. In the next window, select **Convert_Temparture.wsdl** from the Chapter 9 code folder. This Web Service provides two methods for conversion between Fahrenheit and Celsius.

4. The next window lists all available methods and you are required to name the service for their further use in Interchange Server Express.
5. By clicking **Finish**, the Web service will be imported into the System Manager. You can now double-click the freshly imported Web service to open its definition.

6. To test the service:
   a. Right-click the service and select **Test**.
   b. Click **Create New Request**.
   c. Right-click the request Business Object and select **Add An Instance**.
      
      **Hint:** you do not need to add instances to the configuration Meta Objects.
   d. Fill in the value(s) in the instance attribute(s).
   e. Click **Invoke Web Service**.
   f. Inspect the response Business Object; you should see the response value.
7. If your Web service is ready to run, you can right-click and select **Export to Activity Editor**.

If you now open your map and start the Activity Editor, you will find the Web Services under **My Library → Web Services** in the Library window. From now on, you can use the Web Service methods for every built-in function.

**Important:** Web services support in the server is not enabled by default. If you must invoke Web services from the server, start the server manually (not using the InterChange Server shortcut from the **Start → Programs** menu): Navigate to the ProductDir\bin folder and run the start_server_webservice.bat file with the usual command-line parameters.
9.1.5 Relationship mappings

Maintaining relationships between key values of the participating Business Objects or setting attribute content based on a lookup table is one of the key requirements of integration projects. You can use the two types of relationships by using Map Designer Express:

- If there is a dynamic relationship between the source and target Business Object, you are defining a cross-reference mapping: see “Cross-referencing identity relationships” on page 279.

- If there is a static relationship between the source and target Business Object, you are defining a lookup relationship mapping: see “Lookup relationships” on page 280.

9.1.5, “Relationship mappings” on page 279 describes how you can define static and dynamic relationships.

Cross-referencing identity relationships

You define cross-referencing relationship mapping by selecting the source attribute with your mouse, dragging it onto the target attribute and selecting Cross-referencing. See Figure 9-19 on page 280.

Important: A cross-referencing relationship is defined at the Business Object level, so only Business Objects can be the source and target of a cross-reference mapping.
Lookup relationships
You define a lookup relationship mapping as a custom mapping by using the Activity Editor. We will use the translation of a country abbreviation as a simple example.

1. Create a US_Info_Info map in the GettingStartedICL Integration Component Library, mapping from the US_Info Business Object to Info Business Object.

2. Drag the UnitName source attribute to your Unit target attribute and select Custom (see “Customizing maps using the Activity Editor” on page 268).

3. Open the Activity Editor by double-clicking the Custom field.
4. Select **General → Relationship → Static Lookup** in the library window and drag it onto the design window.

5. Define three constants by right-clicking the design window and selecting **New Constant**.
   a. Enter **Units** as the constant name and wire this constant to the *relationship name* input of the Static Lookup activity.
   b. Enter **USInfo** as the constant name and wire this constant to the *participant name* input of the Static Lookup activity.
   c. Enter **true** as the constant name and change the type of the constant to boolean. Wire this constant to the *inbound?* input of the Static Lookup activity.

   **Important:** If you create the corresponding outgoing map (*Info_German_Info* in our case), you have to enter **false** in this constant field.

6. Wire the source attribute to the source value input of the Static Lookup activity and the lookup value output to the target attribute. See Figure 9-20 on page 282.

7. Save and close the Activity Editor.
If you test this map, you will see that the target attribute of our Generic Business Object will be filled with a numeric value. This value is the relationship instance ID of the current relationship instance.

You can review the corresponding outgoing map by selecting the Info_German_Info map. To test the lookup, enter the instance ID you had previously created into the Country source attribute.

Important: To run this example, you first have to create the static relationship as described in 9.2, "Relationships" on page 286.

9.1.6 Map documentation

You can create a map document to see all transformations in a single map or between two maps. While checking a map, you might want to view all of its transformations in a single operation, rather than opening and viewing each attribute separately. To do so, you can create a map document that contains all transformations. A map document provides you with an automated way to document native map transformations.
To create map documents, select **File → Create Map Document**.

![Create Map Document](image)

**Figure 9-21  Create map documentation**

Select the project, map name and destination folder for the HTML map document. After clicking **Save** or **Save/View**, the document is generated and optionally displayed by your default Web browser.
SAB404RAddress_EmployeeAddress

Linked Attributes:

<table>
<thead>
<tr>
<th>Source Attribute</th>
<th>Transformation Rule</th>
<th>Destination Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjSAB404RAddress [Verb]</td>
<td>Move</td>
<td>ObjEmployeeAddress [Verb]</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_TYPE</td>
<td>Move</td>
<td>ObjEmployeeAddress.AddressType</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_LINE1</td>
<td>Move</td>
<td>ObjEmployeeAddress.AddressLine1</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_LINE2</td>
<td>Move</td>
<td>ObjEmployeeAddress.AddressLine2</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_CITY</td>
<td>Move</td>
<td>ObjEmployeeAddress.City</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_ZIP</td>
<td>Move</td>
<td>ObjEmployeeAddress.PostalCode</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_STATE</td>
<td>Move</td>
<td>ObjEmployeeAddress.State</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_COUNTRY</td>
<td>Move</td>
<td>ObjEmployeeAddress.Country</td>
</tr>
<tr>
<td>ObjSAB404RAddress.ADDR_KEY</td>
<td>Move</td>
<td>ObjEmployeeAddress.EmployeeId</td>
</tr>
<tr>
<td></td>
<td>Set Value</td>
<td>ObjEmployeeAddress.AddressId</td>
</tr>
</tbody>
</table>

The table shows how an attribute of a source business object is mapped to an attribute of a destination business object. To display Java code for a particular attribute, click its transformation rule hyperlink.

Unlinked attributes in map 'SAB404RAddress_EmployeeAddress':

Figure 9-22  The generated map documentation file

Some hints
We suggest that you use the map editor table view for mapping larger Business Object structures because you may prefer a simpler view which shows only the current mapped attributes.
Attribute comments can improve the readability of your map. However, Map Designer Express does not automatically generate a comment for an attribute. Table 9-5 (based on product documentation) provides some suggested standards for attribute comments based on the type of transformation associated with the destination attribute.

Table 9-5  Map attribute comments

<table>
<thead>
<tr>
<th>Situation</th>
<th>Setting for Attribute Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the child Business Object is not mapped</td>
<td>=No mapping</td>
</tr>
<tr>
<td>Set Value transformation</td>
<td>=SET VALUE(value)</td>
</tr>
<tr>
<td>Move transformation</td>
<td>=MOVE</td>
</tr>
<tr>
<td>Join transformation</td>
<td>=JOIN(srcAttr1, srcAttr2,...)</td>
</tr>
<tr>
<td>Split transformation</td>
<td>=SPLIT(srcAttr[index])</td>
</tr>
<tr>
<td>For child Business Objects, when the mapping is done without calling a submap to indicate the object has to be expanded to see its attributes</td>
<td>=Mapping here</td>
</tr>
<tr>
<td>If the code to call the submap is generated</td>
<td>=SUBMAP(mapName)</td>
</tr>
</tbody>
</table>
| If the attribute's mapping contains Mapping API calls that implement relationships, such as: RetrieveInstances() RetrieveParticipants() maintainSimpleIdentityRelationship() maintainCompositeRelationship() All other methods in the IdentityRelationship class except foreignKeyLookup() and foreignKeyXref() | =Relationship(type) where type can be:  
  - identity  
  - lookup  
  - custom |
| If the attribute's mapping contains foreignKeyLookup()                   | =foreignKeyLookup()           |
| If the attribute's mapping contains foreignKeyXref()                     | =foreignKeyXref()             |
| If the attribute's code does not contain anything except setting the verb | =SET VERB                     |
9.2 Relationships

Most application systems have a different representation for equivalent data, that is, for data having the same semantic meaning. The association between these different representations is called a relationship and the related Business Objects are called participants.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Setting for Attribute Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom transformation that is <em>not</em> one of those listed above (relationship or foreign key)</td>
<td>=CUSTOM(summary)</td>
</tr>
</tbody>
</table>

Relationships are classified according to the following categories based on the type of data in the participant and the number of instances of each participant that can be related:

- A lookup relationship establishes an association between data, such as attributes in Business Objects. The data can be related on a one-to-one,
one-to-many, or many-to-many basis. Lookup relationships typically transform non-key attributes whose values are represented with codes, such as marital status or currency code. Use a lookup relationship if these attribute values are static; that is, new values are not often added or existing values removed. As illustrated in Figure 9-23 on page 286.

- An identity relationship establishes an association between Business Objects or other data on a one-to-one basis. For each relationship instance, there can be only one instance of each participant. Identity relationships typically transform the key attributes of Business Objects, such as ID numbers and product codes. Use an identity relationship if key values are dynamic, that is, key values are frequently added or existing values are removed. See Figure 9-24.

- A non-identity relationship establishes an association between Business Objects or other data on a one-to-many or many-to-many basis. For each relationship instance, there can be one or more instances of each participant. An example of a non-identity relationship is an RMA-to-Order transformation, in which a single RMA (Return Materials Authorization) Business Object can yield one or more Order Business Objects.
A relationship consists of two parts:

- A relationship definition, developed using the Relationship Designer Express. Relationship definitions are part of your Integration Component Library. They include:
  - The relationship name, to uniquely describe the relationship. It should not be identical to the associated Business Object.
  - The definitions of relationship participants: participant name, type, database table and stored procedure.
  - The relationship database definitions: database type, database name and login information. By default, Interchange Server Express uses its own repository database.

- A relationship instance, the runtime object. An instance is simply a set of database column uniquely related by an relationship instance ID. For example, in Figure 9-23 on page 286, the instance ID for the highlighted relationship is 101. The following runtime objects exist:
  - Relationship tables, created at deployment time. A participant table holds the data and the instance ID for one relationship participant.
  - Stored procedures, created at deployment time, to maintain integrity of the relationships.
  - For identity relationships, Interchange Server Express automatically populates the relationship tables during map execution, based on relationship definition and the values from the actual Business Object. For lookup relationships, you have to fill the relationship tables with the appropriate data. See 9.2.4, “Relationship management” on page 303.
To launch Relationship Designer Express, do one of the following:

- From System Manager, perform one of these actions:
  a. From the Tools menu, select Relationship Designer Express.
  b. Click a Relationship folder in a project to enable the Relationship Designer Express icon in the System Manager toolbar. Then click the Relationship Designer Express icon.
  c. Right-click the Relationships folder in a project and select Relationship Designer Express from the Context menu.
  d. Right-click a relationship in the Dynamic or Static folder and select Edit Definitions from the Context menu.

- From a development tool, such as Business Object Designer Express, Map Designer Express, or Process Designer Express, perform one of these actions:
  a. From the Tools menu, select Relationship Designer Express.
  b. In the Programs toolbar, click the Relationship Designer Express button.

- Use a system shortcut:
  Click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Relationship Designer Express.

Important: Interchange Server Express assigns default names for the participant's table and stored procedure:

- [Relationshipname]_[Participant]_T for the table
- [Relationshipname]_[Participant]_SP for the stored procedure

However, you can specify your own names using the advanced settings dialog of the Relationship Designer Express.

Important: For Relationship Designer Express to be able to access relationships stored in System Manager, Relationship Designer Express must be connected to an instance of System Manager. The preceding steps assume that you have already started System Manager. If System Manager is already running, Relationship Designer Express will automatically connect to it.
Defining lookup relationships
A lookup relationship defines the relation between two or more different pieces of data, for example the mapping between country and international phone prefix.

If a participant for a lookup relationship is of the type Data, this means that it could hold every data type, such as String, long, int, double, float, or boolean.

To demonstrate the development process, we will create a lookup relationship that will replace American unit names with German ones (we do not take into account that there are more differences than the name only).

Table 9-6 Lookup value table

<table>
<thead>
<tr>
<th>American Unit Name</th>
<th>German Unit Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fahrenheit</td>
<td>Celsius</td>
</tr>
<tr>
<td>Pound</td>
<td>Kilogram</td>
</tr>
</tbody>
</table>

Perform the following steps to create a relationship definition:

1. Start the Relationship Designer Express from the GettingStartedICL.
2. Create a relationship name by doing one of the following:
   - From the File menu, select New Relationship Definition.
   - Use the keyboard shortcut of Ctrl+N.
   - In the Standard toolbar, click the New Relation button.
   - Right-click GettingStartedICL:Relationship Definitions and select New Relationship Definition.
3. Name the icon for the relationship definition: enter Units.
4. Create a participant definition for each Business Object to be related. To do this, select the relationship definition name and perform one of the following actions:
   - From the File menu, select Add Participant Definition.
   - In the Standard toolbar, click the New Participant button.
   - Right-click the relationship name and select Add Participant Definition.
5. For each participant definition, name the icon for the participant definition. For our example, we need the USInfo and GerInfo participants.

Important: Relationship definition names can be up to eight characters long, can contain only letters and numbers, and must begin with a letter.
a. Select View → Participant Types.

b. Drag the Data icon onto USInfo participant.

c. Repeat the last step for GerInfo.

6. If you want to create the relationship tables in the default location and with default names, you can now save the relationship by pressing Ctrl+S. For our example, we will change the defaults now.

7. Open the Advanced Settings dialog by performing one of the following actions:
   – From the View menu, select Advanced Settings.
   – Right-click the relationship name and select Advanced Settings.

8. Enter the information for the database location, user name and password.
   a. Enter jdbc:db2:SMB_USER for the database URL.
   b. Enter smbadmin for the Login.
   c. Enter smbP4$$word for the Password.
   d. Select DB2 for the Type.

9. Make sure that the Static checkbox is selected.
10. Click **GerInfo**.
   a. Enter GerInfo as the Table name.
   b. Enter SP_GerInfo into the Name field for the stored procedure.

11. Click **UsInfo**.
   a. Enter UsInfo as the Table name.
   b. Enter SP_UsInfo into the Name field for the stored procedure.
12. Click OK.

13. Click Ctrl+S to save the relationship.

The relationship is now ready for deployment and testing.

**Defining the identity relationship**

An identity relationship defines the relation between two or more key attributes, for example between different implementations of customer IDs or order IDs.

A participant for an identity relationship is of type Business Object.

To demonstrate the development process, we will create an identity relationship that will maintain the relationship between US_Info and German_Info for their key attributes (CountryId and ID, respectively).

Perform the following steps to create a relationship definition:

1. Start the Relationship Designer Express from the GettingStartedICL.

2. Create a relationship name by doing one of the following:
   - From the File menu, select **New Relationship Definition**.
   - Use the keyboard shortcut of **Ctrl+N**.
   - In the Standard toolbar, click the **New Relation** button.
– Right-click **GettingStartedICL:Relationship Definitions** and select **New Relationship Definition**.

3. Name the icon for the relationship definition: enter IDs.

**Important:** Relationship definition names can be up to 8 characters long, can contain only letters and numbers, and must begin with a letter.

4. Create a participant definition for each Business Object to be related. To do this, select the relationship definition name and perform one of the following actions:
   – From the File menu, select **Add Participant Definition**.
   – In the Standard toolbar, click the **New Participant** button.
   – Right-click the relationship name and select **Add Participant Definition**.
   – For each participant definition, name the icon for the participant definition. For our example we need USId, GerId and InfoId participants.

5. Select **View → Participant Types**.

6. Drag the **US_Info** Application Specific Business Object on the USId participant.

7. Drag the **German_Info** Application Specific Business Object on the GerId participant.

8. Drag the **Info** Generic Business Object on the GerId participant.


10. Drag the **CountryId** attribute on the US_Info participant definition.

11. Expand the German_Info Business Object.

12. Drag the **ID** attribute on the German_Info participant definition.


14. Drag the **CountryID** attribute on the Info participant definition.

15. If you want to create the relationship tables in the default location and with default names, you can now save the relationship by pressing **Ctrl+S**. For our example, we will change the defaults now.

16. Open the Advanced Settings dialog by performing one of the following actions:
   – From the View menu, select **Advanced Settings**.
   – Right-click the relationship name and select **Advanced Settings**.
17. Enter the information for the database location, user name and password.
   - Enter `jdbc:db2:SMB_USER` for the database URL.
   - Enter `smbadmin` for the Login.
   - Enter `smbP4$$word` for the Password.
   - Select `DB2` for the Type.

18. Make sure that the **Static** checkbox is unchecked.
Figure 9-29 Database settings for identity relationship

19. Click **GerId**.
   a. Enter GerId as the Table name.
   b. Enter SP_GerId into the Name field for the stored procedure.

20. Click **USId**.
    a. Enter USId as the Tablename.
    b. Enter SP_USId into the Name field for the stored procedure.

21. Click **InfoId**.

Select the **IBM WBI managed** checkbox.

**Tip:** If you select **IBM WebSphere Business Integration Managed**, Interchange Server Express will maintain the instance ID's and will not use a separate table for the Generic Business Object.
22. Click **OK**.

23. Click **Ctrl+S** to save the relationship.

The relationship is now ready for deployment and testing.

### 9.2.2 Relationship deployment

During the deployment of a relationship, Interchange Server Express creates the necessary database tables and stored procedures. To empower Interchange Server Express for an automatic compile of these stored procedures, you have to set up a special environment. The DB2 online documentation ([http://publib.boulder.ibm.com/infocenter/db2help/index.jsp](http://publib.boulder.ibm.com/infocenter/db2help/index.jsp)) explains how to set up the environment if you are using the Microsoft Visual C++ compiler.

**Tip:** For our environment, we used a setup based on an article from the DB2 DeveloperWorks Web site ([http://www.ibm.com/developerworks/db2/library/techarticle/0306haungs/0306haungs.html](http://www.ibm.com/developerworks/db2/library/techarticle/0306haungs/0306haungs.html)) using the open source Gnu C Compiler (GCC). You will find an easy procedure which you can follow step by step. You must reboot your workstation to make sure that all settings are in place.
If your environment is in place, drag and drop the relationship definitions from GettingStartedICL onto your running Interchange Server Express instance.

You will get a success message, as shown in Figure 9-31. Reboot Interchange Server Express to activate the relationships.

![Deploy Warning](image1.png)

**Figure 9-31   Deploy success**

If you get an error message such as the one shown in Figure 9-32, click OK and check the Interchange Server Express console for additional error messages. A common reason may be that you have not deployed all dependent Business Objects.

![Error](image2.png)

**Figure 9-32   Deploy error**

A procedure for saving relationships and their database objects and the deployment on other servers is described in 13.5.2, “Relationships” on page 517.

### 9.2.3 Testing relationships

You can test your relationships in the context of the mappings using them. For lookup relationships, this is explained in 9.1.5, “Relationship mappings” on page 279.

**Important:** To test a lookup relationship, you have to populate the relationship tables in advance; see 9.2.4, “Relationship management” on page 303.

Use the following steps to test your identity relationship:

1. Go to **GettingStartedICL** and open the **Us_Info_Info** map.
2. Drag and drop ObjUs_Info onto ObjInfo.

3. Select the IDs relationship, then select USId as participant and click OK; see Figure 9-33.

4. Go to the test page, fill in some values and click test.
   - Make sure that calling context is set to EVENT_DELIVERY.
   - You will be forced to compile your map.
   - Notice the value of CountryId; it was generated by Interchange Server Express (see Figure 9-34 on page 300).

You will see the mapping results.

5. Save the result Business Object for later use.
6. Open the Info_German_Info map.

7. Create an outgoing cross-reference between ObjInfo and ObjGerman_info; see Figure 9-35 on page 301. Select GerId as participant.

8. Test the map using the saved Business Object from step 5 on page 299.
   – Make sure that calling context is set to SERVICE_CALL_REQUEST.
   – You will be forced to compile your map.
Figure 9-35  Outgoing cross-reference
Figure 9-36 Test outgoing cross-reference

9. Save the resulting Business Object.

10. Open the German_Info_Info map.

11. Test the map.
   a. Make sure that calling context is set to SERVICE_CALL_RESPONSE.
   b. Click ObjGerman_Info.
   c. Click Load From and select the saved Business Object from step 9.
   d. Fill in a value in the ID field.
   e. Select the Info Generic Business Object in the Generic Business Object drop-down list; you will have now an additional Business Object in your left test pane.
   f. Click the Info Business Object in the left test pane.
   g. Click Load From and select the saved Business Object from step 5 on page 299.
   h. Run the test.
9.2.4 Relationship management

When you deploy relationship definitions with the option Create Schema enabled or by dragging and dropping from your Integration Component Library, Interchange Server Express generates a relationship table for each participant.

Table 9-7 Participant table structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTANCEID</td>
<td>Integer</td>
<td>The relationship instance ID</td>
</tr>
<tr>
<td>STATUS</td>
<td>Integer</td>
<td>Set to zero (0) when the participant is active</td>
</tr>
</tbody>
</table>
Unlike relationship tables that hold data for identity relationships, lookup tables do not get populated automatically. You must populate these tables by inserting data into their columns. You can populate a lookup table in either of the following ways:

- Create a script that contains SQL INSERT statements to fill the lookup table with the desired data.
- Use Relationship Manager to add rows to the lookup table; see “Using the Relationship Manager” on page 305.

**Important:** When you use SQL statements to insert participant data into a lookup table, make sure you provide a value for the STATUS, LOGICAL_STATE, and TSTAMP columns. All values are required for IBM WebSphere Business Integration Server Express tools to function correctly. In particular, omission of the TSTAMP value causes Relationship Manager to be unable to retrieve the participant data; if no timestamp value exists, Relationship Manager raises an exception.

Use the SQL statements from Example 9-2 on page 305 to populate a relationship instance for our sample Units relationship.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGICAL_STATE</td>
<td>Integer</td>
<td>Will be set to one (1) when the participant is deactivated</td>
</tr>
<tr>
<td>LSTATE_TSTAMP</td>
<td>Timestamp</td>
<td>Date of deactivation for the participant instance.</td>
</tr>
<tr>
<td>UPDT_TSTAMP</td>
<td>Timestamp</td>
<td>Date of last modification for the participant instance.</td>
</tr>
<tr>
<td>TSTAMP</td>
<td>Timestamp</td>
<td>Date of creation for the participant instance.</td>
</tr>
<tr>
<td>DATA or attribute name</td>
<td>varchar 255</td>
<td>The participant data</td>
</tr>
</tbody>
</table>
Example 9-2  Insert relationship instance information using SQL

```
INSERT INTO USInfo
    (INSTANCEID, STATUS, LOGICAL_STATE, TSTAMP, data)
VALUES (1, 0, 0, current timestamp, 'Fahrenheit')

INSERT INTO GerInfo
    (INSTANCEID, STATUS, LOGICAL_STATE, TSTAMP, data)
VALUES (2, 0, 0, current timestamp, 'Celsius')
```

**Using the Relationship Manager**

Relationship Manager is an IBM WebSphere Business Integration tool that graphically displays runtime data in a relationship table. Relationship Manager is useful when you only need to add a few rows in the lookup table and to maintain identity relationships.

1. You can start the Relationship Manager in one of the following ways:
   - From Relationship Designer Express, perform this action:
     From the Tools menu, select **Relationship Manager**.
   - Use a system shortcut:
     Click **Start** → **Programs** → **IBM WebSphere Business Integration Server Express** → **Toolset Express** → **Administrative** → **Relationship Manager**.

2. To manage your relationships, you have to connect to Interchange Server Express first. You can do this in one of the following ways:
   - From the Server menu, select **Connect**.
   - Click the **Connect to Server** icon; see Figure 9-40 on page 308.
3. To select a deployed relationship, select one of the following actions:
   - Click **File → Open... (Ctrl+O)**.
   - Click the **Open Relationship** icon.

4. Now, select the relationship you want to maintain. Click **OK**.
   
   You can now choose whether you want to see all instances or only a subset based on certain selection criteria.

---

**Figure 9-38  Connect to serve**
Chapter 9. Mapping

5. Click **Get Instances**.

You can now:
- Browse existing instances
- Delete instances
- Deactivate and activate instances
- Create new instances or add participants to existing instances

**Tip:** Deactivated participants will be removed from the view. To activate them again, select **View → Show Deactivated Participants of Role** and the participant name. See Figure 9-40 on page 308.
Figure 9-40  Show deactivated participants
Collaborations

This chapter provides an overview of the role of collaborations in the WebSphere Business Integration Server Express. We describe the collaborations that are provided as part of WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus. We also describe the collaborations available as part of the Collaboration capacity packs.

We examine how to develop, implement, and deploy collaborations in the following sections:

► 10.1, “Introduction to collaborations” on page 310.
► 10.2, “Collaborations in WebSphere Business Integration Server Express” on page 318.
► 10.3, “Designing collaborations” on page 323.
► 10.4, “Developing collaborations” on page 326.
► 10.5, “Developing an example” on page 348.
10.1 Introduction to collaborations

In WebSphere Business Integration, business processes are described through collaborations. A collaboration is a software module that contains logic for coordinating the functionality of a business process and the exchange of data between the applications involved in the business process.

Collaborations act on Generic Business Objects (GBO), which are generic representations of data exchanged between disparate applications. When the data is transferred to an application, it is mapped to an Application Specific Business Object (ASBO) representing the underlying data model of the specific application. The same mapping is performed when data is retrieved from an application.

A typical WebSphere Business Integration solution will include several collaborations, each involving various applications. For example:

- A customer synchronization collaboration might involve synchronizing information between a Customer Relationship Management (CRM) application and an Enterprise Resource Planning (ERP) application.
- A bill of material (BOM) collaboration might involve synchronizing information across an ERP application, CRM application, and a Supply Chain Management (SCM) application.
- An order process might involve transferring orders entered through a Web-based e-commerce application to an ERP application and an external courier.

Collaborations use either connectors or the Server Access Interface, and sometimes both, to communicate with the applications. In the case of customer synchronization, the Interchange Server solution might consist of:

- A customer synchronization collaboration
- A connector for the CRM application
- A connector for the ERP application
- Application Specific Business Object definitions representing CRM-specific customer data and ERP-specific customer data
- A Generic Business Object definition representing a superset of customer data, and mapping between the ASBOs and the GBO

Figure 10-1 on page 311 highlights the collaboration in the reference architecture.
10.1.1 Collaboration components

In this section, we will describe the following components of collaborations:

- Collaboration templates
- Collaboration objects - which are based on collaboration template
- Collaboration groups - a collaboration using one or more other collaborations
- Ports - a collaborations external interfaces
- Scenarios - execution units in a collaboration

Collaboration templates
When you install a collaboration, for example through a Collaboration capacity pack, you install a collaboration template. The collaboration template contains the business process execution logic, but is not executable.

Collaboration objects
You create a collaboration object based on a collaboration template. In a collaboration object, you configure the bindings of connectors and specify configuration properties. A collaboration template may be used to create several
collaboration objects. A collaboration object can be deployed and run on a server.

**Collaboration groups**
A collaboration can be configured to use one or more other collaborations, forming a collaboration group. This provides data isolation, and enables you to develop multiple discrete collaborations that can be used in several related business processes.

The input from one collaboration to another in a group is a Generic Business Object.

**Scenarios**
A collaboration consists of one or more processing units, called scenarios. Picture a collaboration as a program and scenarios as sub-routines within that program.

A scenario specifies what should happen when it receives a specific type of Business Object with a specific event (verb). This means that you can have different scenarios depending on the event, for example, one for create, one for delete and so on. You can also have one scenario that handles all events.

**Subdiagrams**
You may also use scenarios as subdiagrams. This enables you to fragment the scenarios and ensure readability of the process. Subdiagrams are defined within a main scenario or a subdiagram, and cannot be run separately.

The CollaborationFoundation collaboration template as viewed in Figure 10-2 on page 313 is an example of a collaboration using several subdiagrams. It has two scenarios, Main and Retrieve, while the Main scenario has several subdiagrams.
Scenario logic

The business process logic in a scenario is separated into the following parts:

1. A trigger initiates the scenario, a Business Object with a specific verb.
2. One or more actions are performed, for example:
   - Inspect collaboration configurations properties
   - Inspect a Business Object
   - Create a new Business Object
   - Compare two Business Objects
   - Send to or retrieve a Business Object from an application and process it
   - Log information, a warning or an error message
3. The scenario ends.

Figure 10-3 on page 314 shows a simple logical overview of a hypothetical scenario.
Collaborations use ports to interact with applications or other collaborations. You are required to define a port for each interface in the application.

Ports are defined on the collaboration template, and consist of the following:

- **Name** - each port has a unique name
- **BO Type** - the output or input of a port is a defined Business Object
- **Verbs** - this is the scenario to start when an event occurs

Figure 10-4 on page 315 shows the ports defined for the SystemTest collaboration template. It states that the From port subscribes to the Employee Business Object, and that the Main scenario should run on each triggering event.
Collaboration templates do not specify which application or collaboration to use for each port.

When you create a new collaboration object, you have to bind the ports in the collaboration template. The ports can be bound to a connector, collaboration or a Web service.

10.1.2 Collaboration processing

In this chapter, we describe the different mechanisms in which a collaboration is triggered and interacts with applications:

- A connector in a publish-and-subscribe interaction
- An access request
- A service call
- Long-lived business processes

Publish-and-subscribe interactions

In a publish-and-subscribe interaction, a collaboration is triggered in the following way:

- A collaboration subscribes to a Business Object with a specific event that can trigger its execution, for example, Customer.Create.

- When an event occurs in an application, it is detected by the application’s event notification mechanism. The connector publishes the event as a Business Object to the collaborations that request it.

Depending on the connector, events can be published either synchronously or asynchronously.
Access requests
A collaboration can be designed to be triggered by direct calls that are sent by an access client, received by the Server Access Interface, and sent to the collaboration as Business Objects. In an Interchange Server implementation, calls sent to collaborations through the Server Access Interface are referred to as access requests. Access requests can originate from external sources or from sources that are configured within the Interchange Server implementation.

Access request interactions are useful when synchronous communication is important, for example, when a customer representative uses a Web browser to request inventory status information over the Internet.

Service calls
A collaboration can make requests to bound ports (connectors) and receive responses from the requests.

Requests
The collaboration makes its service call requests in the form of Generic Business Objects. The connectors transform the Generic Business Objects into data entities that are understood by the specific application or data format for which the connector is designed.

Responses
The service call responses that the collaboration receives from the connectors can be Business Objects containing business data (for example, in the case of retrieve requests) or status reports (successful or unsuccessful).

Supported service calls
Service calls can either be synchronous or asynchronous. Collaborations can support the following types of service calls:

- Synchronous outbound service call
  The synchronous outbound service call uses a synchronous request and response mechanism. The service call sends the request but does not complete until the response arrives and is processed.
  Synchronous service calls support compensation. In addition, they support a time-out value for long-lived business processes.

- Asynchronous outbound service call
  An asynchronous outbound service call sends a request from the collaboration but does not expect or wait for a response before continuing its processing.
  Asynchronous outbound service calls support compensation.
Asynchronous inbound service call

An asynchronous inbound service call waits to receive an incoming event and is used in conjunction with long-lived business processes. When an asynchronous inbound service call is created, it is given a time-out value. If the service call does not receive an incoming event before the time-out expires, an exception is raised.

Asynchronous inbound service calls are available only if the Long Lived Business Process Support option of the collaboration template has been enabled. The feature is used only for exchanges with connectors.

**Note:** Asynchronous inbound service calls do not support compensation.

Long-lived business processes

Collaborations have the capability to interact with connectors in a manner that persists in the context of a service call data flow for a specified period of time, without requiring the originating service call process to wait synchronously. If an appropriate response to the service call is received within the specified time period, the data flow is resumed and processing continues. This feature is referred to as long-lived business processes for collaborations, and can be enabled during the creation of the collaboration template.

10.1.3 Collaboration types

WebSphere Business Integration supports the following collaboration types.

**Synchronization collaborations**

*Synchronization collaborations* synchronize data across systems, and are the most common type of collaboration. They are typically used to synchronize *master* data, which is data that forms the foundation of an enterprise, such as Customer, Item, Contact or Employee information. Examples of synchronization collaborations are CustomerSync, ItemSync, and ContractSync.

**Wrapper collaborations**

A *wrapper collaboration* is a collaboration that handles the verification or synchronization of a Business Object for another collaboration. The calling collaboration sends a top-level Business Object that is referenced on its own triggering Business Object to the wrapper collaboration.
Examples of Wrapper collaborations are Item Wrapper, Customer Wrapper, and Contact Wrapper.

**Process collaborations**

Process collaborations contain abstracted process logic that go beyond the normal synchronization process. They handle tasks such as building an install base based on the receipt of a single triggering Business Object (Product Installation collaboration), verifying that an item has the appropriate extensions (view) before creating a purchasing document (Purchasing collaboration), or performing verification of various master data before the creation of sales orders (SalesOrderProcessing collaboration).

### 10.2 Collaborations in WebSphere Business Integration Server Express

In this section, we describe the collaborations in WebSphere Business Integration Server Express as well as the Collaboration capacity packs.

#### 10.2.1 Base collaboration templates

In this section, we list the collaboration templates that are available for use in both WebSphere Business Integration Server Express Plus and WebSphere Business Integration Server Express. Refer to the online help for details about each collaboration template and how to use them.

**Foundation templates**

There are two foundation templates available in WebSphere Business Integration Server Express. The idea of the foundation templates is to help developers speed up development of new collaboration templates and to make it easy to develop collaborations that adhere to WebSphere Business Integration standards. When developing new collaboration templates, always use either the foundation templates or one of the collaborations in the Collaboration capacity packs as a basis.

**CollaborationFoundation**

The CollaborationFoundation collaboration template contains the logic used by most collaborations for performing synchronization of data between a source and destination application.
The template contains basic functions as: synchronization, validation, aggregation, storage, retrieval, routing and filtering, error handling, and compensation. It also contains features such as:

- **CONVERT_CREATE**
  
  If a Create is received, perform an Update in the destination application if the entity already exists.

- **CONVERT_UPDATE**
  
  If an Update is received, perform a Create in the destination application if the entity does not exist.

- **USE_RETRIEVE**
  
  Retrieves the Business Object from the destination application before it synchronizes data. This property is useful when performing compensation processing and when setting the verb based on whether the Business Object already exists in the destination application.

- **ADDITIONAL_RETRIEVE**
  
  Retrieves the Business Object from the destination application after it synchronizes data. This property is useful when the source application requires a full-valued Business Object to be returned from the destination application, but the connector for the destination application does not return a complete Business Object after creating or updating its data.

The CollaborationFoundation also has five empty subdiagrams, each named Additional Processing, that enable you to extend the business process without modifying the base synchronization logic.

**WrapperFoundation**

The WrapperFoundation collaboration template contains logic for doing the following:

- **Verification**
  
  The collaboration verifies the existence of a Business Object by retrieving it from the destination application. If the Business Object is found, a success status is returned to the calling collaboration. If verification fails, the behavior of the wrapper collaboration depends on the setting of its CONTINUE_WITH_WARNING configuration property.

- **Synchronization**
  
  The collaboration synchronizes the Business Object by retrieving all values from the source application and sending the full-valued Business Object to the appropriate synchronization collaboration.
If the synchronization collaboration succeeds in creating the Business Object in the destination application, the wrapper collaboration returns a success status to the calling collaboration.

If the synchronization collaboration fails to create the Business Object in the destination application, the wrapper collaboration's behavior depends on the setting of its CONTINUE_WITH_WARNING configuration property.

To prevent the called synchronization collaboration from attempting to create an existing object and returning a failure status to the wrapper collaboration, use the following settings for the synchronization collaboration:

- Set the USE_RETRIEVE configuration property to true.
- Set the INFORMATIONAL_EXCEPTIONS configuration property to 3010.

Wrapper templates
In WebSphere Business Integration Server Express, the following wrapper collaboration templates are available for verifying and synchronizing Business Objects:

- ContactWrapper
- CustomerPartnerWrapper
- CustomerWrapper
- ItemWrapper
- SiteWrapper
- VendorPartnerWrapper

Test templates
The SystemTest and CustomerSynchronization collaboration templates are used to test the WebSphere Business Integration Server Express system by sending the Employee or Customer Business Object from one instance of Test Connector to another.

Using a collaboration template
Perform the following tasks to use an existing template:

1. Copy the template to your Integration Component Library.
2. Rename the template to reflect the process you are building.
3. Double-click the collaboration template to open it in Process Designer Express.
4. Use the Ports and Triggering Events tab of the Template Definitions window to change the Business Object types of the existing ports.
5. Perform any other necessary modifications to the template so it meets the requirements of your process.
6. Save and compile the collaboration template.
7. Create a collaboration object based on the template, and bind the ports to connectors.

10.2.2 Collaboration capacity packs

The Collaboration capacity packs provide pre-built collaboration templates for some common business processes.

The collaboration templates are grouped into the following solution areas:

- **Customer Relationship Management**
  Contains templates for synchronizing information such as contacts, billing information, suppliers, and contracts between Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems.

- **Financial and Human Resources**
  Contains templates for synchronizing information such as invoices, payroll, employees, and general ledger transactions across ERP, human resources, and financial systems.

- **Order Management**
  Contains templates for synchronizing information such as prices, items, returns, customers, and orders across ERP, CIM, and financial systems.

- **Procurement**
  Contains templates for synchronizing information such as inventory, purchase records, and vendor information across ERP, Supply Chain Management (SCM), and CRM systems.

For detailed information about the business processes and the collaboration templates in the Collaboration capacity packs, refer to the Online Help and the following Web site:

http://www.ibm.com/software/integration/wbicollaborations

**Installing a Collaboration capacity pack**

Follow these steps to install a Collaboration capacity pack:

1. Start the Launchpad by inserting the WebSphere Business Integration Server Express or Express Plus CD in your computer.
2. Select **Install Capacity Pack** from the left menu and then click the **Install Collaboration capacity pack** button.
   A new installation window will eventually appear with a welcome screen.
3. Click **Next** on the welcome screen.
4. Read the license agreement, select the button stating that you accept the terms, and click **Next**.

5. Select the Collaboration capacity pack you want to install and click **Next**.

![Collaboration Capacity Pack for WebSphere Business Integration Server Express Plus V4.3 Installer](image)

**Figure 10-5 Select Collaboration capacity pack to install**

6. Confirm that you have selected the correct Collaboration capacity pack and click **Next**.

7. The installation starts. Click **Finish** when the summary screen appears.

8. Exit the Launchpad.

**Note:** You can only have *one* Collaboration capacity pack installed at any time. If you try to install a second one, you will be notified that you must first uninstall the existing pack before installing a new one.

**Using a Collaboration capacity pack collaboration template**

Before you can use a Collaboration capacity pack collaboration template, you must import it into your Integration Component Library:

1. Right-click the Integration Component Library you want to import the collaboration template into and select **Import from Repository File**.

2. If you want to import one collaboration template, select the **Interchange Server repository file** option and click **Browse**.
a. Select the collaboration template you want to import. They are located in the `<wbi_install_path>\repository\CollaborationCP` directory.

b. Click Open.

3. If you want to import all collaboration templates, select the Import from a Repository Files directory option and click Browse.

   a. Select the following directory:
      `<wbi_install_path>\repository\CollaborationCP`
   
   b. Click OK.

4. Click Finish to import the collaboration template or templates.

You use a collaboration from a Collaboration capacity pack in the same manner as with the default collaboration templates as described in “Using a collaboration template” on page 320.

### 10.3 Designing collaborations

This section describes how to design collaborations and the methodology involved in the process.

#### 10.3.1 Methodology

The methodology you use in projects should contain the following WebSphere Business Integration specific parts for collaborations:

- **Functional overview and scope**
  
  Provide a brief description of the business context in which the collaboration will function, and the business process it supports. Supply a diagram and description of the applications involved and their function in the collaboration.

- **Collaboration logic - scenarios**
  
  This part should describe each triggering event in the collaboration:
  
  - Provide a detailed description of the processing flow for each event. Include the diagrams and subdiagrams whenever possible.
  
  - Include any fields or attributes that are part of the customized scenario.
  
  - Describe any modifications made to the application to support this verb scenario.
  
  - The design should also contain the termination outcomes of the scenario.
Participating objects

Provide a brief description of the objects that are participants in the collaboration:

– Application Specific Business Objects and Generic Business Objects
– Maps
– Static and dynamic relationships
– Other objects

**Tip:** Use the *CollaborationFoundation Template* as the basis for any new collaboration development. This template has been developed to accelerate collaboration development, especially if the new collaboration is a synchronous collaboration. Alternatively use one of the collaboration templates provided with the Collaboration capacity packs.

**Tip:** If there is a need to build wrapper collaborations to handle dependent or related objects, use the *WrapperFoundation Template*. This will accelerate the development of any new wrapper collaborations.

### 10.3.2 Collaboration standards

Consider using standards when designing and naming collaborations. These practices will help make the application design consistent and improve readability and maintainability of the integration solution. Some suggestions are as follows:

**Collaboration template**

– The name of the collaboration template represents the business process.
– Capitalize each word.
– Do not use spaces. This is enforced by the System Manager.
– Avoid underscores, they are used in naming collaboration objects.
– Avoid abbreviations except in long concatenated names.
– For synchronization collaborations, use the *Sync* suffix.
– For wrapper collaborations, use the *Wrapper* suffix.

Examples: CustomerSync, ARInvoiceSync, ContractWrapper.

**Collaboration object**

– The name of the collaboration object should be the template name followed by the application names, separated by underscore.

Example: CustomerSync_SAL301R_to_SAB404R
Ports

- If the collaboration subscribes to a single event, name the Port From. If the collaboration subscribes to multiple events of different Business Objects, use the Generic Business Object name as a suffix.

  Examples: From, FromContract

- Follow the same standard for the To port.

  Examples: To, ToContract

- The name of the port where the collaboration requests retrieval of data from the destination application should be DestinationAppRetrieve.

- The name of the port where the collaboration requests retrieval of data from the source application should be SourceAppRetrieve.

- The name of the port through which a collaboration sends an object to a wrapper collaboration should be To<wrapper template name>, for example, ToContractWrapper.

Collaboration properties

- Property names should be in uppercase, with words separated by underscore. For example, USE RETRIEVE.

- Property values should be in lowercase. For example, true, false, verify.

User-defined variables

- User-defined variables should be declared in the scenario definitions window, not in the collaboration definitions window.

- Be consistent in declaring variables. If you use hungarian notations (for example, i for int in iQuantity), be sure to use it consistently. Either you use hungarian notations, or you don’t.

Miscellaneous

- Tag the top corner of the Main Scenario diagram with a label specifying the version of the collaboration. The version tag should be consistent with that collaboration’s technical specification.

- Indicate the version of the CollaborationFoundation template used to develop the collaboration.

- Use the Copy() method as consistently as possible instead of the Duplicate() method. This means that all things of which copies will be made must be declared and instantiated in the scenario definitions window.
10.4 Developing collaborations

This section describes how to use the Process Designer Express tool to develop collaboration templates and how to create a collaboration object based on a template.

10.4.1 Overview of Process Designer Express

This section provides a brief overview of Process Designer Express.

Opening Process Designer Express

You can open Process Designer Express from the Start menu, the System Manager, or from another tool:

▸ From the Start menu

Select Start → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Process Designer Express

▸ From the System Manager

a. From the tools menu; select Tools → Process Designer (or press Ctrl+2).

b. Expand the Integration Component Library in which you want to create the collaboration template. Right-click the Collaboration Templates folder and select Create New Collaboration Template.

c. Double-click an existing collaboration template, either from the Integration Component Library, or from the user project. The selected collaboration template will open.

▸ From another tool

Select Tools → Process Designer Express (or press Ctrl+2).

Working in Process Designer Express

When you open a collaboration template in Process Designer Express the template tree in the left pane displays the definitions, scenarios and messages in a hierarchical format. If you double-click an element in the template tree, the details will appear in the main window.

There are three windows available:

▸ Template definitions window

Double-click the Definitions icon to open this window, or press Ctrl+T.

This window enables you to set collaboration template properties and comments:
– In the General tab you can specify if the collaboration supports long-lived business processes, the minimum transaction level, and the Java package in which the collaboration resides.
– In the Declarations tab, you define imports and template variables.
– In the Properties tab, you specify user-defined collaboration template properties.
– In the Ports and Triggering Events tab you specify the ports and the triggering events (Business Objects and verbs).

▶ Scenario editor window

Double-click a Scenario icon to open this window. In this window, you create and edit business process activities. You can add:
– nodes, such as start and end nodes, activity nodes, subdiagram nodes, iterator nodes, and decision nodes.
– service calls from an activity node.
– transition links between activities.

▶ Template messages window

Double-click the Messages icon to open this window, or press Ctrl+M. This window enables you to edit the message file. You specify a unique message id, the message with parameters, and an explanation of the message.

Import and export options

In Process Designer Express you have the ability to create collaborations based on Business Process Execution Language (BPEL) and Unified Modelling Language (UML) files. You can also export files in these formats.

▶ BPEL

When a collaboration is exported to BPEL format, the following files are created:
– .bpel - contains the main template information.
– .wsdl - defines the external interfaces to the collaboration.
– .bpelGUI.xml: contains information about the graphical representation of activity diagrams. It is used in situations where BPEL files are imported back into Process Designer Express.

**Note:** All three file types are required if you plan to import BPEL files.
When a collaboration is exported in UML format, a .xmi file is created in XMI 1.1 format.

10.4.2 Developing and deploying collaborations

The steps for developing and deploying a collaboration are:

- Create a collaboration template.
- Specify template property information.
- Define and create scenarios.
- Create a message file.
- Compile the collaboration template.
- Create a collaboration object based on the template.
- Deploy the collaboration template and object.

Creating a collaboration template

Collaboration templates are created using Process Designer Express. When you create a new collaboration template, you are required to provide a unique template name. The collaboration template is stored in an Integration Component Library.

Specifying template property information

Template property information is specified in the Template Definitions window, where you provide the information in the following tabs:

- General
  Specify a description of the collaboration template, if it supports long-lived business processes, the minimum transaction level, and the Java in which the collaboration resides.

- Declarations
  Define template variables and view system-generated template variables.

- Properties
  You can specify user-defined collaboration template properties. User-defined properties have a name, type and a default value.

- Ports and Triggering Events
  Define the Business Object associated with each port. You also specify the triggering events, identified by a port or ports, a Business Object type and a verb.
**Minimum transaction level**

A transaction level for a collaboration can be set to the following values:

- **None**
  
  _None_ means that the collaboration is not transactional. If an exception occurs, it is logged and the collaboration terminates.

- **Minimal Effort**
  
  The collaboration is transactional, and compensation is defined for the scenario subtransactions. If an exception occurs, the collaboration is rolled back and executes compensation for each sub-transactional step.

- **Best Effort**
  
  In addition to compensation, data isolation is used to ensure that data has not changed since its previous use. Best effort isolation checking leaves a small window of time during the isolation check is vulnerable to changes by other application transactions.

- **Stringent**
  
  The application locks the data when isolation is checked, thus removing the window of vulnerability.

---

**Defining template and scenario variables**

Within a collaboration you can set variables on two levels:

- **Template**
  
  Template variables are like class variables in Java. The scope of the variables is all the collaboration scenarios. You set the template variables in the Declarations tab of the template definitions.

- **Scenario**
  
  The scope of a scenario variable is the current scenario and all its subdiagrams. To set scenario variables, right-click the **scenario** and select **Open Scenario Definition**.

Both template and scenario variables are available for use as variables in the Activity Editor.

---

**Defining collaboration configuration properties**

Collaboration templates have two types of configuration properties:

- **Standard**
  
  _Standard_ properties provide information that all collaborations need, such as tracing level and an e-mail address for message notifications. All collaborations have the same standard configuration properties.
Collaboration-specific properties are optional and are defined by a collaboration developer. The collaboration uses the value of the property to determine an aspect of its behavior.

When you specify a collaboration-specific property, you specify the following information:

- **Name**: This should be in uppercase, words separated by underscore.
- **Property type**: Can be a Boolean, Date, Double, Float, Integer, String, Time, or URL.
- **Description**: A description of the property’s purpose.
- **Max Length**: If the type is String, you can specify a maximum length.
- **Max Multiple Values**: You can specify how many values are accepted for the property. This attribute is not often used.
- **Values**: Specify the values and a default value.

Figure 10-6 shows a Boolean property that defines whether or not the collaboration performs an update in the destination application if a create trigger is received and the entity already exists in the destination.

![Template Definitions](image)

Figure 10-6 Sample collaboration property
Ports and triggering events

**Note:** Define ports *before* you develop the scenarios, because you will want to use the ports in the scenarios to perform service calls.

Define the triggering events *after* you develop the scenarios, because you need to define which scenario to run for each triggering event.

In this tab, you define the following information:

- **Ports**
  
  *Ports* are the external interfaces of the collaboration. They are variables that represent Business Objects that the collaboration receives or produces at runtime.

- **Triggering events**
  
  Collaborations subscribe to Business Objects through *triggering events*. You specify which scenario to run for each verb the Business Object supports, potentially none. You do not specify scenarios for ports and Business Object verbs that do not trigger the collaboration, for example, a To-port in a synchronization collaboration.

Defining and creating scenarios

A collaboration template can consist of one or more scenarios. A scenario is the collaboration code that performs certain actions based on an incoming Business Object. The Business Object can represent an event from a connector, or an access call from an access client.

Scenarios consist of execution logic in a main activity diagram and zero or more subdiagrams that are called from the main activity diagram. You can structure a scenario in the same way as you can structure a procedural program. You can have one main scenario that contains all processing logic, or a main scenario that calls subdiagrams (methods) which execute a specific unit of work. For example, you may have a main scenario that evaluates the triggering Business Object verb and calls a Create, Update or Delete subdiagram depending on the verb.

**Note:** A scenario cannot call another scenario, nor a subdiagram of another scenario.
Create a scenario by doing the following:

1. Right-click the **Scenarios** icon and select **New Scenario**.
2. Provide a unique scenario name (unique within the template) and optionally a description of the scenario. Click **OK**.
3. The activity diagram for the scenario is displayed in the main window. By default a start node is available. At this point you create the processing logic. The activity diagram uses Unified Modelling Language (UML) to model the business process. In “Working with activity diagrams” on page 336 we describe how to use activity diagrams.

   **Note:** You must assign at least one triggering event to a scenario. Failure to assign at least one triggering event produces a run time error.

**Creating a message file**

Messages are used to trace and log information, warnings and exceptions. Messages are defined by:

- **Message ID**
  
  This uniquely defines the message within the collaboration. When code is written to trace or log a message, you reference the message by its message id.

- **Message**
  
  A message consists of text and parameters. The code that executes the trace or logging of the message provides values to the parameters at run time. Parameters are defined by a number enclosed within curly brackets, for example {1}, {2} and so on.

- **Explanation**
  
  Provide a description of the message.

You can also provide a description of the message file. Normally the description would contain information about the message number ranges.

   **Tip:** Standardize the message numbers so the message numbers and number ranges are consistent across all collaborations.

Follow these steps to create a message:

1. Double-click the **Messages** icon to open the Template Messages window.
2. Specify the **unique message id**, **message with parameters**, and optionally a **description** of the message.
3. **Close** the Template Messages window.

Figure 10-7 shows the messages for the CollaborationFoundation. Notice the description of the number ranges.

![Figure 10-7 Messages in the CollaborationFoundation template](image)

See “Logging and exception handling” on page 346 on how to work with tracing and logging.

### Compiling the collaboration template

Collaboration templates must be compiled before they can be deployed to an Interchange Server. You must compile the template every time you perform a change in it. If you do not compile the template every time you change it, you will get an error message when you try to deploy it to a server. The following files are created or updated when you compile a collaboration template:

- Java source file: `<collaboration name>.java`, for example, `CustomerSync.java`.
- Executable class file: `<collaboration name>.class`, for example, `CustomerSync.class`. 
Creating a collaboration object

A collaboration consists of two parts, a collaboration template and a collaboration object. The collaboration template contains all the execution logic and the external interfaces (ports), while a collaboration object binds the external interfaces to applications through connectors. Both collaboration templates and objects are deployed to a server, but it is the collaboration object that is the runnable object.

You can create multiple collaboration objects for each template. For example: You might have a CustomerSync collaboration template that synchronizes customers from one application to another. You might then reuse the template by creating one collaboration object that synchronizes customers from application A to application B, and another for synchronizing customers from application B to application A.

When you create a collaboration object you specify the following:

- **Collaboration name**
  A collaboration object name should consist of the template name the object is based on followed by the applications involved, for example, CustomerSync_AppA_to_AppNB.

- **Bind the ports**
  Bind the ports defined in the template to the connectors. The connectors must support the Business Objects that are defined for the triggering events.

- **General properties**
  In the general properties section of a collaboration object, you can specify properties such as the transaction level, tracing level and the maximum number of concurrent events.

Note: At runtime, the collaboration runtime environment locates the appropriate message file for the collaboration locale (inherited from Interchange Server) from the collaborations\messages subdirectory. For example, if the collaboration locale is U.S. English (en_US), the collaboration runtime environment retrieves messages from the <collaboration name>_en_US.txt file.

To compile a template, select **File → Compile**, or press **Ctrl+F7**.
Specify user-defined variable values

Specify the values of any user-defined variables defined on the collaboration template.

When the collaboration object is created, it is displayed in a graphical view. You can also switch to a tree view where you can see the maps used. Figure 10-8 shows the tree view of the collaboration object created in the sample development part of this book.

Figure 10-8  Collaboration object tree view

Deploying a collaboration

When you deploy a collaboration, both the collaboration object and template must be deployed. You will not be able to deploy a collaboration object unless the template is compiled and deployed.

The collaboration can be deployed in two ways:

- A user project

  The collaboration object and template can be added to a user project. As opposed to Integration Component Libraries, user projects can be deployed to an Interchange Server.

- Directly to an Interchange Server

  You can deploy a collaboration object and template to an Interchange Server without going through a user project. Drag the components from the ICL project to the Interchange Server in the Interchange Server Component Management outline of the System Manager.

When you deploy a collaboration, you must restart the Interchange Server. If you are deploying an existing collaboration object, you must stop the collaboration object on the server before you deploy the updates. When the Interchange Server is restarted, you can start the collaboration object again.
10.4.3 Working with activity diagrams

In this chapter we will describe the various components in an activity diagram, and how to create and use them. The components we will cover are:

- Start nodes represent the start of the processing flow.
- Action nodes represent an action to be performed.
- Transition links represent the control flow between nodes.
- Decision nodes represent branching of the processing flow based on a condition.
- Service calls represent a request to or response from an external interface.
- Subdiagrams are separate discrete units of logic within a main diagram.
- Iterators are specialized subdiagrams that implement loops or iterations.
- End nodes terminate the processing flow, either successfully or with a failure.

Figure 10-9 shows the action buttons in Process Designer Express.

![Actions in Process Designer Express](image)

**Start nodes**

Start nodes define the start of processing flow in the activity diagram. It is automatically created when you create a scenario or a subdiagram. You must only have one start node in an activity diagram.

The start node label of a scenario is automatically set to the scenario name. The start node label of a subdiagram is SubDiagram_<unique id>.

**Action nodes**

An action node represents a step in a collaboration. An action node has a label, description and optionally operations through code, such as:

- Getting and setting the attribute values in Business Objects
- Checking the verb in an incoming event
- Comparing attribute values to constants or to other attribute values
Creating an action node
To create an action node, do the following:

1. Right-click the activity diagram and select action node.
2. Place the action node by clicking the desired location in the activity diagram.
3. Double-click the action node to open its properties.
4. Provide a label and a description of the action.
5. Click Edit to build the action code.
6. When you have built the code, click Apply and Close to save it.

Building code
You can build code in two ways:

- Use the Activity Editor

  The Activity Editor is a tool for visually producing code through function blocks. When you save the activity, it generates Java code you can edit.

  To edit the Java code generated by the Activity Editor, click the Java tab and select Tools → Edit code.

  **Note:** If you edit the Java code that the Activity Editor generates, you will not be able to return to the graphical view. You will also not be able to migrate the code to future releases of WebSphere Business Integration Server Express.

In “Developing an example” on page 348 we will use the Activity Editor for certain functions. For details on how to use the Activity Editor, refer to the online help.

- Write Java code

  You can write Java code yourself. Click the Java tab in the Activity Editor and select Tools → Edit Code to open the editor.

  WebSphere Business Integration Server Express provides an API for programming against the different components within a collaboration.

  **Note:** If you write custom Java code you will not be able to migrate the code to future releases of WebSphere Business Integration Server Express.
In “Developing an example” on page 348 we will describe how to write Java code to raise an exception in a collaboration. For details on the collaboration API, refer to the Online Help.

**Transition links**

Transition links connect nodes and specify the control flow of the activity diagram. Transition links can have a label and a description.

**Rules for transition links**

The following rules apply for transition links to and from the different node types:

- Action nodes can have an unlimited number of incoming transition links, and only one outgoing links.
- Decision nodes can only have one incoming transition link and seven outgoing links.
- Subdiagram and iterator nodes may have an unlimited number of incoming transition links and only one outgoing link.

Transition links can be either *orthogonal* or *free-form*. The difference between these links are purely graphical.

**Creating transition links**

Follow these steps to create a transition link:

1. Right-click the activity diagram and select **Transition Link** or **Transition Link (Orthogonal)**.
2. Click the bottom edge of the node where you want the link to start. If the mouse pointer does not look like a circle with a cross in it, you are not allowed to link from the node.
3. Click the top edge of the node where you want the link to end. If the mouse pointer does not look like a circle with a cross in it, you are not allowed to link to the node.

**Note:** The transition link should be blue. If it is red, a transition link is missing preceding the red transition link.

**Business Object probes**

Business Object probes monitor attribute values of a Business Object at runtime. A probe is placed on a transition link and is activated or deactivated through the Collaboration Properties in System Manager.
Follow these steps to specify a Business Object probe:

1. Right-click the transition link you want to put a probe on and select **Properties**.
2. Select the **Enable Business Object Probe** check box.
3. Expand the **business object** and select the **attributes** you want to monitor.
4. Click **Apply**.
5. Use the System Manager to enable or disable the probe.

The following rules apply to Business Object probes:

- They cannot be used on the incoming transition link for a decision node.
- They cannot be used on service call links.
- They cannot probe into N-cardinality child Business Objects.

**Decision nodes**
Decision nodes are like *if...else* statements. Using a decision node you can branch to more than one action based on a set of conditions. The simplest form of a decision node is displayed in Figure 10-10. It states that if the triggering Business Object verb is Create, branch to the Create action node that performs the service call to the destination application. If it is not Create, continue processing.

**Branch types**
There are three types of branches in a decision node:

- **Normal**
  
  A normal branch has a condition associated with it. If that condition is met, the branch is taken. You can have multiple normal branches. By default, normal branches are represented by a blue square.

---

**Figure 10-10  Sample decision node**

![Sample decision node diagram](image-url)
Exception

An exception branch has a specific exception type associated with it. The condition of an exception branch tests that the system variable current exception is equal to the exception type to which you set the branch. You can have multiple exception branches. By default, exception branches are represented by a red square.

Default

The default branch is taken when none of the other branch conditions are true. Each decision node can have one, and only one, default branch. This branch is optional. By default, it is represented by a black square.

Note: A decision node can have a maximum of seven branches.

Creating a decision node

Follow these steps to create a decision node:

1. Right-click the activity diagram and select Decision Node.
2. Place the decision node by clicking the desired location in the diagram.
3. Double-click it to open the decision node properties.
4. Provide the label and description of the decision node.
5. For each branch, specify the following:
   a. The type, either Normal, Exception, or Default
   b. The condition
      • If the type is Normal, you specify the condition writing Java code, for example, isCreate == true. The easiest way to do this is to use the Condition Builder. It will display the available variables. Once you select the variable, you can select the operator and the condition.
      • If the type is Exception, select the exception class from in the condition column.
      • If the type is Default, do not specify a condition.
   c. The branch label. This is the text that should be applied to the transition link that connects the branch to a node.
6. Click Apply and Close.
7. Create transition links from the decision node branches to the desired nodes.

Service calls

An action node can only perform a service call by attaching a service node to it. The action generates the service call and handles the service call results on its outgoing transition links. An action and a service call work as a pair; the service call performs the remote input/output function. Following a service call, there
should always be a check if an exception has occurred. You do this by creating a decision node.

Service calls can either be synchronous or asynchronous. Service calls are discussed in more detail in “Service calls” on page 316.

**Creating a service call**

To create a service call, do the following:

1. Right-click the **action node** you want to associate the service call with and select **Add service node**. Notice that a synchronous service call has been added to the right of the action node.

2. Double-click the **service node** to open the properties for it.

3. Provide a description of the service call.

4. Specify the service call information:
   a. The port to which the service call is sent
   b. The verb of the Business Object
   c. The variable that contains the Business Object to send

5. Specify the compensation information. This only applies to transactional steps, such as Create, Update and Delete. The normal types of compensation are:
   - If the service call is a Create, the compensation is a Delete.
   - If the service call is a Delete, the compensation is a Create.
   - If the service call is an Update, the compensation is an Update restoring the original values.

6. Optionally, click the **Advanced** button to set the following options:
   a. The service call type, synchronous or asynchronous.
   b. The time-out values to be used with inbound service calls.
   c. The correlation set used for attribute-matching for outbound service calls.
   d. Click **OK** to apply the settings.

7. Click **Apply** and **Close** to save the service call settings.

8. Create a decision node that evaluates the result of the service call. A service call can result in a ServiceCallException. The decision node should have two branches, one Exception branch, and one Normal branch. The Exception branch should link to an action node that logs the exception and ends the collaboration with a failure. The Normal branch should link to an action node that traces the result of the service call.

Figure 10-11 on page 342 shows an example processing logic of a service call.
Figure 10-11 Example of a service call

Note: An action node can only have one service call associated with it.

Tip: Upon return of a service call, the Business Object variable contains the result of the call. The data in the original Business Object is lost if the service call retrieves new data for the Business Object. Therefore, if you anticipate needing the values in the original, it is useful to copy the original Business Object into a temporary variable in an action that calls the service call.

Subdiagrams
Subdiagrams are used to separate discrete units of logic from the main diagram. You may have one subdiagram for each verb the main diagram handles, for example, you can have a Delete, Create, Update, and Retrieve subdiagram.

You work with subdiagrams in the same way as you work with the main diagram. Notice that when a subdiagram completes successfully it returns to the parent diagram. When an exception occurs, the exception can be passed to the parent diagram.

Creating a subdiagram
You create a subdiagram from within a main diagram or another subdiagram in the following way:

1. Right-click the activity diagram and select Subdiagram Node.
2. Place the subdiagram node by clicking at the desired location in the activity diagram.
3. Double-click the subdiagram node to open the properties for it.
4. Provide a label and a description and click Apply and Close.
5. Right-click the node and select Open Subdiagram to open the activity diagram for the subdiagram.

A subdiagram node is usually followed by a decision node that verifies the result from the subdiagram execution, to handle any exceptions the subdiagram might have raised, for example. However, in some cases you may want to handle the exception within the subdiagram and end it successfully.

**Important:** Whenever a subdiagram ends with the End Failure node, the collaboration runtime environment terminates the entire collaboration. This means that the parent diagram may perform actions following a failed subdiagram, but it cannot result in an End Success node.

Figure 10-12 shows an example of handling the result from a subdiagram node.

![Example of handling the result of a subdiagram node](image)

**Iterators**

An *iterator* is a specialized form of a subdiagram that implements looping and iterations. In an iterator node you can:

- Iterate through all the attributes of a Business Object.
- Iterate through all elements within an array of Business Objects.
- Loop through a defined variable.
**Creating an iterator**

To create an iterator, do the following:

1. Right-click the **activity diagram** and select **Iterator Node**.
2. Place the iterator node by clicking the desired **location** in the activity diagram.
3. Double-click the **iterator** node to open the properties for it.
4. Provide a label and a description.
5. Specify the variable that will hold the item being processed during iteration.
6. Specify the iterator type:
   a. The variable holding the Business Object, if it is an iterator that iterates through the attributes of a Business Object.
   b. The variable holding the array of Business Objects, if it is an iterator that iterates through the elements of a Business Object array.
   c. If it is a loop:
      i. Provide the initial value of the counter variable.
      ii. Specify the condition that must be true in order for the loop to execute.
      iii. Specify the method of incrementing the value of the counter variable
7. Click **Apply** and **Close**.
8. Right-click the **node** and select **Open Subdiagram** to open the activity diagram of the iterator.

**Note:** A break can be added to an iterator's activity diagram to force premature termination of the iteration. When the iterator's execution path reaches the break symbol, the iterator terminates and control is passed back to the parent diagram.

Figure 10-13 on page 345 shows the different options in the iterator properties.
End nodes
You can terminate an activity diagram in two ways:

- Success
  Ending a collaboration in success means that the triggering event was handled successfully. An exception may have occurred, but in that case the activity diagram handled it.

- Failure
  Ending a collaboration in failure means that the activity diagram was unable to execute properly.

Creating end nodes
To create an end success node, or an end failure node, follow these steps:

1. Right-click the activity diagram and select End Success Node or End Failure Node.
2. Place the node by clicking the desired location in the activity diagram.

10.4.4 Logging and exception handling

In this section we discuss some best practices on handling exceptions and logging in collaborations.

Logging
Consider using standards when implement logging in your collaborations. These practices will help make the application design consistent and improve readability and maintainability of the integration solution. Some suggestions are as follows:

- Error condition: use the raiseException() method instead of the logError() method.
- Warning: Use the trace() method instead of the logWarning() method to raise a warning when the collaboration logic makes a decision on behalf of the user.
- Information: Use the trace() method instead of the logInfo() method to print out an information message.

Exception handling
Exceptions are handled as specified in Example 10-1.

Example 10-1  Collaboration exception handling

```
Line 1: String sExceptionType = currentException.getType();
Line 2: String sExceptionMessage = currentException.getMessage();
Line 3: raiseException(
  Line 4:   ServiceCallException,
  Line 5:   4000,
  Line 6:   ToBusObj.getType(),
  Line 7:   ToBusObj.getVerb(),
  Line 8:   ToBusObj.keysToString(),
  Line 9:   sExceptionType,
  Line 10:  sMessage);
```

1. Before calling the raiseException() method call, set the variables sExceptionType and sMessage (lines 1 and 2).
2. Specify the type of exception in the raiseException() method call, for example, ServiceCallException (line 4).
3. Specify the numeric message number for the error (line 5). The message number must correspond to a message specified in the collaboration Messages window.
4. Specify the parameters for the message in the exact order as they occur in the message text from the file. Message parameters are specified as numbers within curly brackets: \{1\}, \{2\} and so on.

The message with id 4000 in the Collaboration Template is:

Collaboration Failed: \{1\}.{\{2\} with keys \{\{3\}\} synchronization failed and the exception is \{4\}.{\{5\}.

It takes five parameters:

- Business Object type (line 6), for example, Employee, Item, or Customer.
- Business Object verb (line 7), for example Create, Update, or Delete.
- Generic Business Object keys (line 8), for example, 1423. The keysToString() method will also print the key attribute name.
- Application-specific exception type caught by the Interchange Server (line 9).
- Application-specific exception message caught by the Interchange server (line 10).

**Note:** The raiseException() method is limited by a maximum of five parameters for the exception message.

**Note:** You can also raise exceptions using the General\Utilities\Raise Error Type and General\Utilities\Raise Error Message function blocks. However, these function blocks do not let you specify error messages and parameters.

**Tip:** Standardize error messages so that the message numbers and number ranges are consistently used across all collaborations.

**Propagating an exception**

When an exception occurs, you must raise the exception up the chain of collaborations to the main collaboration. The way errors are propagated up to the main scenario of the collaboration is by:

- Including a raiseException(currentException) statement in every Failure action nodes at the top levels of all the subdiagrams.
- Including the raiseException(sExceptionType, sMessage) statement in the Failure action nodes of the Main scenario diagram.

**Tracing**

Tracing is performed as specified in Example 10-2 on page 348.
Example 10-2  Collaboration tracing

Line 1: trace(
Line 2: 1,
Line 3: 2000,
Line 4: processingBusObj.getType(),
Line 5: processingBusObj.getVerb(),
Line 6: processingBusObj.keysToString());

1. The first parameter to the trace() method is the trace level that causes the message to be generated (line 2). If the trace level on the collaboration is lower than the one specified in the method, the message will not be generated.

2. The next parameter (line 3) is the message id of the trace message specified in the collaboration Messages window.

3. The following parameters in lines 4 through 6) are parameters to the specified message.

Note: The trace() method above is limited by a maximum of 5 parameters for the exception message. However, there is also a trace method that takes an array of parameters. This will let you specify more parameters. The signature for this method is:

```java
void trace(int traceLevel, int messageNum, Object[] paramArray)
```

Note: You can also trace using the function blocks located in General\Logging and Tracing. However, these function blocks do not let you specify more than 3 parameters for the trace message.

10.5 Developing an example

In this section we will describe how to develop and test a collaboration template that synchronizes customer information between two imaginary applications. We are basing the collaboration on the Business Objects specified in Chapter 8, “Business Objects” on page 209, which uses the maps specified in Chapter 9, “Mapping” on page 247.

We will simulate the two applications by using the SourceConnector and DestinationConnector supplied with WebSphere Business Integration Server Express.
In this example we will perform the following tasks:

- Create the collaboration template. The collaboration template name will be MyCustomerSync.
- Specify collaboration template definitions. We will specify the variables and ports we will use in the scenarios.
- Specify messages. We will create informational messages, warning messages for unsupported actions, and error messages for exceptions.
- Create a scenario. The collaboration will contain one main scenario and to subdiagrams, one for each supported verb.
- Edit template definitions. We specify which scenario to run for each triggering event.
- Modify connectors. The collaboration will be tested using the SourceConnector and DestinationConnector. These connectors need to be modified to add agent support for the Business Objects we have created.
- Create the collaboration object. We will create a collaboration object based on the template, and bind the ports to the modified SourceConnector and DestinationConnector.
- Deploy the collaboration. We will deploy the collaboration to the Interchange Server using the GettingStartedICS user project.
- Test the collaboration. In WebSphere Business Integration Server Express you can use the Test Connector tool to simulate applications. We will test the collaboration using two Test Connectors, one for the SourceConnector, and one for the DestinationConnector.

10.5.1 Creating the collaboration template

Follow these steps to create the collaboration template:

1. Right-click the **Collaboration Templates** folder in the GettingStartedICL project and select **Create New Collaboration Template**.
2. Fill in the following details in the New Template dialog box:
   - **Template Name**: MyCustomerSync
   - **Description**: Synchronizes customer data between source and destination application.
3. Click **OK**.

**Note**: The example collaboration is only meant to show how to use Process Designer Express, and various tasks within the tool.
10.5.2 Specifying collaboration template definitions

Now we need to provide more details in our new template. We must provide definitions of the ports and events that our template uses.

1. Double-click the Definitions icon in the left pane. The Template Definitions window appears in the right pane.

2. Select the Ports and Triggering Events tab, and add the following ports:
   a. SourceApp
      - Port: SourceApp
      - BO Type: Customer
   b. DestinationApp
      - Port: DestinationApp
      - BO Type: Customer

   We will add the scenarios to the triggering events later.

3. Click Apply.

4. Click Close.

10.5.3 Specifying messages

Now we define the message that our template will use.

1. Double-click the Messages icon in the left pane. The Template Messages window appears in the right pane.

2. Add the information message:
   - Message id: 3000
   - Message: Collaboration Succeeded: {1}.{2} with keys ({3}) synchronization successful.
   - Explanation: The triggering Business Object was successfully synchronized in the destination.

3. Add the warning message:
   - Message id: 5000
   - Message: {1} is an unsupported verb.
   - Explanation: The verb is not supported.

4. Add the error message:
   - Message id: 8000
   - Message: Collaboration Failed: {1}.{2} with keys ({3}). Exception type is: {4}. Exception message is: {5}.
Chapter 10. Collaborations

5. Add the following description for the message text file:

Trace messages, range 3000 - 4999
Warning messages, range 5000 - 6999
Error messages, range 8000 -

6. **Close** the Template Messages window.

**Note:** {1}, {2} and so forth are parameters to the message. At runtime these variables are replaced with values. For example for message 3000, {1} will be replaced with Customer.Update. The following message will appear in the log:

Customer.Update is an unsupported verb.

### 10.5.4 Creating a scenario

In this section we will create the following:

- A main scenario
- A subdiagram for each supported verb, Create and Delete.

Figure 10-14 on page 352 shows how the main scenario should look.
Creating a main scenario
To create a main scenario, do the following:

1. Right-click the Scenarios icon in the left pane and select New Scenario.
2. Specify the following information:
   ▶ Scenario Name: Main
   ▶ Description: Synchronizes Create and Delete from source to destination.

3. Click OK.

4. Right-click the scenario and select the Open Scenario Definition. Specify the following variables:
   a. **bCreate**
      - Type: boolean
      - Name: bCreate
      - Initial value: false.
   b. **bDelete**
      - Type: boolean
      - Name: bDelete
      - Initial value: false.

We will set the variables to determine which verb (operation) the triggering Business Object has. Based on the value of these variables we will either perform a create or delete on the destination application. If we receive any other verb we will add a warning to the log.

Figure 10-15 on page 354 shows the scenario definition and the specified variables.

5. Click **Update** and **Apply**.
In this section we will create an action node that performs the following operation:

- Get the verb from the triggering Business Object
- Checks that the verb is equal to Create
- Sets the value of the bCreate variable.

To create a bCreate template variable, do the following:

1. Right-click within the Scenario window and select **Action Node**. Click below the start node to place the action node.

2. Double-click the **action node** to open properties box. Specify the following information:

---

**Tip:** Select the following view settings to enhance readability of the diagram:

- View UIDs: **off**
- View Labels: **on**

These settings are available from the View menu.

---

**Figure 10-15** Definition of variables on the scenario level
– Label: Verb=Create?
– Description: Sets the bCreate template variable

3. Click Apply.

4. Click the Edit... button to open the Activity Editor.

5. Drag the following components from the Library outline to the Graphical window:
   – Variables\triggeringBusObj (BusObj)
   – General\APIs\Business Object\Get Verb
   – General\String\Text Equal Ignore Case
   – Variables\bCreate (boolean)

6. Right-click within the Graphical window and select New Constant.

7. Click within the yellow box and type create.

8. Connect the entities as displayed in Figure 10-16. For example, drag the right arrow from triggeringBusObj to the left arrow of Get Verb.


10. Select File → Close.

11. Observe the generated Java code in the Action Properties window.

12. Click Apply and Close.

![Figure 10-16 Check if verb is Create](attachment:image)

**Creating a Transition Link**

Transition Links connect actions, decisions, subdiagrams and iterators. They represent the flow of the diagram. The flow is always from top to bottom.

To create a Transition Link from the start node to the Verb=Create? action node, do the following:

1. Right-click within the Scenario window and select Transition Link.

2. Click the bottom of the start node.
3. Drag the mouse to the Verb=Create? action node and click the top of it. A blue arrow should now connect the start node with the action node.

**Creating a decision node for Create**
In this section, we create a decision node to determine which action to take based on the value of bCreate.

To create a decision node for Create, do the following:

1. Right-click within the Scenario window and select **Decision Node**. Click below the previously created action node to place the decision node.
2. Double-click the **decision node** to open the properties box.
3. Add the following information:
   - **Label:** Create?
   - **Description:** Run SyncCreate if bCreate is true
4. Select the box numbered 2 in the list and enter the following information:
   - **Type:** Normal
   - **Condition:** bCreate
   - **Branch Label:** true
5. Select the box numbered 3 in the list and enter the following information:
   - **Type:** Normal
   - **Condition:** !bCreate
   - **Branch Label:** false
6. Click **Apply** and **Close**.
7. Create a Transition Link from the Verb=Create? action node to the newly created decision node.

The Decision Node specifies that the action sequence to the right is executed if bCreate is true, while the action sequence below is executed if bCreate is false.

**Creating subdiagram for Create**
In this step we will only create an empty subdiagram for the Create action sequence.

1. Right-click within the Scenario window and select **Subdiagram Node**. Click to the right of the previously created decision node to place the subdiagram node.
2. Double-click the **subdiagram node** to open the properties box.
3. Add the following information:
   - **Label:** Create
Description: Creates the Business Object on the destination application

4. Click **Apply** and **Close**.

5. Create a Transition Link from the Create? decision node to the subdiagram. Use the *Transition Link (Orthogonal)* instead of the ordinary Transition Link. Observe that the link is labeled *true*. This relates to the branch label specified in the decision node.

**Setting the bDelete template variable**
Create an action node below the previously created Decision node and perform the same steps as in “Setting the bCreate template variable” on page 354, with some differences. Replace the following steps:

- **Step 2**: set the label to **Verb=Delete?**, and the **description** to Sets the bDelete template variable.
- **Step 5**: use **Variables\bDelete (boolean)** instead of **Variables\bCreate (boolean)**.
- **Step 7**: set the constant value to **delete** instead of **create**.
- Create a Transition Link between the decision node and the action node.

The action should be as displayed in Figure 10-17.

![Figure 10-17 Check if verb is Delete](image)

**Creating a decision node for Delete**
Create a new decision node below the previously created action node and connect the two with a Transition Link. Follow the same steps as in “Creating a decision node for Create” on page 356.

Replace all instances of create with delete. Figure 10-18 on page 358 displays the properties of the decision node.
Creating a subdiagram for Delete
Create a new subdiagram and place it to the right of the previously created Decision Node. Fill in the following properties for the subdiagram:

- **Label:** Delete
- **Description:** Deletes the Business Object on the destination application

Connect the Decision Node and the subdiagram with a Transition Link (Orthogonal).

Creating a warning for an unsupported verb
If the triggering verb is neither Create nor Delete, we will write a warning message to the log. The warning message will be in the following format:

`BusinessObject.Verb is not supported`
We have already created a message (5000) which enables us to print this message.

1. Create a new action node below the previously created decision node.

2. Double-click the **action node** and set the following properties:
   - **Label**: Unsupported verb! (5000)
   - **Description**: Print warning to log if verb is not Create or Delete

3. Click the **Edit...** button to open the Activity Editor.

4. Drag the following components from the Library outline to the Graphical window:
   - Variables\triggeringBusObj (BusObj)
   - General\APIs\Business Object\Get Business Object Type
   - General\APIs\Business Object\Get Verb
   - 2 instances of General\String\Append Text
   - General\Logging and Tracing\Log Warning\Log Warning ID 1

   We will be using Log Warning ID 1 since we only have one parameter to the message.

5. Create the following constants:
   - Create a . (period). We will use this to bind the Business Object type and verb, for example, Customer.Create.
   - Create a message id of 5000, the message we want to log.

6. Connect the entities as displayed in Figure 10-19 on page 360.

7. Select **File → Save**.

8. Select **File → Close**.


10. Click **Apply** and **Close**.

11. Create a Transition Link between the previously created decision node and the action node.
Creating an end success node

If the Business Object verb is not supported we only want to log a warning. We do not want the collaboration to fail. We will therefore create a Transition Link from the warning action node to an end success node. Perform the following steps:

1. Right-click within the Scenario Window and select **End Success Node**.
2. Click below the warning action node to place it.
3. Double-click the **end success node** to set these properties:
   - **Label**: End with warning
   - **Description**: Collaboration successfully ended but with warning
4. Click **Apply** and **Close**.
5. Create a Transition Link between the warning action node and the end success node.

Creating a flow following Create

When we perform a create on the destination application there are two possible outcomes:

- The service call completed successfully. We trace that the collaboration ended successfully.
- The service call failed. We log the exception and end the collaboration with a failure.

To create this logic we will perform the following tasks:

1. Create a decision node that determines if a `ServiceCallException` has occurred. This is the exception we receive if the service call could not be executed on the destination application.
2. Create an action node that logs the exception if the exception occurs.
3. Create an end failure node if the exception occurs.
4. Create an action node that traces that collaboration has ended successfully if we do not receive an exception.
5. Create an end success node if we do not receive an exception.

**Creating a decision node**

Follow these steps to create a decision node that determines if a ServiceCallException has occurred:

1. Right-click within the Scenario Window and select **decision node**.
2. Place the decision node below the Create subdiagram.
3. Double-click the **decision node** to open the properties box.
4. Type the following information:
   - **Label:** Exception?
   - **Description:** Has a ServiceCallException occurred?
5. Select the **box numbered 2** in the list and enter the following information:
   - **Type:** Exception
   - **Condition:** ServiceCallException
   - **Branch Label:** ServiceCallException
6. Select the **box numbered 3** in the list and enter the following information:
   - **Type:** Normal
   - **Branch Label:** success
7. Click **Apply** and **Close**.
8. Create a Transition Link from the Create subdiagram to the newly created decision node.

Figure 10-20 on page 362 shows the properties of the decision node.
Creating an action node for exception logging

In this example we will write the Java code for logging ourselves.

Follow these steps to create an action node that logs the exception:
1. Create a new action node below and to the right of the Delete subdiagram.
2. Double-click the action node and apply the following properties:
   - Label: ServiceCallException (8000)
   - Description: Log the ServiceCallException
3. Click the Edit... button to open the Activity Editor.
4. Click the tab labeled Java in the main window.
6. Type in the Java code as listed in Example 10-3 on page 363.
7. Select File → Save.
8. Click **Yes** to continue.
9. Select **File → Close**.
10. Click **Apply** and **Close**.
11. Create a Transition Link from the exception decision node to the action node.

**Example 10-3  Java code for logging exception**

```java
// Get exception information
String exceptionMessage = currentException.getMessage();
String exceptionType = currentException.getType();

// Log that the collaboration failed.
raiseException(
    ServiceCallException, 8000,
    triggeringBusObj.getType(),
    triggeringBusObj.getVerb(),
    triggeringBusObj.keysToString(),
    exceptionType,
    exceptionMessage);
```

**Creating an end failure node**

If an exception occurs, we log the exception and end the collaboration with a failure. This means that an administrator must use the Flow manager to determine why the collaboration has failed.

Follow these steps to create the end failure node:

1. Right-Click in the Scenario window and select **End Failure Node**.
2. Create a Transition Link from the exception action node to the end failure node.

**Creating an action node for tracing**

We will now create an action node which traces that the collaboration has ended successfully. Perform the following steps:

1. Create a new action node below the exception decision node.
2. Double-click the **action node** and set the following properties:
   - **Label**: Trace Success (3000)
   - **Description**: Trace that the collaboration ended successfully

3. Click the **Edit** button to open the Activity Editor.

4. Drag the following components from the Library outline to the Graphical window:
   - Variables\triggeringBusObj (BusObj)
   - General\APIs\Business Object\Get Business Object Type
   - General\APIs\Business Object\Get Verb
   - General\APIs\Business Object\Key To String
   - General\Logging and Tracing\Trace\Trace ID 3

   We are using Trace ID 3 because there are three parameters to the message.

5. Create the following constants:
   - Set the tracing level to **5**. This specifies that we will only trace the message if the tracing level is set to 5.
   - Make the message id **3000**. This is the message id of the message we want to trace.

6. Connect the entities as displayed in Figure 10-21 on page 365.

7. Select **File** → **Save**.

8. Select **File** → **Close**.

9. Click **Apply** and **Close**.

10. Create a Transition Link between the previously created decision node and the action node.
Figure 10-21  Trace that collaboration has ended successfully

**Creating an end success node**

If an exception does not occur, we trace that the collaboration ended successfully and end the activity sequence with an end success node.

Follow these steps to create the end success node:

1. Right-Click in the **Scenario** window and select **End Success Node**.
2. Create a Transition Link from the trace action node to the end success node.

**Creating a flow following Delete**

As with the Create subdiagram, the Delete subdiagram can also return either a success or a failure. We will therefore create the exact same sequence of actions following a Delete as with the Create. Perform the same tasks as specified in “Creating a flow following Create” on page 360.

**Tip:** You can copy and paste nodes using either **Edit → Copy** and **Edit → Paste**, or **Ctrl+C** and **Ctrl+V**.

**Editing the Create subdiagram**

At this point we have created the main flow of the collaboration, but, we have not performed any synchronization to the destination application yet. We have
created two subdiagrams that will perform the service calls to the destination application; Create and Delete.

The subdiagrams will perform the service calls to the destination application, as displayed in Figure 10-22.

Follow these steps to create the Create subdiagram:
1. Open the Create subdiagram by double-clicking it in the left window.
2. Create a new action node with label Perform Create.
3. Right-click the action node and select Add service node.
4. Double-click the service node and enter the following property information:
   - Port: DestinationApp
   - Verb: Create
   - BO variable: triggeringBusObj
5. Click Apply and Close.
6. Create a decision node that evaluates if a ServiceCallException has occurred. If a ServiceCallException has occurred, end the subdiagram with an end failure node.

**Editing the Delete subdiagram**

Create the Delete subdiagram by following the same six steps as in “Editing the Create subdiagram” on page 365, replacing the verb Create with Delete.
10.5.5 Editing template definitions

In “Specifying collaboration template definitions” on page 350 we set the template definitions, but we did not specify the scenarios for the triggering events, because the scenarios were not created yet.

In this step we will add the Main scenario to the triggering events:
1. Double-click the Definitions icon to open the Template Definitions window.
2. Select the Ports and Triggering Events tab.
3. Select the Main scenario for all the events in the SourceApp port.
4. Click Apply and Close.

Figure 10-23 shows the Ports and Triggering Events tab for the collaboration template.

![Figure 10-23 Ports and triggering events for MyCustomerSync_Source_to_Destination](image)

10.5.6 Modifying connectors

In this chapter we will modify the SourceConnector and DestinationConnector supplied with WebSphere Business Integration Server Express, using these to simulate our source and destination applications.

By default the SourceConnector and DestinationConnector only support the Customer Generic Business Object. We need to add support for the Business Objects we created in the example development in Chapter 8, “Business Objects” on page 209.
Modifying SourceConnector
Perform the following steps to modify the SourceConnector:

1. Copy the SourceConnector from the WBIExpressLibrary Integration Component Library (ICL) to the GettingStartedICL project.
2. Double-click the SourceConnector to open it in Connector Configurator Express.
3. Select the Supported Business Objects tab.
4. Remove agent support for the Customer Generic Business Object. In our example we will use an Application Specific Business Object (MyAppCust) that maps to the Customer Generic Business Object.
5. In line number 2, select the MyAppCust Application Specific Business Object and enable agent support.
6. Select File → Save → To project to save the connector.
7. Select File → Close to close Connector Configurator Express.

Modifying DestinationConnector
Perform the following steps to modify the DestinationConnector:

1. Copy the DestinationConnector from the WBIExpressLibrary Integration Component Library (ICL) to the GettingStartedICL project.
2. Double-click the DestinationConnector to open it in Connector Configurator Express.
3. Select the Supported Business Objects tab.
4. Remove agent support for the Customer Generic Business Object. In our example the Customer Generic Business Object will map to the SAL301RUSR Application Specific Business Object.
5. In line number 2, select the SAL301RUSR Application Specific Business Object and enable agent support.
6. Select File → Save → To project to save the connector.
7. Select File → Close to close Connector Configurator Express.

10.5.7 Creating the collaboration object
We have created the collaboration template and added support for our Business Objects to the SourceConnector and DestinationConnector. At this point, we will create the collaboration object based on our template, and bind the ports to our connectors.

Create the collaboration object by doing the following:
1. Right-click the **Collaboration Objects** folder in the GettingStartedICL project and select **Create New Collaboration Object**.

2. Select the **MyCustomerSync** collaboration template and provide the following collaboration object name:
   MyCustomerSync_Source_to_Destination.

3. Click **Next**.

4. Select the **SourceConnector** in the BindWith column to bind it with the SourceApp port.

5. Select the **DestinationConnector** in the BindWith column to bind it with the DestinationApp port.

6. Click **Next**.

7. Select the following options:
   – System Trace Level: 5 - And detailed message contents
   – Collaboration Trace Level: 5

8. Click **Finish**.

The collaboration object should appear as in Figure 10-24.

![Figure 10-24 MyCustomerSync_Source_to_Destination collaboration object](image)

### 10.5.8 Deploying the collaboration and its dependent objects

To review what we have done before we deploy the collaboration, we have performed the following tasks:

- Created the following Business Objects in Chapter 8, “Business Objects” on page 209:
  - Customer Generic Business Object which uses the packaged Address Business Object.
  - MyAppCust Application Specific Business Object which represents the customer data in the source application.
  - SAL301RUSR Application Specific Business Object which represents customer data in the destination application. The SAL301URUSR Business Object has a SAL301RUSR_ADDR child Business Object which represents a customer's address data.
Created the following maps in Chapter 9, “Mapping” on page 247:

- MyAppCust_Customer, which maps data from the MyAppCust Application Specific Business Object to the Customer Generic Business Object.

- Customer_SAL301R_USR, which maps data from the Customer Generic Business Object to the SAL301RUSR Application Specific Business Object. This map uses the Sub_Address_SAL301R_ADDR submap to map between the Address Generic Business Object to the SAL301RUSR_ADDR Application Specific Business Object.

- SAL301R_USR_Customer which maps data from the SAL301RUSR Application Specific Business Object to the Customer Generic Business Object. This map uses the Sub_SAL301R_ADDR_Address submap to map between the SAL301RUSR_ADDR Application Specific Business Object to the Address Generic Business Object.

Created the MyCustomerSync collaboration template that executes the synchronization of customer data from our source application to our destination application. For example purposes the template only supports the Create and Delete verbs.

Modified the SourceConnector and DestinationConnector to support the Application Specific Business Objects we created.

Created the MyCustomerSync_Source_to_Destination collaboration object and bound the SourceApp port to the SourceConnector and the DestinationApp to the DestinationConnector.

Creating the user project

We will first create a user project that we can deploy to the Interchange Server.

1. Select File → New → User Project from the System Manager menu.

2. Type GettingStartedICS as the project name.

3. Expand GettingStartedICL in the Available Integration Component Libraries and select the following objects:

   - Business Objects\Address
   - Business Objects\Customer
   - Business Objects\MyAppCust
   - Business Objects\SAL301RUSR
   - Business Objects\SAL301RUSR_ADDR
   - Collaboration Objects\MyCustomerSync_Source_to_Destination
   - Collaboration Templates\MyCustomerSync
   - Connectors\DestinationConnector
   - Connectors\SourceConnector
   - Maps\Customer_SAL301R_USR
   - Maps\MyAppCust_Customer
4. Click Finish.

The user project structure will look like Figure 10-25.

![Figure 10-25 GettingStartedICS user project structure](attachment:image.png)

**Deploying the user project**

In contrast to an Integration Component Library, a user project can be deployed to an Interchange Server.

**Note:** The Interchange Server must be running and the System Manager must be connected to it before you can deploy to the server.

**Tip:** Although an Integration Component Library cannot be deployed to an Interchange Server, you can deploy single objects within one to a server by dragging the objects from the project to the server.
To deploy the GettingStartedICS user project, perform the following steps:

1. Right-click the GettingStartedICS user project and select Deploy user project.
2. Select the Interchange Server from the server list.
3. Select to deploy the entire GettingStartedICS user project.
4. Click Finish.

After a few seconds a message box state that the deployment has succeeded.

**Note:** Check the following if you were unable to deploy the project successfully:
- That the collaboration template successfully compiles.
- That the maps successfully compile.

If you have already deployed the collaboration object or connectors, be sure to stop them.

**Restarting the Interchange Server and starting the collaboration object**

Before you can use the collaboration and connectors you must restart the Interchange Server.

1. Right-click the Interchange Server and select Shut Down → Gracefully.
2. Start the Interchange Server by selecting Start → Programs → IBM WebSphere Business Integration Server Express → Interchange Server Express → Interchange Server Express.

**Tip:** In a development process it is advisable to keep the Interchange Server log file as small as possible to increase readability. One way of ensuring this is to add a command to delete the log file at the top of the batch file that starts the Interchange Server.

For example:

```bash
del C:\IBM\WebSphereServer\InterChangeSystem.log
setlocal
REM about to call the shared env file to set the ORB and JRE properties etc...
```

3. Connect to the server in System Manager by right-clicking the server and select Connect.
4. Enter the username and password (admin and null by default) and click OK.
5. Expand the Connectors folder and make sure the connectors are running. Start them if they are not running.
6. Expand the Collaboration Objects folder. Start the MyCustomerSync_Source_to_Destination collaboration object if it is not running.

10.5.9 Testing the collaboration

Since we do not have a real source and destination application we will use the Test Connector tool to simulate the two applications.

In this chapter we will do the following:

- Set up two Test Connectors, one for the source application and one for the destination application.
- Test a synchronization of Customer.Create and Customer.Delete.
- Test a synchronization using Customer.Update. This should produce a warning in the log file.
- Test a synchronization where the destination application replies with a failure.

Setting up Test Connectors

We will create two Test Connectors, one for each application. Perform the steps below to create the Test Connector for the SourceConnector. Repeat the steps for the DestinationConnector.

1. Select Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Test Connector.
2. From the Test Connector menu, select File → Create/Select Profile.
3. Select the SourceConnector from the list and click OK.
4. Select File → Connect to start the connector.

Tip: Set the two Test Connector instances side by side, with the DestinationConnector to the right of the SourceConnector.

Testing Create and Delete

Perform the following steps to test that the Create event works:

1. Select MyAppCust from the BO Type list.
2. Click the Create button to create a new instance of the MyAppCust Business Object.
3. Enter a name for the instance, for example, MyAppCust.Create.

4. Select **Create** from the list of verbs if this is not set by default.

5. Type in sample values for the Business Object attributes, for example:
   - CustId: 1001
   - Name: John Doe
   - Status: Active
   - Created: 06/31/2004 11:12:10
   - Address: Some rd. 213
   - City: Durham
   - Zip: 27713
   - State: North Carolina
   - Phone: 919 555 1234

6. Select **Request → Send** from the menu to send the Business Object.

Figure 10-26 on page 375 displays the Business Object that is sent from the SourceConnector.
Figure 10-26  Business Object that is sent from the SourceConnector

When the Business Object is sent, the following happens:

- The SourceConnector sends the Application Specific Business Object to the Interchange Server.
- The Interchange Server maps the MyAppCust Application Specific Business Object to the Customer Generic Business Object.
- The MyCustomerSync_Source_to_Destination collaboration object subscribes to the Customer.Create event, and is therefore initiated.
- The collaboration object performs a synchronous service call to the destination connector.
- The Interchange Server maps the Customer Generic Business Object to the SAL301RUSR Application Specific Business Object.
- The Interchange Server sends the Business Object to the DestinationConnector, at which point you will see Business Object in the BO.
Request List of the Test Connector for the DestinationConnector. Double-click the business object to see or change its attribute values.

For the collaboration to complete, the destination application must send a reply to the Interchange Server. In this case we will send a Reply Success:

1. Select the business object in the BO Request List.
2. Select Request → Reply → Success from the Test Connector menu.
3. Open the Interchange Server log to see the trace message. It will read:
   Customer.Create with keys (CustomerId=1001) synchronization successful

Perform the same test with the same steps in this section using the Delete verb.

**Producing the warning message**

In the example, we only support the verbs Create and Delete. The collaboration contains logic for writing a warning message to the log file if an unsupported verb has triggered the collaboration.

Perform these steps to test the unsupported verb:

1. In the Test Connector instance for the SourceConnector, change the verb to Update.
2. Select Request → Send from the menu to send the Business Object.
3. Open the log file and note the warning message:
   Customer.Update is an unsupported verb.

Note that the collaboration ends successfully even though we haven’t sent the Business Object to the destination.

**Testing failed events**

In the case where the service call to the destination application produces an exception we log this exception and fail the collaboration. Failed collaboration can be viewed and acted upon in the Flow manager.

Follow these steps to test a failed event:

1. In the Test Connector instance for the SourceConnector, change the verb to Create.
2. Select Request → Send from the menu to send the Business Object.
3. In the Test Connector instance for the DestinationConnector, select:
   a. The Business Object in BO Request List.
   b. Request → Reply → Fail from the Test Connector menu.
4. Open the Flow manager:
   a. Select Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → Flow Manager.
   b. Enter the username and password and click OK to connect to the Interchange Server.

5. In the Flow manager, select Event → Find.

   You can see the failed collaboration in the list of failed events.

6. Double-click the failed collaboration to see the details, as displayed in Figure 10-27 on page 378. Click OK to close the details.

7. You can perform the following actions on failed events:
   - Delete, which deletes the event.
   - Submit, which starts the collaboration at the step where it failed.
   - Refresh Submit, which re-starts the whole collaboration.

   Right-click the collaboration and select Submit.

8. Enter tracing level 5 and click OK.

   The collaboration will perform the service call to the destination application, and the Business Object will appear in the Test Connector instance for DestinationConnector.

9. Select the business object in the BO Request List of the DestinationConnector.

10. Select Request → Reply → Success to successfully complete the collaboration.
**Figure 10-27** Details of a failed event in Flow manager

<table>
<thead>
<tr>
<th>Event Status</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Owner</td>
<td>MyCustomerSync_Source_to_Destination</td>
</tr>
<tr>
<td>Connector</td>
<td>SourceConnector</td>
</tr>
<tr>
<td>Event</td>
<td>Customer.Create</td>
</tr>
<tr>
<td>Time</td>
<td>2004-07-08 08:51:18.29</td>
</tr>
<tr>
<td>Message</td>
<td>Error 9003 Collaboration exception at step 27 in MyCustomerSync_Source_to_Destination, thread: SourceConnector_1069291026165_9cXCommon, <a href="mailto:FlowExecContext@7c50e64.Main">FlowExecContext@7c50e64.Main</a>; Error 9003 Collaboration Failed: Customer.Create with keys (CustomerId=1001). Exception type is: ConsumerException. Exception message is: Error 11158 Collaboration exception at step 13 in</td>
</tr>
<tr>
<td>Key Attribute</td>
<td>CustId=1001</td>
</tr>
</tbody>
</table>
This chapter discusses the administrative toolset provided by Toolset Express.

We examine tools such as System Manager, Flow manager, the Failed Event Manager, the System Monitor, and the Log Viewer, and present problem-solving techniques, including the use of logging and tracing. We also look at the Collaboration Debug tool under System Manager.

The key topics in this chapter are:

- 11.1, “Administrative tools for the developer” on page 380
- 11.2, “Starting the tools” on page 380
- 11.6, “Using statistics for system interrogation” on page 413.
11.1 Administrative tools for the developer

The administrative tools can provide answers and assistance to developers as they implement Enterprise Application Integration (EAI) solutions using WebSphere Business Integration Server Express Plus.

There are two types of tools provided:

- Tools that provide statistics, status, log and trace information about the execution of the various elements of the system. Some of these tools are:
  - Statistics. See 11.6, “Using statistics for system interrogation” on page 413.

- Runtime manipulation tools that interact with elements of the system. These tools are:
  - Debugging collaborations. See “Debugging collaborations” on page 399.
  - Managing events either from the server with Flow manager as described in 11.5.1, “Using Flow manager” on page 406 or via the Failed Event Manager as described in 11.5.2, “Using the Failed Event Manager” on page 410.
  - Starting and stopping collaborations, connectors and connector agents on 11.7, “Starting and stopping components” on page 420.

For full details, consult the WebSphere Business Integration Server Express Plus documentation in the System Administration Guide.

11.2 Starting the tools

There are a number of administration tools supplied with WebSphere Business Integration Server Express Plus. There is a certain degree of duplication, as one suite of tools is for use directly with the Interchange Server and the others are to be used with a browser elsewhere in a network.

11.2.1 Starting Interchange Server Express

There are two ways to start WebSphere Business Integration Server Express Plus:

- From the Start menu, click Start → Programs → IBM WebSphere Business Integration Server Express → Interchange Server Express. This will open a command window that remains open and executes while the server is running.
From the Service control manager, start the service CWInterchange WebSphereICS. Start the service control manager by clicking Start → Programs → Administrative Tools → Services or by right-clicking My Computer on the desktop and clicking Manage. In the computer manager dialog, expand Services and Applications to display the services option. Click Services. To start a service, select the service from the list and right-click, then click Start. You can change the properties of the service by right-clicking and selecting Properties to change the service to start automatically.

11.2.2 Starting System Manager

To start the System Manager tool, click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → System Manager.

After the tool starts, be sure that you are using the System Manager view. If the InterChange Server Instance icon in the lower left hand server component management window is marked with a red circle, the server must be connected before the log and tracing settings can be configured.

To connect to the server, right-click WebSphereICS and select Connect from the menu. A dialog box will appear with the logon requirements. Use the user ID and password configured for your installation of the product. After you click OK, the red circle on the icon for the server instance should change to one with a green triangle as shown in Figure 11-1 on page 382.
11.2.3 Starting the Flow manager

To start the Flow manager tool, click **Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → Flow Manager**.

After the Flow manager starts, the window shown in Figure 11-21 on page 407 will be displayed.

11.2.4 Setting up to run the Web-based tools

For our example scenario, we are using WebSphere Application Server - Express as our Web server to run sample applications and to support the Web-based administration tools available in WebSphere Business Integration Server Express Plus. Other options are available and can be found in the WebSphere Business Integration Server Express Plus administration guide.

We used two techniques to install the WebSphere Application Server instance to enable the ICSMonitor and other managed sites: using command windows and by installing a service.
Using command windows
To run the WebSphere Application Server instance for the ICSMonitor, we cloned the standard shortcuts created to start and stop server1 as part of the WebSphere Application Server - Express installation and changed the instances of server1 to ICSMonitor in the shortcut properties window as shown in Figure 11-2.

![Image]

Figure 11-2  Configuring a shortcut to start a WebSphere Application Server instance for ICSMonitor

Installing a service
The alternative is to set up the WebSphere Application Server instance as a server. The following command, when executed from a command prompt, will install the instance of WebSphere Application Server as a service.

Example 11-1  Installing ICSMonitor as a service

```
C:\IBM\WebSphere\Express51\AppServer\bin\WASService.exe -add "ICSMonitor" -servername ICSMonitor -wasHome %WAS_HOME% -logRoot "%WAS_HOME%\logs\ICSMonitor"
```

The service can be setup to start with the machine, eliminating a step in the setup process.
11.2.5 Starting ICSMonitor

To use the Web-based ICSMonitor, a Web server must be configured as described in 11.2.4, “Setting up to run the Web-based tools” on page 382.

To access ICSMonitor, an URL similar to the one shown here, customized to the port where your Web server is installed that includes the name of the server or alias hosting the service, should be entered in your browser:

http://sab404r:7089/ICSMonitor

An initial login window will be displayed as shown in Figure 11-3. Click Browse if you wish the Web page to populate the Server list with server values and the field is not populated. Make a selection from the list or enter the Server name, User Name, and Password for a suitable user configured for the system (for example, the default administrator user ID and password) and click Login.

![Figure 11-3  ICSMonitor - Login](image-url)
After a successful login, the browser displays the System Overview window for the System Monitor as shown in Figure 11-4.

![System Overview](image)

*Figure 11-4  ICSMonitor - System Manager*

From this window, the other options available under the ICSMonitor are available. Some of these are as discussed in:

- “Connector statistics - ICSMonitor” on page 415
- “Collaboration statistics” on page 417

### 11.2.6 Starting Failed Event Manager

To run the Web-based administration tools such as the Failed Event Manager, a Web server must be installed and running. See 11.2.4, “Setting up to run the Web-based tools” on page 382 for a specific example or see the WebSphere Business Integration Server Express Administration guide.

To access Failed Event Manager, a URL similar to the one shown in the example should be entered in your browser, customized to the port that your Web server
is installed on and including the name of the server or the alias hosting the service:

http://sab404r:7089/FailedEvents

An initial login window, as shown in Figure 11-3 on page 384, will be displayed. Enter the Interchange Server server name, user ID, and password of a suitable user configured for the system (for example, the default administrator user ID and password). If your login is successful, the window illustrated in Figure 11-25 on page 411 will be shown.

### 11.3 Logging and tracing

Logging and tracing refer to the same kind of information; however, tracing provides more detail, and WebSphere Business Integration Server Express Plus can manage these records separately if required. Logging and tracing are configured from a panel in the System Manager tool.

#### 11.3.1 Configuring logging and tracing

To change the general logging and tracing settings, start the System Manager tool as described in 11.2.2, “Starting System Manager” on page 381. Right-click the WebSphereICS (Production) instance and, from the menu that appears, click Edit Configuration. After a short delay, the WebSphereICS configuration tab will appear in the upper right. This tab is itself another series of tabs. To configure logging and tracing, click the Trace/Log Files tab.

After these steps, if no configurations have been changed, your System Manager window should resemble Figure 11-5 on page 387.
Note the left and right frames at the top of the configuration panel. The left frame controls the overall logging and the right frame allows the tracing information to be handled separately. Developers will have requirements that differ from the final production runtime system. You may wish to generate trace information and without reviewing all the trace results to find your event information. Trace files also can grow until they fill a hard drive and old trace information has little value.

The configuration illustrated in Figure 11-6 on page 388 shows log events and trace events being written to different files. The rolling archive of files shows that we are only keeping the last 4 trace files and 16 event log files. We will allow the log file to become much larger than the trace file.

**Note:** The system will finalize the log file and archive it when it reaches the specified size, but the trace file will not be finalized and archived until the `<VARIABLE>` server is stopped and started.
Turning on tracing in components

The amount of trace information can be controlled by each component. For our sample scenario, we will turn on maximum tracing in the sample collaboration, the source adapters and the destination adapters.

Turning on tracing in the CustomerSynchronization collaboration

Tracing in collaborations is turned on from the System Manager. Make sure that the System Manager is started and connected and the Interchange Server component management view is active. (How to start System Manager and connect to the InterChange Server are discussed in 11.3, “Logging and tracing” on page 386.)

Expand the tree under the Interchange Server by clicking the plus (+) symbol beside WebSphere Interchange Server (Production) to show the components available for management in the active server. Expand Collaborations Objects by clicking its plus (+) symbol. Click:

CustomerSynchronization_SourceToDestination::CustomerSynchronization

This will select the collaboration for the demonstration sample. The System Manager window will show the details illustrated in Figure 11-7 on page 389.
Right-click the selected collaboration and select **Properties** from the menu that appears. The dialog shown in Figure 11-8 on page 390 will be displayed. There are two trace level lists available. In this example, we have set the System Trace Level and the Collaboration trace level to the maximum value of 5. Click **OK** to set the values and close the dialog.
Turning on tracing in connectors

From the Interchange Server Component Management view, navigate to Connectors as shown in Figure 11-9 on page 391. Right-click **SourceConnector** and select **Properties** from the menu that appears.
The properties dialog shown in Figure 11-10 shows the configuration after the AgentTraceLevel property has been updated to 5. The value is updated by clicking the cell beside the property name and selecting 5 from the list that appears. Click OK to confirm the changes and close the dialog.
11.3.2 Generating sample tracing

In the discussion in this section, the Test Connector was used to generate the sample logging and tracing. The scenario used is the same as in README_SystemTest_Sample.rtf but will be summarized here. This document can be accessed from the Start menu by selecting Start → Programs → IBM WebSphere Business Integration Server Express → Samples → System Test → Readme System Test Sample.

If you are familiar with this verification process, proceed to 11.3.3, “Viewing the tracing information” on page 393 to examine the trace output.

After starting the Interchange Server, start the Test Connector that will be the source. From the Start menu, click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Test Connector.

Repeat to start another instance of the Test Connector. This will be the destination.

Using the source Test Connector dialog, click File → Create/Select Profile. When the Connector Profile dialog appears (Figure 11-11), select SourceConnector and click OK.

Then, using the destination Test Connector dialog, click File → Create/Select Profile. When the Connector Profile dialog appears, select DestinationConnector and click OK.

In each Test Connector dialog, click File → Connect.
In the test connector source instance, click **Edit** \(\rightarrow\) **Load BO**. From the File selection dialog, make sure that the path looks similar to this:

```
C:\IBM\WebSphereServer\Samples\SystemTest
```

The path may differ slightly, depending on where the product WebSphere Business Integration Server Express Plus was installed. There should be two files of type BO. Select **Customer.bo** and click the **Open** button. When the instance dialog appears, type any name in the Enter Name field and click **OK**. The Source connector dialog will be populated with information and the BO type will be Customer.

Using the Source Test Connector, initiate a request. Click **Request** \(\rightarrow\) **Send**. Change the Destination Test Connector and reply to the request by selecting **Customer.Create** from the BO Request List. Then, click **Request** \(\rightarrow\) **Reply** \(\rightarrow\) **Success**.

### 11.3.3 Viewing the tracing information

The trace information goes into a file as specified in the process described in 11.3.1, “Configuring logging and tracing” on page 386. If you view the file using an application such as Notepad after running the Installation verification procedure, you will find trace records (one per line). Each record will have:

- Date and time stamp
- System
- SS (Sub-system)
- Other system information
- Type with the value trace
- Mesg, with specific trace information

There will also be lines in the file that have additional data such as customer information. More information about these fields can be found in the WebSphere Business Integration Server Express Plus documentation in the *System Administration Guide* in the chapter that describes troubleshooting.

If you look at the raw file, you will notice that the lines in the file are not continuous with time. Each of the systems that make up WebSphere Business Integration Server Express Plus will cache information and update the trace file periodically.

WebSphere Business Integration Server Express Plus provides a view that has capabilities for sorting and filtering the log records, and it is probably the most useful way to view log and trace records. An overview of the log view is provided in 11.3.4, “Using the Log Viewer to look at trace” on page 394.
A trace can still be used even if the tracing to a separate file is not specified. In this case, the trace information will be intermingled with the event information. It is possible to use the filter option available with the Log Viewer to filter to only show trace records.

### 11.3.4 Using the Log Viewer to look at trace

To start the Log Viewer, from the Start menu click **Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → Log Viewer.**

When the Log Viewer starts, the trace file needs to be opened. Click **File → Open,** and navigate to the configured trace file using the Open dialog.

After the Log Viewer has loaded the file, click **Date Time** in the display grid to quickly sort the values. More sorting options are available using a control dialog on the menu by clicking **View → Sort,** but sorting by date and time is sufficient for this discussion.

**Note:** Sorting by date and time will lose the natural order of all rows with the same time stamp. Due to the limitations of the host operating environment’s clock, this can mean a number of records will be displayed in a different order to that in which they occurred.

If you need to understand the actual order of events within one system, you must view the raw trace file.

**ConnectorAgent - Source initiation trace records**

Figure 11-11 on page 392 shows trace information for the time a Customer request was sent as detected by the ConnectorAgent during a polling loop.
Using the Log Viewer, we can locate and examine the low level progress of the business function as it interacts with WebSphere Business Integration components. The values highlighted in Figure 11-12 show the relevant messages from the server log:

- **Polling App Connector** - as the Connector Agent begins another polling loop
- **Polling for subscription events** - as the Connector Agent checks for administration updates
- **Received BusObj from App Connector** - the Connector Agent has detected the Business Object
- **Posting BusObj to Interchange** - alerting the \(<\text{VARIABLE}>\) server about the Business Object
- **Business Object before delivering to the Interchange Server** - this row has a button in the BusObj column. Clicking this button will display the details of the Business Object. (See Figure 11-13 on page 396.)
- **Entering deliverBusObj** - prepare to deliver the Business Object to the server
- **Triggered event Customer.Create** - the name of the request
- **deliverBusObj returning 0** - successfully passed to server
After the ConnectorAgent has notified the WebSphere Business Integration Server of the Business Object, the collaboration system begins processing. The sub-system in this case is CustomerSynchronization.

Figure 11-14 on page 397 shows the Log Viewer with the records generated by the collaboration highlighted.

After a closer inspection of the raw trace file in this case, we discovered that the records were somewhat out of order. The order in the discussion below is based on the raw trace file. The order of the records may not be important, depending on the issue being investigated. It may be sufficient to check for correct processing.

**Note:** You should look for these ConnectorAgent trace records to determine if your ConnectorAgent is configured correctly and the required triggering or other mechanism is working. You can examine the data to check if the data format for the Business Object is as expected.

**Collaboration - CustomerSynchronization trace records**

After the ConnectorAgent has notified the WebSphere Business Integration Server of the Business Object, the collaboration system begins processing. The sub-system in this case is CustomerSynchronization.

Figure 11-14 on page 397 shows the Log Viewer with the records generated by the collaboration highlighted.

After a closer inspection of the raw trace file in this case, we discovered that the records were somewhat out of order. The order in the discussion below is based on the raw trace file. The order of the records may not be important, depending on the issue being investigated. It may be sufficient to check for correct processing.
Most of the trace entries are reasonably obvious. The significant events for the initial processing of the collaboration are:

- Performing event sequencing
  - WebSphere Business Integration Server Express Plus has the capability to ensure that Business Object processing on specific instances of Business Objects occurs in a predetermined order.
- Received an object from SourceConnector with ObjectEventId.
- Creating new CustomerSynchronization_SourceToDestination object instance.
- Started working on event: Customer.Create.
- Sending Item: Customer.Create, DestinationName:DestinationConnector.
- Detail for Customer.Create
  - This is similar to ConnectorAgent trace discussion. You can click **Customer** to show a dialog displaying the details of the Business Object.
DestinationConnector and collaboration finalization - Trace

When the Connector Agent at the Destination Connector completes the processing of the customer creation operation on the destination system, there is some finalization processing at the collaboration and the destination connector.

The collaboration object notifies the destination connector that the processing of the Business Object is complete and the collaboration is able to unlock the Business Object to allow other objects with the same identification identifiers that may be under synchronization control to continue. Notification to the Destination Connector allows the Destination Connector to remove any information required to restart the transaction in the event of failure.

The relevant log records, include messages from the Connector Agent at the Destination Agent as follows:

- Polling every 5000 millisecond
- Polling for subscription events

The destination connector also looks for administration events.

The collaboration then performs some further processing and log messages as follows:

![Figure 11-15 Destination Connector and collaboration finalization](image-url)
Chapter 11. Administration tools

11.4 Debugging collaborations

For debugging collaborations, we will use the same installation verification scenario we used to view logging and tracing. (See 11.3.2, “Generating sample tracing” on page 392 for information about how to set up this scenario.)

In this example, we will use the SourceConnector development tool and the Destination Connector interacting with this collaboration:

CustomerSynchronization_SourceToDestination::CustomerSynchronization

Using the collaboration debugging tool, we can show a step-by-step view of the execution flow.

Make sure that the System Manager is started and connected and the Interchange Server component management view is the active view. (See 11.3, “Logging and tracing” on page 386 for information about starting the System Manager and connecting to the Interchange Server.)

Expand the tree under the Interchange Server by clicking the plus (+) symbol beside WebSphere Interchange Server (Production) to show the components available for management on the active server. Clicking the plus (+) symbol beside Collaborations. Click:

CustomerSynchronization_SourceToDestination::CustomerSynchronization

The System Manager window will display details as shown in Figure 11-7 on page 389.

Right-click the selected collaboration and click Attach Debugger. The System Manager window will display a new window. This is the Collaboration Debugger illustrated in Figure 11-16 on page 400).
Double-clicking **Main** in the center panel displays the details of the collaboration. Right-clicking the center panel and clicking **Show labels** updates the window to with more information as shown in Figure 11-16.

From this window, it is possible to set break points and stop the flow as the collaboration executes and examines BusObj details.
Figure 11-17  Collaboration debug initial detail

Double-clicking each of the circles and plain rectangles toggles the break point selected. Figure 11-18 on page 402 illustrates the debug window with all the break points set.

To illustrate the Collaboration debugger process, we use the Source Connector and Destination Connector. We use the Source Connector to send requests to the DestinationConnector and using the DestinationConnector, return a success in one scenario and a failure in another.
11.4.1 Debugging a success flow

Use the process described in 11.3.2, “Generating sample tracing” on page 392 to invoke the debugger.

Having set the all the break points by double-clicking the rectangles and circles, if we perform the send operation from the Source Connector, the flow will stop in the debugger.
The Collaboration Debugger menu in the System Manager will display a yellow flag and a pause symbol below CustomerSynchronization with the text Event n - suspended, where n is the increments for each run of the collaboration.

Right-clicking the Event n - suspended value in the left panel and clicking Display will show the same flag and event number on the central panel as illustrated in Figure 11-19.

![Collaboration Debugger - WebSphere Studio Workbench 5.0](image)

**Figure 11-19  Collaboration stopped at the main entry point**

Right-clicking the Main circle or the central panel and clicking Step Over will move the flag to the next element.
Right-clicking the **Main** circle or the central panel and clicking the **Step Over** menu item will also perform the synchronous call with the DestinationConnector to the To.Create action. If we change to the Destination Connector instance, the **BO Request List** grid shows the **Customer.Create** entry.

To see the success path (the first scenario) in the debug tool, click **Customer.Create** from the Destination Connector, then select **Request → Reply → Success** from the menu that appears.

Return to the System Manager and right-click **Event n - suspended** value in the left panel. Select **Display** on the menu that appears. This will display a yellow flag on Event 5. The flagged event is the last step in the flow through a successful collaboration and is shown as green concentric circles in Figure 11-20 on page 405.

**Note:** Figure 11-20 on page 405 has been configured to show unique identification numbers by right-clicking the background of the main central pane and clicking **Show uniquelDs** in the menu that appears. In a more complex example, this could be a useful reference for taking notes or communicating with other developers.
11.4.2 Debugging a failure flow

Perform the same operations as those in the success flow scenario described in 11.4.1, “Debugging a success flow” on page 402, to start the Source Connector send operation.

The Collaboration Debugger menu in the System Manager will display a yellow flag and pause symbol below CustomerSynchronization with the text Event n - suspended, where n is increments for each run of the collaboration.
Right-clicking **Event n - suspended** in the left panel and clicking the **Display** option will show the same flag and event number on the central panel as illustrated in Figure 11-19 on page 403. Again, the collaboration will be stopped at the main entry component. Press **F6** or right-click the main panel and select **Step Over** from the menu to move the yellow flag to Action Node 2.

Pressing **F6** or selecting from the menu to step to the next break point will send the item to the Destination Connector. The pause flag will continue to appear with Action Node 2 because the call to the Destination connector is a synchronous call.

To cause the collaboration to take the failure path, move to the Destination Connector instance, click **Customer.Create** in the BO Request List. Select **Request → Reply → Fail** from the menu that appears.

If you return to the System Manager, the main debug panel will now display the yellow pause flag at Action Node 4 on the failure path.

Using the **F6** key or menu item to step to the next break point will send the item to End Failure Node 6. To complete the flow, right-click the panel again and select **Run**.

See 11.5, “Managing failed events” on page 406 for information about failed events.

### 11.4.3 Finished debugging

When you have completed the collaboration debugging session, you must return to the System Manager Interchange Server Component Manager, right-click **CustomerSynchronization** and select **Detach Debugger** from the menu.

### 11.5 Managing failed events

WebSphere Business Integration Server Express product maintains information of all failed flows in an internal database. The administration support program, Flow manager, and the Web-based Failed Event Manager are tools that allow administrators to retry failed flows.

#### 11.5.1 Using Flow manager

The Flow manager tool is started from the Administrative tools on the startup menu. (The full command is described in 11.2.3, “Starting the Flow manager” on page 382.) After a period of time, the window shown in Figure 11-21 on page 407 will appear.
Locating failed events with Flow manager

Flow manager allows events to be searched by a number of criteria:

- Status (any, Failed, In Transit, Possible Duplicate, Deferred, Waiting)
- Date and time
- Error text and Business Object details

Note: The error text and Business Object details filter is valuable to production administrators, but for the initial simple scenarios that a developer faces, the default any option will probably be sufficient.

For example, if Flow manager is started after the exercise described in 11.4, “Debugging collaborations” on page 399, then a failed event will be located and the window will be updated when Find is clicked from the Flow manager dialog as shown in Figure 11-22 on page 408.

Actions can be performed on these failed events by right-clicking the selected row and selecting the desired option from the menu that appears. Actions to be performed are described in 11.5.3, “Failed event actions” on page 412.
Further details are available on the failed item by right-clicking the selected row and selecting:

- **Display Detail** as shown in Figure 11-23 on page 409
- **Show Event Details** as shown in Figure 11-24 on page 409
Figure 11-23  BusObj detail for a failed event

Figure 11-24  Failed event detail
11.5.2 Using the Failed Event Manager

The Failed Event Manager is a Web based service. How to start Failed Event Manager is described in 11.2.6, “Starting Failed Event Manager” on page 385.

After the Failed Event Manager starts, a filter selection window is displayed. This window performs the same functions as the filters provided with Flow manager for use locally on the Interchange Server. Using the window, a developer or administrator may locate failed events.

For example, if Failed Event Manager is started after the exercise described in 11.4, “Debugging collaborations” on page 399 is completed, you may locate a failed event by clicking Submit in the Failed Event Manager Query window shown in Figure 11-25 on page 411.
A new window will be appear that displays the failed event. This window is illustrated in Figure 11-26 on page 412.
The Table of Failed Events window contains a table with these columns:

- Status
- Point of Failure
- Connector
- Business Object
- Date and time
- Message
- Key(s)
- Detail

Actions can be performed on these failed events by clicking the check box associated with each row and clicking the button associated with the action. Examples are illustrated in Figure 11-23 on page 409 and Figure 11-24 on page 409.

11.5.3 Failed event actions

There are a number of possible actions that can be performed from either Flow manager or Failed Event Manager. The management actions are:

- Delete - No action is required. Perform house keeping and remove the failed event.
Submit - This is one of the main actions to be performed for administrators. This object submits the BusObj to the collaboration to be processed again. The target system may not have been available or it may not have functioning correctly at the time of initial processing, the system is now ready, and the action should be successful.

Refresh submit - This action behaves like the submit action, but the source system is interrogated by the connector to refresh the BusObj.

Cancel waiting - This action cancels long-lived business processes. This option is not available in our scenario.

Flow manager and Failed Event Manager have additional options for displaying, details, saving, and printing. Which options are available varies depending on the tools.

11.6 Using statistics for system interrogation

Within the administration tools of WebSphere Business Integration Server Express Plus, there are a number of places where statistics can be found that can aid in the search for issues.

Using the tools, it is possible to get statistics for:

- The number of events sent to and received from connectors - This could be useful to a developer if he or she believes an event should have been generated, but it was not. The statistics provide suggestions of places to go to try to understand the issue.
- The number of successful or failed items - If the number of failed items for example increases, then the failure paths of the collaboration should be a developer’s focus.

11.6.1 Connector statistics

Using either System Manager on the server or the Web-based ICSMonitor, it is possible to obtain the number of events sent and received from the different connectors in the system. Detailed statistics about Business Objects used by the different connectors are also available.

**Connector statistics - System Manager**

To view statistics within System Manager, start the System Manager tool as described in 11.2.2, “Starting System Manager” on page 381.

From the Interchange Server Component manager view, expand the Interchange Server instances and the current server instance until the Connectors option is
available. Right-click **Connectors** and select **overview** from the menu. This will provide an overview of events on all Interchange Server connectors as shown in Figure 11-27.

The Connector Overview - WebSphereICS tab shows a grid with each connector and the following information:

- Connector
- Agent State
- Total Received
- Total Sent

For a statistical view of a specific connector from the Interchange Server Component manager view, expand the Connectors option, and right-click the required connector. Select **Statistics**.

The Connector Specific Statistics tab shows information about the total number of Business Objects, as well as the number of each type sent and received as illustrated in Figure 11-28 on page 415.
Figure 11-28 System Manager Statistics - Connector detail

**Connector statistics - ICSMonitor**

Connector statistics at both the overview and the detailed level can also be obtained with the ICSMonitor. To start the ICSMonitor see 11.2.4, “Setting up to run the Web-based tools” on page 382.

An overview of the statistics can be seen by selecting **Connector Overview** from the View: list to the left of the ICSMonitor Web page.
This Web page shows the a list of connectors and summary information about times, the status of the Agents, and the total number of Business Objects sent and received.

The details for a specific connector can also be displayed. Click the Connector option under the View: list to the left of the ICSMonitor Web page.

The Web page displayed contains two frames: one for Business Objects and the other for Connector Subscriptions. From each frame, you can select the connector for which you wish to gather statistics.

The Business Objects frame will show the total of each type of Business Object that has been sent and received.

The Connector Subscriptions frame will show collaboration and initiation details for each connector.
An example of the Connector statistics in the ICSMonitor Web page is illustrated in Figure 11-30.

![ICSMonitor - connector detail statistics](image)

**Figure 11-30  ICSMonitor - connector detail statistics**

### 11.6.2 Collaboration statistics

Using either System Manager on the server or the Web-based, it is possible to obtain the total number of flows and the number of failed flows for each collaboration in the system.

**Collaboration statistics - System Manager**

To view collaboration statistics in the System Manager, start the System Manager tool as described in 11.2.2, “Starting System Manager” on page 381.
From the Interchange Server Component Manager view, expand the Interchange Server instances and the current server instance until the **Collaborations** option is available. Then right-click **Collaborations** and select **overview** from the menu. This will display an overview of all Interchange Server collaboration flows as illustrated in Figure 11-31.

![Figure 11-31 System Manager Statistics - Collaboration Overview](image)

Statistics for a specific collaboration can be found by expanding the **Collaboration Objects** object, right-clicking the specific collaboration object, and selecting **Statistics** from the menu.
Collaborations statistics - ICSMonitor

Collaboration statistics at the overview level can also be obtained with the ICSMonitor. To start the ICSMonitor see 11.2.4, “Setting up to run the Web-based tools” on page 382.

An overview of the collaboration statistics can be seen by selecting Collaboration Overview from the View: list to the left of the ICSMonitor Web page. A sample Collaboration Overview Web page is shown in Figure 11-33 on page 420.
There are other collaboration views available from the ICSMonitor, such as graphs that display collaboration event rates. These tools would be useful to a developer wishing to understand the performance characteristics of a specific server installation.

11.7 Starting and stopping components

In System Manager and ICSMonitor, there are commands available for starting and stopping connectors, connector agents, collaborations, and the Interchange Server instance.

11.7.1 Controlling components from System Manager

Controlling components is done from the Interchange Server Component Management view in System Manager.

Start System Manager as described in 11.2.2, “Starting System Manager” on page 381 and expand the Interchange Server Component Management view options to show Collaboration Objects and Connectors.
To perform actions that change the running state of a component, right-click the component and select the action from the menu. Figure 11-34 shows System Manager being used to change the state of a connector.

**Figure 11-34  System Manager - ready to perform action on a Connector**

### 11.7.2 Controlling components from ICSMonitor

Administration actions can be performed from a number of the more specialized ICSMonitor Web pages, but it is possible to perform all actions from the System Overview page.

The tool bar at the top of the System Overview panel of the System Overview Web page is used to control the components. To perform an action, click the check box next to the component and then click the appropriate button on the toolbar.

Full details of the available actions and the meaning of these components can be found in the WebSphere Business Integration Server Express Plus Administration guide in the Administering components of the system chapter.
**Figure 11-35** ICSMonitor - ready to perform action on some connectors

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>Status</th>
<th>Start Time</th>
<th>Total Up Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphereICS</td>
<td>Server</td>
<td></td>
<td>June 24, 2001 03:38 AM EDT</td>
<td>10 min 31 sec</td>
</tr>
<tr>
<td>CustomerSynchronization_SourceToDestination</td>
<td>Collaboration Running</td>
<td>June 24, 2001 03:38 AM EDT</td>
<td>10 min 14 sec</td>
<td></td>
</tr>
<tr>
<td>DestinationConnector</td>
<td>Connector</td>
<td>Running</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceConnector</td>
<td>Connector</td>
<td>Running</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SystemTestObject</td>
<td>Collaboration Running</td>
<td>June 24, 2001 03:38 AM EDT</td>
<td>10 min 15 sec</td>
<td></td>
</tr>
</tbody>
</table>
In this part, we design and implement a more detailed integration example using WebSphere Application Server - Express and WebSphere Business Integration Server Express.
Chapter 12. Sample requirements, design, and specification

This chapter details the business requirements for a sample integration solution developed for a fictitious real estate agency using IBM WebSphere Application Server - Express and IBM WebSphere Business Integration Server Express Plus. We describe the business problem and then provide detailed design and specifications for the sample solution implemented. This discussion is divided into:

- 12.1, “Introduction to the samples” on page 426
- 12.2, “Business problem description” on page 427
- 12.3, “Objectives of the solution” on page 429
- 12.4, “Process modeling” on page 430
- 12.5, “System integration design” on page 448
### 12.1 Introduction to the samples

Business applications need to communicate, interact, and integrate with other business solutions and data. Solutions are developed to promote this integration and interactivity. Specifying the right integration pattern must be done by considering the specific business process requirements. These requirements encompass not only the specific needs of the integration to be deployed, but also the enterprise's IT infrastructure and legacy system technology.

When a system is defined and developed to meet the customer needs and problems it must address a typical set of requirements, such as:

- Integration
- Interoperability
- Reliability
- Accessibility
- Availability

One of the most important tasks that a solution must accomplish is to eliminate or reduce the risk of data input duplication. Different sources of the same information normally cause inconsistency errors due to asynchronism, which means that the same variable can have different values at the same time in disparate databases. For example, if a solution that is designed to manage a warehouse inventory has different methods for inputting material amounts (for example, an invoice register application and a check-in application) and these applications are not integrated, it can cause inventory inconsistencies. The Warehouse manager's question then becomes: “What is the correct information?”

One of the ways to deal with this task is to develop interfaces that update different databases. However, this approach is not recommended because it takes too much time and requires specific technical skills. It is an expensive solution that involves many work hours and is difficult to maintain. For the most part, these interfaces also have no reconfiguration functionality.

WebSphere Business Integration promotes this integration and data reliability easily and affordably, with high productivity and scalability. This system supports specific solution requirements, minimizing development efforts, architectural efforts, and work force costs.

WebSphere Business Integration Server Express is designed to assist mid-sized companies with the integration of business processes. It is a core component of the business process integration strategy for small and medium businesses (SMBs) offered by IBM. This middleware makes it easier for customers to integrate their existing IT systems, business processes, and applications.
WebSphere Business Integration Server Express itemizes conceptual elements and then provides the solutions for each of these elements. The conceptual architecture consists of the following elements:

- **Model** - Create a simulation of the business processes.
- **Integrate** - Tie the various discrete isolated process parts.
- **Connect** - Bring employees, customers, and business partners into a network.
- **Monitor** - Ensure that the business processes are performing as expected.
- **Manage** - Improve business effectiveness by modification of process.

SMBs plan for the reality of growth; however, they typically do not have the means to develop an exhaustive system strategy as part of these plans. In other words, understanding what it means to have different applications running in parallel and causing data replication may not be included in their growth and system development strategies. However, the strong dependency on control to keep the company on a growing path drives the need for an integration solution. WebSphere Business Integration Server Express is an attractive option because applications can be tied together with an attractive return on investment (ROI).

### 12.2 Business problem description

A real estate agency has a system that empowers its business. The company uses SAL301R, a Web-based application that provides it with a centralized property catalog. This solution allows buyers to use the Web to get information about the homes that the real estate agency is offering. The application’s users are mainly visitors, customers, agents, and an administrator. Visitors can search the catalog, according to their interests, by type, area, and price range and view the property details. The application also offers new visitor registration. This registration provides the company with a list of potential buyers. When customers have an interest in purchasing property, they can select specific offerings and add them to their interest list. Then, they can make a request to view a property or order brochure about it. Agents can register, change property information, get customer interests, and generate user and agent property interest reports. The administrator is able to create news items and enable or disable agent accounts.

The problems that an application integration needs to resolve are:

- **SAL301R** (the realtor’s Web application) is not integrated with the agency’s legacy system, which is where the company’s official administrative information resides. This means that many inconsistencies occur during the manual transfer of information, from the Web application, to the legacy system (SAB404R). This lack of synchronization normally causes errors and faulty analysis, and the final result of this disconnect is that the users, mainly managers, do not trust “the system.” In these cases, parallel controls such as
Excel and Access files are created in the hopes of resolving this mistrust, but these files are normally not updated adequately.

For example, if an agent changes the price of a specific property in SAL301R but not in SAB404R, the price of the property viewed by a customer will be different than the price viewed by the company. If a sales manager asks for a report of property prices, the manager will receive different information. In addition, if this property is sold at the Web price, and SAB404R calculates the agent bonus from this sale, the figure will be incorrect. One of the most important pieces of information to this company is the property inventory. This information is regarded as the core business of the real estate agency. When an agent introduces a new property using SAL301R and this information is not transferred to SAB404R, the sales manager does not know what the company’s inventory is and what should be offered to the customers.

- The process of requesting a new property advertisement in a newspaper is completely manual and depends on the agent. So, if a new property has been offered and the agent forgets to call the newspaper, customers will not have access to this information. Of course, this property is announced in the SAL301R, but many potential buyers use the newspaper as a source for real estate listings.

- All contact with customers (buyers, sellers, and visitors) by an agent is made through manual interaction. Considering that this kind of business relies on relationships, these contacts must be sensitive to the timing of a deal. Because this process is completely manual, some contacts may be delayed or overlooked, and this can compromise transactions.

- The escrow agent is not automatically linked to the real estate agency. So, the Escrow Open and Escrow Close activities are completely controlled by office personnel. When a buyer or seller wants to know about his or her escrow process, an office employee must track it, and this can take a long time.

- Sales figure reports are generated based on the information contained in SAB404R. Since the systems are not integrated, this important information may be inaccurate, which is a real concern for the estate agency owners.

The following process description illustrates the problems caused by using separate systems:

- Register new property

  A customer calls the real estate agency with a request to sell his home. The agent registers the new property for the seller using SAL301R. This process creates an available property in the property catalog database. The agent then sends an advertisement to the newspaper by fax, phone call, or letter and this advertisement notifies buyers that a new property is on the market. The agent also creates the property (with specific information such as price and location) and property status in SAB404R’s inventory table.
Sales process

A buyer provides a bid to the agent, and the agent presents this offer to the seller. If the seller accepts the price, the escrow process is started. At that time, the agent must remove the property from the Available list in the property catalog database and must contact the newspaper and request that the property advertisement be removed. If the escrow process is successful, then the agent must call the seller and buyer and let them know. The real estate agency then determines the agent’s bonus based on the property’s sale price.

If the escrow process fails, then the agent sends an advertisement to the newspaper by fax, phone call, or letter to notify buyers that the property is on the market again. The agent must also call the buyer and seller to notify them that the transaction did not occur.

12.3 Objectives of the solution

When defining the requirements of the solution you must investigate the business needs that the integrated system must meet. The integration specialist must work cooperatively with business analysts and architects to clarify these requirements in order to achieve the project goals. Therefore, clearly defined objectives are paramount to the outcome of the project.

Considering the problems we have described, the objectives of this integration solution must be to address and resolve each problem with the current systems.

The product catalog database in SAL301R must be integrated with SAB404R’s database. This integration must ensure that the data in both databases is updated and synchronized. This integration must prevent data inconsistency and provide the real estate agency owners and managers with the reliability that they need to run the business. It means that the system must guarantee:

- All property registrations in SAL301R are reflected in the inventory control portion of SAB404R.
- The newspaper is automatically notified when the properties are registered and provided with property descriptions so that the properties are advertised.
- Registration of visitors and their interests as potential customers is reflected in SAB404R.
- All changes in property price made with SAL301R is reflected in the property section of the official register that resides in SAB404R.
- When a buyer makes a bid that is accepted by the real estate agency and by the seller, the agent inputs this information in SAL301R, then the property
status also needs to be changed from Available to Under Escrow in SAB404R.

- The buyer and seller are automatically notified when the bid is accepted and the escrow process is started.
- The escrow agency is automatically informed that the process has begun and that the buyer’s and seller’s documents will be delivered.
- If the escrow process is successful, automatic notification is sent to newspaper to remove the property’s advertisement.
- If the escrow process is successful, automatic notification about the transaction is sent to the buyer and seller.
- If the escrow process is successful, the property status in inventory control is changed from Under Escrow to Sold in SAB404R.
- If the escrow process is unsuccessful, automatic compensation to revert all transactions is done.
- Sales figures and the agent bonus are calculated after a successful transaction.
- Records for all new visitors to the Web site, property offers, property prices, new employees, new agents, and customers are updated or created.

In order to clarify the objectives of the project, the system integration must satisfied all activities described on the Process Modelling, which is discussed in 12.4, “Process modeling” on page 430.

### 12.4 Process modeling

Process modeling is necessary in order to define the solution requirements and to gain a clear comprehension of the business process for each activity. Therefore, we modeled the anticipated process flow changes.

#### 12.4.1 Integration solution description

Process modeling allows the system architect to identify the business processes that are candidates for an integration solution. A business process is a set of interconnected business activities with a beginning and an end, as well as clear inputs and outputs. In this regard, the role of the architect is to study, define, and model the process that will be automated.

Each business process should consist of a process definition that describes the required business activities. Integration specialists, business analysts, and
architects work together to define each business process, answering questions such as:

- What activity must be done for each business process?
- What are the principle activities for each business process?
- What are the characteristics for each business process or component?
- What is the sequence of the activities?

The following process description presents the scenarios in the new integration solution that will be deployed for the real estate agency:

1. Register new property

A seller calls the agency and asks to have an agent sell his or her home. The agent registers the new property for the seller using SAL301R. This process creates a property in the property catalog database. A trigger creates the same entry in SAB404R inventory table and sends a notification to publish the new property in the newspaper.

Figure 12-1 shows the integration between SAL301R and SAB404R.

![Figure 12-1 Register new property process diagram](image)

2. Sale process

Once a seller and buyer have been matched, the agent changes the status of the property from Available to Under Escrow, which sends a message to the external escrow agent to approve the final sale of the property. A message is
sent to the buyer and seller notifying each of their obligations for fulfilling a
successful escrow closing and property title transfer. The system also
updates the status of the property in SAB404R’s inventory table, the
newspaper, and the property catalog database.

At this time, there is a pause in the process during which the system waits for
a response to the message sent to the escrow agent. The escrow agent
collects the settlement statement (sales contract) and holds the financial
documents and agreements for both the buyer and seller. The agent also
responds to the system message that will trigger the proper response in the
system. There are two possible scenarios that may occur at this point:

– Failed sale

  If the buyer or seller cannot meet his or her obligations, then the escrow
  agent sends a message that the sale should not be completed. The
  system then compensates for the changes made to the property status in
  SAB404R’s inventory table, the newspaper, and the property catalog
database and notifies the buyer and seller of the failed sale.

– Successful sale

  If the buyer and seller meet their obligations then the escrow agent sends
  a message that the sale should be completed. The system then updates
  the status to sold in SAB404R’s inventory table and in the property catalog
  DB as well as notifying the buyer and the seller of the successful sale. An
  update is sent to the newspaper advising that the property is no longer on
  the market. Also the system then updates the sales figure in SAB404R’s
  employee table.

Figure 12-2 on page 433 illustrates the scenarios and the processes they follow
to complete the escrow parent process.
Figure 12-2  Sale process diagram - escrow process started-up

Figure 12-3 on page 434 shows details about the interactions between the agent, seller, buyer, and escrow agent.
In order to clarify our process requirements, we modeled the complete solution and this is discussed in 12.4.2, “Business modeling using IBM WebSphere Business Integration Workbench” on page 434.

12.4.2 Business modeling using IBM WebSphere Business Integration Workbench

Modeling the process is the first step in the integration system architecture and design. The model provides a business analyst, at the modeling and execution levels, with the ability to perform mappings and translations between data across different systems.

IBM WebSphere Business Integration Workbench provides a common workspace to facilitate communication between a business and IT professionals while they model the process. The workspace allows enterprises to transform the necessary business content to IT content and enables the transformation of IT content to business content.
WebSphere Business Integration Workbench provides the Business Modeler (core), the UML Transformer (Modeler), Form Designer, and XML Mapper (extensions).

WebSphere Business Integration Workbench allows the business analyst to:

- Create, simulate, analyze, and select optimal new process designs
- Share process model information through the Web with a secure repository for access and version control
- Convert business process models to formats that can be exported directly to MQSeries® Workflow
- Monitor work-in-process items and perform corrective actions

Additionally, actual process metrics answer what-if questions in modeled scenarios and make process improvements.

WebSphere Business Integration Workbench provides developers and business analysts with the ability to define systems requirements and describe the integration objectives through modeling. It enables developers to model the systems in Unified Modeling Language (UML), gives them a tool to map the important information contained in UML models back to business process models, and allows developers and business analysts the opportunity to transform common elements between UML models and business process models.

The purpose of the WebSphere Business Integration Workbench repository is to provide a place to store different types of data related to a specific process. It is divided into two categories:

- **Organization data** - These are data that pertain to the attributes of the organization. It is possible to create, modify, or access organization data by using commands from the Organization Data menu. The organization data items are: Organization Information, Calendars, Time Zones, Locations, Currencies, Resources, Organization Units, Resource Allocation, Chart of Accounts, and External Entities. With this functionality, it is possible to model the organization to which the process belongs.

- **Process data** - These are data that pertain to the attributes of business processes. It is possible to create, modify, or access process data by using commands from the Process Data menu. The process items are: Activities, External Processes, Phi Types, Phis, Phi States, Media, Classifications, Delay Reasons, Variables, Authorizations, Decisions, and Choices. The business analyst can use this process data in various process models. Once a data record has been created (in the repository or in a process model), this record is stored in the repository and it is available to other process models.
The process model created using WebSphere Business Integration Workbench is called an Activity Decision Flow Diagram. It represents the flow of the process. The objects in the Activity Decision Flow Diagram are: Tasks, Process Objects, decisions, decision choices, inputs/outputs, external entity or process, and connectors.

WebSphere Business Integration Workbench has specific instruments in its Activity Decision Flow toolbar to create the objects required for building the business process model's diagram:

- **Process** - The Process (square) tool is used to insert a process into Activity Decision Flow Diagrams.
- **Task** - The Task (rectangle) tool is used to insert a task into Activity Decision Flow Diagrams.
- **Input/Output** - The Input/Output (greek letter Phi) tool is used to insert a Phi, the input/output between activities in Activity Decision Flow Diagrams.
- **Connector** - The Connector (arrow) tool is used to sequentially link objects in Activity Decision Flow Diagrams.
- **External Entity** - The External Entity (oval) tool is used to insert external entities and external processes. An external process is an activity that is performed by an external entity.
- **Decision** - The Decision (diamond) tool is used to insert decisions into Activity Decision Flow Diagrams. This decision can be binary or multiple.
- **Decision Choice** - The Decision Choice (octagon) tool is used to insert choices into Activity Decision Flow Diagrams. A choice must be linked to a decision.
- **Activity Group** - When an activity is selected with this tool, it will be converted to an Activity Group Object.
- **Partner Interaction** - The Partner Interaction tool is used to insert Partner Interaction objects into Activity Decision Flow Diagrams.
- **Stop** - The Stop (traffic stop sign) tool is used to insert discontinuances into Activity Decision Flow Diagrams.
- **Go To** - The Go To (star) tool is used to insert Go To objects into Activity Decision Flow Diagrams.

After business process is modeled, it is possible to analyze it in terms of cases. Process cases are determined by decisions, and each decision creates a fork in the road or path of activities. Each path of activities can be executed when a process is performed. The option that is selected when a decision occurs determines which fork or path of activities will be performed. Each option creates a single road through the process, with only one road being followed at a time.
These individual paths are called cases. WebSphere Business Integration Workbench provides the functionality of individualizing and analyzing the specific paths separately from the process’s complexity.

WebSphere Business Integration Workbench allows data from each case to be visualized and reviewed in different formats. The three types of formats are tables, charts, and diagrams.

In Figure 12-4, a typical view of a WebSphere Business Integration Workbench Activity Decision Flow diagram with Activity Decision Flow toolbar is shown.
An example of a UML activity diagram created with WebSphere Business Integration Workbench is presented in Figure 12-5.

12.4.3 Business process model

For the real estate agency, two different process models need to be created. In both cases, the process model presented refers to current integrated systems. The first model refers to the Register New Property process. The second model refers to the Sale Process.

- Register New Property Process Model

  This is a model of the process whereby the agent registers a new property using SAL301R and this registration feeds into SAB404R. The model is presented in the Figure 12-6 on page 439.
As demonstrated in Figure 12-6, after registration, the integration solution will update SAB404R, putting the property registration in inventory. Since SAB404R is part of the company's internal IT structure, this connectivity can be provided by a Java Database Connectivity (JDBC) adapter. When SAB404R is updated, this means that it is ready to output an inventory report.

When a new property is registered, the newspaper automatically receives an electronic request to advertise that it is for sale. Since this connection is external, the connectivity is provided by a Web Services adapter.

Sale Process Model

This is a model of the property sale process. It starts when the buyer looks for a property using SAL301R and ends at the moment when the transaction is considered complete. In this model, all decisions in the process are considered. During this process, the property status in SAL301R and SAB404R can change several times and so can the sale price. These changes that are related to the sales process must be correctly represented.
in both systems and will be done with the implementation of a trigger in SAL301R’s database. The installation of this trigger is based on the fact that any change entered by the agent is executed using SAL301R. When a transaction fails, the integration solution must be alerted to compensate for all activities that generated any output. The model is presented in the figures that follow.

Figure 12-7   Sales Process Model I

Figure 12-7 illustrates the activity when a potential buyer logs into SAL301R. The first process decision is related to the status of this customer. If the customer is not registered, he or she can choose to register or leave. If the customer registers, then the integration application updates SAB404R in an electronic document transaction. If the customer is registered (or has just registered), the property search activity can be done.

The next process decision is making a bid on a property. When the customer finds an interesting property that meets his or her requirements, then the customer can make a bid as shown in Figure 12-8 on page 441.
If a customer does not bid on a property, he or she can decide whether to receive notifications from the agent about other properties. The agent, then, will send updates to the customer. If the customer bids on a property using SAL301R, then the agent receives a notification to analyze the proposal.

The next process decision occurs after the proposal is examined. If the buyer’s offer is interesting to the agent and to the company, then the agent notifies the seller about the bid.
If the agent or company does not accept the buyer’s bid, then the buyer can make another offer or quit the process as shown in Figure 12-9.

When the seller receives the bid, he or she can accept it or not. If the seller does not accept the bid, the agent notifies the buyer, and the buyer can make another offer or quit the process.

If the seller accepts the bid, then the sales process continues, and the escrow process is started. SAB404R automatically sends the notification to the escrow agent that a new process is starting up, and the seller and buyer documents will be released. At this time, the process is in escrow, and, in SAB404R, the property status changes from Available to Under Escrow.
After the escrow analysis process, it is time for the next important decision: Does the buyer have good credit? If the buyer does, then the property status changes from Under Escrow to Sold. The system sends e-mails to the buyer and seller to notify them the transaction was successful. If the buyer does not have good credit, then the agent contacts the buyer and seller to deliver the news. This is shown in Figure 12-10 and Figure 12-11 on page 444.
The requirements of this integration system are focused on automating as much as possible, from the interaction of the actors in the process to making all property status, customers and employees changes. This functionality will give the real estate managers the confidence that no potential buyer is being forgotten during the process. In the same way, this integration will give the company the assurance that all properties offered for sale are registered in SAL301R and that their registrations are adequately reflected in SAB404R.

One thing that is quite important is how to deal adequately with the business actors (sellers and buyers) when the transaction is not successful. If these customers are not treated well, now they may go to another real estate agency in the future and do business with someone else. However, it is natural for the agents to focus more on transactions that have been successful and pay less attention to the those that are not complete.
Figure 12-12  Sale Process Model VI

This integration solution will provide an appropriate way to treat buyers and sellers when the transaction is not successful. Therefore, this process model gives importance to the description of the activities that involve notifying customers, as shown in Figure 12-12.
Figure 12-13 is a summary of the process described during the modeling process that shows the major activities of the process and each connection between them.

As we stated previously, WebSphere Business Integration Workbench provides the opportunity to isolate and analyze each process case, validate the process, and evaluate the important aspects of the behavior in each case. WebSphere Business Integration Workbench can help bring up relevant aspects that characterize a case as a percentage of occurrence, cost, and cycle time.
For each case, a separate Activity Decision Flow Diagram that contains only the activities that occur in the path of that case can be generated.

The Register New Property Process model has only one process case. The Sale Process has more than one. In this particular business process model, there are ten cases that can be traced in the Process Activity Decision Flow Diagram. However, for processes that have thousands of cases, this analysis is just as useful.

<table>
<thead>
<tr>
<th>User Registered?</th>
<th>Want to bid on property?</th>
<th>Want to add to your interests?</th>
<th>Do you want notifications?</th>
<th>Is bid approved?</th>
<th>Accept Bid?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Registered User</td>
<td>yes</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>New User</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Registered User</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>Registered User</td>
<td>no</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Registered User</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Registered User</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Registered User</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Registered User</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Registered User</td>
<td>yes</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Registered User</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 12-14  Process Cases I**

Figure 12-14 and Figure 12-15 present all the Sale Process model's cases. The relevant aspect that drives this process is the role played by the customer searching for a transaction (buyer). This makes sense when you consider that this customer is the person that the real estate agency wants to reach because he or she is the driver of profitability for the real estate agency.

<table>
<thead>
<tr>
<th>Accept Bid?</th>
<th>Change Bid?</th>
<th>Change Bid? 01</th>
<th>Does Buyer have Credit?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 12-15  Process cases II**
12.5 System integration design

To completely meet the solution objectives, a complete integration between SAL301R application (the Web application) and SAB404R system (legacy system) must be implemented. This integration will ensure synchronism between SAL301R’s database and legacy system’s database. This integration will automate some activities that are currently executed manually. The data synchronism will give the customers, employees and, managers confidence in the system and process. In the end, this system integration will leverage the company’s income and profit because of faster and better customer service.

To deploy this integration, we use WebSphere Business Integration Server Express, because it is middleware that is well designed for use by SMBs. WebSphere Business Integration Server Express provides adapters and process templates that enable a company to tie applications and systems without developing something new, such as an interface that would become one more application to patch and maintain. Instead, the company can use and take advantage of existing IT structure, thereby leveraging its technology investments. WebSphere Business Integration Server Express provides a back-end integration solution with easy and fast implementation and the flexibility to use or create an array of adapters required for integration.

This product has the following characteristics:

- Flexibility for a business integration solution, growing with the customer’s business needs with affordable per-processor pricing
- Deployability of adapters and collaboration tools that simplify implementation
- Infrastructure flexibility, running on multiple platforms
- Connectivity to a wide range of back-end systems
- Availability of different templates that speed the process implementation
Figure 12-16 is a representation of the role of WebSphere Business Integration Server Express in the integration solution.

The most important component of WebSphere Business Integration Server Express, the Interchange Server Express - ICS, enables the automation and synchronization of business activities executed by different and multiple applications. As for the functionality of executing business processes concurrently, Interchange Server Express can deal with transactions and interchanges efficiently and control how transactions are executed according to previously defined business rules.

The Interchange Server Express model is an efficient way to maintain control over several systems at the same time. The connecting application architecture on the server imparts scalability by enabling the simultaneous processing of the same business events.
12.5.1 Methodology - system integration development using Interchange Server

The methodology involved in system integration development using Interchange Server has the following steps:

1. Design the collaboration process. The first step is the process design, including process modeling, process analysis, and solution requirements definition. In this step, it is necessary to specify the collaboration templates that will support business processes and whether the flows will be designed. In this step, you also identify the Business Objects, maps, and connectors that will be used for building the collaborations.

2. Design and build Business Object definitions and mapping. Interchange Server provides the functionality for creating Business Objects easily and associating them with collaboration templates.

3. Build the collaborations. Collaborations are software modules written in Java that describe business processes. Collaboration objects are runtime objects and collaboration templates are entities that reside in the Interchange Server repository. These actions need to be taken during the process of building the collaborations:

   a. Create different collaborations. This has two activities: Creating collaboration templates and editing properties. The properties, port, and triggering information for a collaboration template can be modified using the Process Designer tool.

   Properties allow the InterChange Server to determine a collaboration's runtime behavior and can be categorized as standard or specific. Standard properties are those typically available to all collaborations. Standard properties are filtering, error handling, and synchronization.

   Ports are variables that represent the Business Objects that the collaboration object receives or releases. They are bound to application connectors or to other collaborations in the form of collaboration groups.

   b. Implement scenario and activity diagrams. These scenarios contain code by which a collaboration implements the business rules and processes. The operations performed by scenarios normally involve obtaining and manipulating Business Objects.

   c. Implement error and message error handling.
12.5.2 Sample design

In the sample integration solution for the real estate agency, WebSphere Business Integration Server Express - InterChange Server is positioned as shown in Figure 12-17.

![ICS role in the integration solution](image)

As illustrated, Interchange Server will be the intermediary between the Web application and the legacy system and will provide the connectivity to the escrow agent and newspaper.

The middleware that will be deployed in this example is based on fact that all data regarding customers, properties, employees will be transferred from one application to another with as many of the business rules defined as necessary. The following types of connectors will be used to deploy this data synchronization:

- JDBC adapters
- Web services adapters
- E-mail adapters

The JDBC adapters will integrate the Web application (SAL301R) and the legacy system (SAB404R) through Interchange Server Express. These adapters are all Application Specific Business Object (ASBOs). The connection to the newspaper that asks for a property announcement will be deployed by the Web Services adapter. The interaction with customers (buyers and sellers) will be provided by an e-mail adapter. The data transactions will be handled by mapping process definitions in the middleware development.
The integration between SAL301R’s database and SAB404R’s database will be done through the collaboration with the following Business Objects: JDBC adapters, e-mail adapters, and Web Services adapters.

Generic Business Objects (GBOs) will allow data sharing and transferences.

To ensure that all data move from one application to another, specific triggers will be implemented in each database to activate the updating process. Figure 12-18 is a diagram of the integration solution.

![Diagram of Integration System Design with ICS](image)

Figure 12-18 Integration system Design with ICS

We will now describe the required Business Objects in this system integration in detail.

A Business Object can be defined as a representation of an activity of the business domain, including, for example, its business name, definition, attributes, behavior, relationships, and rules.
Business Objects are elements that facilitate data exchange. When they are associated with an application they are called ASBOs. GBOs exist within Interchange Server. Every Business Object that will be transferred between applications typically needs an associated map. Maps are parts (Java code) that execute data transformation. These transformations define attribute changes from the source Business Object to the destination Business Object.

Business Objects are created or instantiated at runtime. They are not stored in the Interchange Server repository. In the definition activity of a Business Object, it is possible to specify information as well as the attributes that are used to define the objects. The definition also specifies the verbs that the object supports, such as create, retrieve, update, and delete.

Interfaces such as JDBC allow access to database manipulation and update commands. Library routines, interfaced with the database, integrate a data call into a general programming environment.

JDBC has three specific roles:
► Establish a connection with a database.
► Send SQL statements.
► Process the results.

The advantages of JDBC technology are:
► Leveraged company data - With JDBC technology, businesses are not confined to any specific and proprietary architecture. It is possible to continue to use the installed databases and access information, even in different database management systems.
► Simplified solution development - Application development is easy and cost effective, because it encapsulates the complexity of many data access tasks, doing the hard work behind the scenes.
► Simplified solution deployment - no configuration of network computers is necessary.

Collaborations are where the business process logic resides. They work within the Interchange Server and house the business processes and intelligence. These business rules coordinate actions, data movement, and processing entities. The data collaboration exchange is a form of Business Object. Data transformation and coordination resides in the collaborations so that integration takes place at the business process level, not at the application level.
12.5.3 Sample - Business Objects

Figure 12-19 shows all the Business Objects created with Interchange Server Express for the sample integration to achieve the objectives defined for the real estate agency.

![Business Objects created to sample implementation](image)

Figure 12-19   Business Objects created to sample implementation

Figure 12-20 presents the EmailInventory object. It is related to the customer notification activity. It was created to allow buyers and sellers receive an e-mail when the property transaction is completed. That is the reason why the InventoryId field is key.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventoryId</td>
<td>Integer</td>
</tr>
<tr>
<td>EmailContent</td>
<td>Email</td>
</tr>
<tr>
<td>RecipientName</td>
<td>String</td>
</tr>
<tr>
<td>ToAddress</td>
<td>String</td>
</tr>
<tr>
<td>MessageSubject</td>
<td>String</td>
</tr>
<tr>
<td>MessageContent</td>
<td>String</td>
</tr>
<tr>
<td>FromAddress</td>
<td>String</td>
</tr>
</tbody>
</table>

Figure 12-20   Business Object - EmailInventory
The GBOs created to handle properties, employees, and addresses within Interchange Server are shown below. These objects will be filled with data from the ASBOs in a mapping process.

The Figure 12-21 presents the Address object. It was created to gather all address information that is important to the company that will be used during the sale process.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressId</td>
<td>Integer</td>
</tr>
<tr>
<td>ObjectEventId</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-21  Business Object - Address*

Figure 12-22 presents the Employee object. This object provides all relevant information about the company’s employees, and EmployeeId is the key field.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeId</td>
<td>String</td>
</tr>
<tr>
<td>EmployeeNumber</td>
<td>String</td>
</tr>
<tr>
<td>EmpEffectiveOnDate</td>
<td>String</td>
</tr>
<tr>
<td>EmpEffectiveTillDate</td>
<td>String</td>
</tr>
<tr>
<td>Prefix</td>
<td>String</td>
</tr>
<tr>
<td>LastName</td>
<td>String</td>
</tr>
<tr>
<td>FirstName</td>
<td>String</td>
</tr>
<tr>
<td>MiddleName</td>
<td>String</td>
</tr>
<tr>
<td>Suffix</td>
<td>String</td>
</tr>
<tr>
<td>SocialNumber</td>
<td>String</td>
</tr>
<tr>
<td>WorkPhone</td>
<td>String</td>
</tr>
<tr>
<td>WorkPhoneExt</td>
<td>String</td>
</tr>
<tr>
<td>HomePhone</td>
<td>String</td>
</tr>
<tr>
<td>Fax1</td>
<td>String</td>
</tr>
<tr>
<td>Fax2</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-22  Business Object - Employee*
The Business Objects related to employee information are presented in Figure 12-23 through Figure 12-26 on page 459. These are the Business Objects: EmployeeAddress, EmployeeHR_Info, EmployeeHR.Misc, and EmployeeJob.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employeeld</td>
<td>String</td>
</tr>
<tr>
<td>AddressId</td>
<td>String</td>
</tr>
<tr>
<td>AddrEffectiveOnDate</td>
<td>String</td>
</tr>
<tr>
<td>AddrEffectiveTillDate</td>
<td>String</td>
</tr>
<tr>
<td>AddressType</td>
<td>String</td>
</tr>
<tr>
<td>AddressLine1</td>
<td>String</td>
</tr>
<tr>
<td>AddressLine2</td>
<td>String</td>
</tr>
<tr>
<td>AddressLine3</td>
<td>String</td>
</tr>
<tr>
<td>AddressLine4</td>
<td>String</td>
</tr>
<tr>
<td>City</td>
<td>String</td>
</tr>
<tr>
<td>County</td>
<td>String</td>
</tr>
<tr>
<td>State</td>
<td>String</td>
</tr>
<tr>
<td>Region</td>
<td>String</td>
</tr>
<tr>
<td>PostalCode</td>
<td>String</td>
</tr>
<tr>
<td>Country</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-23  Business Object - EmployeeAddress*
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeId</td>
<td>String</td>
</tr>
<tr>
<td>PersonnelStatus</td>
<td>String</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>String</td>
</tr>
<tr>
<td>BirthPlace</td>
<td>String</td>
</tr>
<tr>
<td>BirthCountry</td>
<td>String</td>
</tr>
<tr>
<td>BirthState</td>
<td>String</td>
</tr>
<tr>
<td>DateOfDeath</td>
<td>String</td>
</tr>
<tr>
<td>Gender</td>
<td>String</td>
</tr>
<tr>
<td>Age</td>
<td>String</td>
</tr>
<tr>
<td>MaritalStatus</td>
<td>String</td>
</tr>
<tr>
<td>MaritalStatusDate</td>
<td>String</td>
</tr>
<tr>
<td>PreferredLanguage</td>
<td>String</td>
</tr>
<tr>
<td>OfficialLanguage</td>
<td>String</td>
</tr>
<tr>
<td>MaxEducation</td>
<td>String</td>
</tr>
<tr>
<td>ImmigrationStatus</td>
<td>String</td>
</tr>
<tr>
<td>Nationality</td>
<td>String</td>
</tr>
<tr>
<td>EthnicGroup</td>
<td>String</td>
</tr>
<tr>
<td>MilitaryStatus</td>
<td>String</td>
</tr>
<tr>
<td>DisabledFlag</td>
<td>String</td>
</tr>
<tr>
<td>VeteranFlag</td>
<td>String</td>
</tr>
<tr>
<td>SmokerFlag</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-24  Business Object - EmployeeHR_Info*

Figure 12-24 represents the Business Object for human resources that contains the employee information that is most likely to be captured and recorded when an employee is hired.
Figure 12-25 shows the Business Object that represents the miscellaneous information about an employee that is needed by Human Resources.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeHRMiscId</td>
<td>String</td>
</tr>
<tr>
<td>HireDate</td>
<td>String</td>
</tr>
<tr>
<td>ReEntryDate</td>
<td>String</td>
</tr>
<tr>
<td>SeniorityDate</td>
<td>String</td>
</tr>
<tr>
<td>LastWorkedDate</td>
<td>String</td>
</tr>
<tr>
<td>ServiceDate</td>
<td>String</td>
</tr>
<tr>
<td>TerminationDate</td>
<td>String</td>
</tr>
<tr>
<td>ExpectedReturnDate</td>
<td>String</td>
</tr>
<tr>
<td>LastIncreaseDate</td>
<td>String</td>
</tr>
<tr>
<td>BusinessTitle</td>
<td>String</td>
</tr>
<tr>
<td>SupervisorId</td>
<td>String</td>
</tr>
<tr>
<td>ProbationDate</td>
<td>String</td>
</tr>
<tr>
<td>PayChkDistOption</td>
<td>String</td>
</tr>
<tr>
<td>PayChkAddrOptn</td>
<td>String</td>
</tr>
<tr>
<td>PayChkLocnOptn</td>
<td>String</td>
</tr>
<tr>
<td>PayChkLocnCode</td>
<td>String</td>
</tr>
<tr>
<td>PayChkName</td>
<td>String</td>
</tr>
<tr>
<td>MailDropld</td>
<td>String</td>
</tr>
</tbody>
</table>

Figure 12-25  Business Object - EmployeeHRMisc
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeJobId</td>
<td>String</td>
</tr>
<tr>
<td>JobEffectiveOnDate</td>
<td>String</td>
</tr>
<tr>
<td>JobEffectiveTillDate</td>
<td>String</td>
</tr>
<tr>
<td>EffectiveSequence</td>
<td>String</td>
</tr>
<tr>
<td>Action</td>
<td>String</td>
</tr>
<tr>
<td>ActionDate</td>
<td>String</td>
</tr>
<tr>
<td>ActionReason</td>
<td>String</td>
</tr>
<tr>
<td>JobCode</td>
<td>String</td>
</tr>
<tr>
<td>JobTitle</td>
<td>String</td>
</tr>
<tr>
<td>EmployeeStatus</td>
<td>String</td>
</tr>
<tr>
<td>PositionCode</td>
<td>String</td>
</tr>
<tr>
<td>CompanyCode</td>
<td>String</td>
</tr>
<tr>
<td>PersonnelArea</td>
<td>String</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>String</td>
</tr>
<tr>
<td>TaxLocationCode</td>
<td>String</td>
</tr>
<tr>
<td>PayGroup</td>
<td>String</td>
</tr>
<tr>
<td>Shift</td>
<td>String</td>
</tr>
<tr>
<td>ShiftRate</td>
<td>String</td>
</tr>
<tr>
<td>ShiftFactor</td>
<td>String</td>
</tr>
<tr>
<td>EmployeeClassification</td>
<td>String</td>
</tr>
<tr>
<td>RegularEmpFlag</td>
<td>String</td>
</tr>
<tr>
<td>FullTimeEmpFlag</td>
<td>String</td>
</tr>
<tr>
<td>StandardHours</td>
<td>String</td>
</tr>
<tr>
<td>EEOClassification</td>
<td>String</td>
</tr>
<tr>
<td>FTE_Rate</td>
<td>String</td>
</tr>
<tr>
<td>GLPayType</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-26  Business Object - EmployeeJob*

Figure 12-26 shows all the important information about the employee's job.
Figure 12-27 shows all the important information about properties within the Property object. This GBO will allow SAB404R to be updated.

ASBOs were created to move data to and from the specific applications: the Web application (SAL301R) and the legacy system (SAB404R). As we stated earlier, these ASBOs are adapters within Interchange Server and it will provide data integration required in this solution.

Figure 12-28 shows the SAB404RAddress Business Object. This JDBC object contains address data in any role that is relevant to the company. It is related to all address data within the legacy system (SAB404R).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>INV_ID</td>
<td>String</td>
</tr>
<tr>
<td>INV_STATUS</td>
<td>String</td>
</tr>
<tr>
<td>INV_TYPE</td>
<td>String</td>
</tr>
<tr>
<td>INV_AGENT_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>INV_ADDR_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>INV_ASK_PRICE</td>
<td>String</td>
</tr>
<tr>
<td>INV_SALE_PRICE</td>
<td>String</td>
</tr>
<tr>
<td>INV_DATE_LISTED</td>
<td>Date</td>
</tr>
<tr>
<td>INV_DATE_SOLD</td>
<td>Date</td>
</tr>
<tr>
<td>INV_DESC</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-27  Business Object - Property*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>ADDR_TYPE</td>
<td>String</td>
</tr>
<tr>
<td>ADDR_OWNER</td>
<td>Integer</td>
</tr>
<tr>
<td>ADDR_LINE1</td>
<td>String</td>
</tr>
<tr>
<td>ADDR_LINE2</td>
<td>String</td>
</tr>
<tr>
<td>ADDR_CITY</td>
<td>String</td>
</tr>
<tr>
<td>ADDR_ZIP</td>
<td>String</td>
</tr>
<tr>
<td>ADDR_STATE</td>
<td>String</td>
</tr>
<tr>
<td>ADDR_COUNTRY</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-28  Business Object - SAB404RAddress*
Figure 12-29 shows the SAB404REmployee object. This JDBC object contains all of the company’s employee data. It is related to all important employee information in the legacy system (SAB404R).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>EMP_TYPE</td>
<td>String</td>
</tr>
<tr>
<td>EMP_FIRST_NAME</td>
<td>String</td>
</tr>
<tr>
<td>EMP_LAST_NAME</td>
<td>String</td>
</tr>
<tr>
<td>EMP_DATE_HIRED</td>
<td>Date</td>
</tr>
<tr>
<td>EMP_ID</td>
<td>String</td>
</tr>
<tr>
<td>EMP DOB</td>
<td>Date</td>
</tr>
<tr>
<td>EMP SEX</td>
<td>String</td>
</tr>
<tr>
<td>EMP TAX_ID</td>
<td>String</td>
</tr>
<tr>
<td>EmployeeAddress</td>
<td>SAB404RAddress</td>
</tr>
</tbody>
</table>

Figure 12-29  Business Object - SAB404REmployee

Figure 12-30 shows the SAB404RInventory object. This JDBC object carries the property information important to the agency. It is related to important data about the company’s inventory that resides in the legacy system (SAB404R).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>INV_ID</td>
<td>String</td>
</tr>
<tr>
<td>INV_STATUS</td>
<td>String</td>
</tr>
<tr>
<td>INV_TYPE</td>
<td>String</td>
</tr>
<tr>
<td>INV_AGENT_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>INV_ADOR_KEY</td>
<td>Integer</td>
</tr>
<tr>
<td>INV_ASK_PRICE</td>
<td>String</td>
</tr>
<tr>
<td>INV_SALE_PRICE</td>
<td>String</td>
</tr>
<tr>
<td>INV_DATE_LISTED</td>
<td>Date</td>
</tr>
<tr>
<td>INV_DATE_SOLD</td>
<td>Date</td>
</tr>
<tr>
<td>INV_DESC</td>
<td>String</td>
</tr>
</tbody>
</table>

Figure 12-30  Business Object - SAB404RInventory

Figure 12-31 illustrates the adapter related to the integration solution for the escrow agent. This object is not well defined because the specifics of integration with this actor will not be covered in this book.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventoryId</td>
<td>Integer</td>
</tr>
<tr>
<td>ObjectEventId</td>
<td>String</td>
</tr>
</tbody>
</table>

Figure 12-31  Business Object - SAB404EscrowInventory
Figure 12-32 presents the ASBO that will allow send e-mails to sellers and buyers. As described before this integration will require an e-mail adapter to provide this connection.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventoryId</td>
<td>Integer</td>
</tr>
<tr>
<td>ObjectEventId</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-32 Business Object - SAB404RNewsInventory*

The next set of Business Objects are related to the Web application (SAL301R). This solution is described in *WebSphere Business Integration Server Express - A Development Example for New developers - SG 246301*. The figures that follow show the objects that will activate the integration system proposed in this work.

SAL301RPROP, shown in Figure 12-33, contains the property data that the agent generates in SAL301R. The information that will be moved to SAB404R is contained in these fields. This is a JDBC object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPID</td>
<td>Integer</td>
</tr>
<tr>
<td>USRID</td>
<td>Integer</td>
</tr>
<tr>
<td>Property&gt;Type</td>
<td>SAL301RPRCP_TYPE</td>
</tr>
<tr>
<td>PROPTYPEID</td>
<td>Integer</td>
</tr>
<tr>
<td>PROPTYPEDESC</td>
<td>String</td>
</tr>
<tr>
<td>ObjectEventId</td>
<td>String</td>
</tr>
<tr>
<td>Property&gt;Status</td>
<td>SAL301RPRCP_STS</td>
</tr>
<tr>
<td>PROPSTSID</td>
<td>Integer</td>
</tr>
<tr>
<td>PROPSTSDESC</td>
<td>String</td>
</tr>
<tr>
<td>ObjectEventId</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRCNTRYCD</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRPOSTCD</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRRON</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRMUNI</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRSTRTNAM</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRBLDGNUM</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRUNITIUM</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRPOBOX</td>
<td>String</td>
</tr>
<tr>
<td>PROPADDRADDLTEX</td>
<td>String</td>
</tr>
<tr>
<td>PROPPRICE</td>
<td>String</td>
</tr>
<tr>
<td>PROPPICURL</td>
<td>String</td>
</tr>
<tr>
<td>PROPCRDON</td>
<td>Date</td>
</tr>
</tbody>
</table>

*Figure 12-33 Business Object - SALRPROP*
Figure 12-34 shows the SAL301PROP_STS object. This object will integrate property status changes in SAL301R's database with SAB404R. Any activity that changes a property's status, such as a new listing in inventory (Available), a bid process (Under Escrow), and a successful transaction (Sold), will be reflected in SAB404R through this ASBO.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPSTSID</td>
<td>Integer</td>
</tr>
<tr>
<td>PROPSTSDESC</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-34 Business Object - SAL301PROP_STS*

When a property is listed in SAL301R, a description of the property is required because customer searches are description driven. The object shown in Figure 12-35 provides the integration system with these descriptions. The PROPTYPEDESC field is a string that allows property specification. SAL301PROP_TYPE is a JDBC object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPTYPEED</td>
<td>Integer</td>
</tr>
<tr>
<td>PROPTYPEDESC</td>
<td>String</td>
</tr>
</tbody>
</table>

*Figure 12-35 Business Object - SAL301RPROP_TYPE*

Figure 12-36 illustrates the SAL301RUSR object. This JDBC object contains information about the application’s users.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRID</td>
<td>Integer</td>
</tr>
<tr>
<td>USRUSERNAME</td>
<td>String</td>
</tr>
<tr>
<td>USRPASSWORD</td>
<td>String</td>
</tr>
<tr>
<td>USRROLE</td>
<td>String</td>
</tr>
<tr>
<td>USRFIRSTNAME</td>
<td>String</td>
</tr>
<tr>
<td>USRLASTNAME</td>
<td>String</td>
</tr>
<tr>
<td>USREMAILADDR</td>
<td>String</td>
</tr>
<tr>
<td>USRMEMURL</td>
<td>String</td>
</tr>
<tr>
<td>USRPHNINUM</td>
<td>String</td>
</tr>
<tr>
<td>USRACTIVE</td>
<td>String</td>
</tr>
<tr>
<td>USRCDNON</td>
<td>Date</td>
</tr>
<tr>
<td>Address</td>
<td>SAL301RUSR_ADDR</td>
</tr>
</tbody>
</table>

*Figure 12-36 SAL301RUSR*

Figure 12-37 on page 464 shows the SAL301RUSR_ADDR object. This JDBC object carries the information about the users' addresses.
12.5.4 Sample - collaborations

The collaborations that will be implemented are:

- **Employee_Sync**
  
  This collaboration is related to the synchronization of employee data between the Web application (SAB301R) and the legacy system (SAB404R).

- **Inventory_Sync**
  
  This collaboration is related to the synchronization of property inventory data between the Web application (SAB301R) and the legacy system (SAB404R). This specific collaboration has different events that must be implemented:
  - **Inventory_Create** - This event is related to the synchronization of data between the databases when a property is entered in the company’s inventory.
  - **Inventory_Update** - This event is related to the synchronization of data between databases when specific property information such as price is changed. This event handles the property status change when one of these events occurs:
    - **Inventory_Sold** - this event occurs when a property is sold to a buyer.
    - **Inventory_Escrow** - this event occurs when a seller accepts a buyer’s bid and the escrow process is started up.

The Web Service adapter that will be implemented is:

```
News_Inventory  ->  News_Address
```
This handles the notification event that happens when a bid is accepted by a buyer and the property is sold.

The e-mail adapters that will be implemented are:

- Email_Inventory -> Email_Address
- Email_Inventory -> Email_Sales_Info
- Email_Inventory -> Email_Agent_Info

These adapters will allow the agent, buyers, and sellers to receive notification regarding property transactions.

The interactions between the Web application (SAL301R), the legacy system (SAB404R), and the databases within the Interchange Server are provided by connectors. The next figures illustrate the configuration of these connectors.

The configuration of the Web application (SAL301R) connector is shown in Figure 12-38 on page 466, including the connector's standard properties, the values of these properties, and the update method configured to it.
Figure 12-38  ICS - SAL301R Connector
The configuration of the legacy system (SAB404R) connector is illustrated in Figure 12-39, including the connector's standard properties, the values of these properties, and the update method configured to it.

![Figure 12-39 ICS - SAB404R Connector](image-url)

---

**Table: Standard Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Update Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminQueue</td>
<td>SAB404R.CONNECTOR/ADMINQUEU</td>
<td>component restart</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>SAB404R.CONNECTOR/ADMINOUTQUEU</td>
<td>component restart</td>
</tr>
<tr>
<td>AgentTraceLevel</td>
<td>5</td>
<td>dynamic</td>
</tr>
<tr>
<td>ApplicationName</td>
<td>SAB404RConnector</td>
<td>component restart</td>
</tr>
<tr>
<td>BrokerType</td>
<td>ICS</td>
<td>component restart</td>
</tr>
<tr>
<td>CharacterEncoding</td>
<td>437</td>
<td>component restart</td>
</tr>
<tr>
<td>ConsumerEventHandlerFlows</td>
<td>1</td>
<td>component restart</td>
</tr>
<tr>
<td>ContainerManagedEvents</td>
<td>1</td>
<td>component restart</td>
</tr>
<tr>
<td>ControllerStoreAndForwardMode</td>
<td>true</td>
<td>dynamic</td>
</tr>
<tr>
<td>ControllerTraceLevel</td>
<td>5</td>
<td>dynamic</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>SAB404R.CONNECTOR/DELIVERYQUEU</td>
<td>component restart</td>
</tr>
<tr>
<td>DeliveryTransport</td>
<td>JMS</td>
<td>component restart</td>
</tr>
<tr>
<td>DuplicateEventElimination</td>
<td>true</td>
<td>component restart</td>
</tr>
<tr>
<td>EnabledStateBasedForwarding</td>
<td>true</td>
<td>component restart</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>SAB404R.CONNECTOR/FAULTQUBEU</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsFactoryClassName</td>
<td>CoCommon.Messaging.Ims.EMQGswi</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsMessageBrokerName</td>
<td>WebSphereIcs.queue.manager</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsNumConsumerRequests</td>
<td>10</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsPassword</td>
<td>********</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsUserName</td>
<td></td>
<td>component restart</td>
</tr>
<tr>
<td>JmsMaxQueueSize</td>
<td>128k</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsMaxInflightSize</td>
<td>128k</td>
<td>component restart</td>
</tr>
<tr>
<td>JmsMaxInflightSize</td>
<td>1m</td>
<td>component restart</td>
</tr>
<tr>
<td>Locale</td>
<td>en_US</td>
<td>component restart</td>
</tr>
<tr>
<td>LogMaxInterchangeEnd</td>
<td>true</td>
<td>component restart</td>
</tr>
<tr>
<td>MaxEventCapacity</td>
<td>2147483647</td>
<td>dynamic</td>
</tr>
<tr>
<td>MessageIdSchema</td>
<td>SAS404RConnector.msg</td>
<td>component restart</td>
</tr>
<tr>
<td>OwrdAutoRestartAgent</td>
<td>false</td>
<td>dynamic</td>
</tr>
<tr>
<td>OwrdInitialDelay</td>
<td>1000</td>
<td>dynamic</td>
</tr>
<tr>
<td>OwrdRetryTimeinterval</td>
<td>10</td>
<td>dynamic</td>
</tr>
<tr>
<td>PollEndTime</td>
<td>HHHMM</td>
<td>component restart</td>
</tr>
<tr>
<td>PollFrequency</td>
<td>10000</td>
<td>dynamic</td>
</tr>
<tr>
<td>PollStartTime</td>
<td>HHHMM</td>
<td>component restart</td>
</tr>
<tr>
<td>RepositoryDirectory</td>
<td>&lt;REMOTE&gt;</td>
<td>agent restart</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>SAB404R.CONNECTOR/REQUESTQ</td>
<td>component restart</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>SAB404R.CONNECTOR/RESPONSEQ</td>
<td>component restart</td>
</tr>
<tr>
<td>RequestRetryCount</td>
<td>3</td>
<td>dynamic</td>
</tr>
<tr>
<td>RequestRetryInterval</td>
<td>1</td>
<td>dynamic</td>
</tr>
<tr>
<td>SynchronousRequestQueue</td>
<td>SAB404R.CONNECTORSYNCHRONQ</td>
<td>component restart</td>
</tr>
</tbody>
</table>
Note: The solution requirements discussed in this chapter are for a realistic business process, wherein each activity and role is typical of real business. However, considering that the sample presented in this book had to be released quickly, and we took advantage of the existing Web application, some of the functionality foreseen in this requirements chapter were not implemented in this sample. These are:

- Escrow notification process - This should be enabled with JMS technology.
- Buyer and seller e-mail notification - In accordance with the design of SAL301R (the Web application) and due to the lack of this functionality with the application available, the e-mail notification process was changed so that only the sales manager received e-mail notification.
- Compensation process.
Sample development

This chapter describes how to develop and implement our redbook sample solution as specified in Chapter 12, “Sample requirements, design, and specification” on page 425. The following sections are included:

- 13.3, “Preparing the development environment” on page 471.
- 13.6, “SAL301R to SAB404R: property synchronization” on page 548.
- 13.8, “Additional material, tips and tricks” on page 592.

In this chapter, we give you an account of the tasks we used to develop our redbook sample solution and we also include additional material that may be of value to developers.

This development chapter concentrates on the development tasks, while a discussion of the requirements for our sample can be found in Chapter 12, “Sample requirements, design, and specification” on page 425. Prerequisite installation material is referred to as necessary.
13.1 The sample problem

The process of developing a redbook sample is, by definition, artificial. For our sample, we had goals as to what to show. Like developers on any project, we had constraints on time and the skills of our development resources.

The SAL301R WebSphere Application Server - Express sample solution was chosen as the system on one side of our integration project since it was readily available and solved a business problem most people could relate to. It was not difficult to imagine that the enterprise using this system also had additional systems purchased from different vendors at different times. It was also easy to imagine the kinds of manual processes, such as double entry, that would exist to provide integration between the front-end Web system and other legacy systems.

We imagined an Inventory system for the stock control and reporting on the real estate being sold and a Personnel system to manage the Agents and other staff. Simple applications and databases were developed as necessary to simulate these applications.

In addition to showcasing the capabilities of WebSphere Business Integration Server Express Plus to provide synchronization, we wanted to show the products being used to integrate with other technologies, such as:

- Web services, which could be used, for example, by a sophisticated newspaper to receive advertising requests.
- E-mail, to communicate with the vendor and the buyer.

Note: Upon further examination, we discovered that our Web sample applications did not have the functionality to allow us to e-mail buyers and sellers. Rather than changing the sample applications, we changed our e-mail integration example so that an e-mail is sent to a sales manager instead.

- JMS.

Our goal in developing this sample WebSphere Business Integration solution coincided with a common business driver or constraint of not being able to modify any existing applications.

After considering problem creation and the consultant, system architect and designer perspectives, we had to don the project manager hat and prioritize the parts of our sample solution into releases. The development phases became:

1. Synchronization between the SAL301R application and the legacy Agent system for users and property/inventory.
2. Implementation a Web service for a newspaper.
3. E-mail to the sales manager.
4. JMS to the escrow agent; this was not developed.

13.2 Development begins

After prioritizing the phases of the project, the components that comprised the solution were evaluated and allocated to the developers in the team. The development team used CVS for all the development as the repository for the various artifacts built for the solution.

13.3 Preparing the development environment

This section will describe how to set up the development environment.

13.3.1 System Manager

Open the System Manager by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → System Manager. The System Manager will look similar to Figure 13-1 the first time it is opened.

![System Manager - WebSphere Studio Workbench SDK](image)

*Figure 13-1 System Manager*
For our sample application, we will need one new Integration Component Library (ICL) and one corresponding user project. The ICL we used was SAB404RRealtyICL and the user project was SAB404RRealtyICS.

Creating a new Integration Component Library

1. Create a new library by right-clicking the Integration Component Libraries folder and selecting New Integration Component Library. See Figure 13-2.

![Figure 13-2 Create Integration Component Library](image)

2. In the Project Name field on the next window, enter the name of the new Integration Component Library. In our case, we entered SAB404RRealtyICL.

Creating a new user project

1. Create a new user project by right-clicking the InterChange Server Projects folder and selecting New ICS project. See Figure 13-3 on page 473.

**Note:** The pre-installed WBIExpressLibrary contains the Business Objects and Collaboration templates that come with WebSphere Business Integration Server Express.
2. In the Project name field on the next page, enter the name of the project. In our case, this is SAB404RRealtyICS. Select the SAB404RRealtyICL library as shown in Figure 13-4 on page 474 to make it available to the user project.
Registering the Interchange Server instance
We have to start the InterChange server before we can register it as an instance in the System Manager.

1. Start the InterChange Server by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Interchange Server Express → Interchange Server Express.

Once the server is up and running, you should be able to see the command windows PersistantNameServer and Interchange Server Express. Look for the ready keyword in both windows, as shown in Figure 13-5 on page 475.
2. Register the InterChange Server with the System Manager by right-clicking the InterChange Server Instances folder and selecting Register server.

3. In the Register new server window, enter a hostname, username and password as shown in Figure 13-7 on page 476. In the password field, type null.
Once all the necessary projects are created and a server instance is registered, the System Manager should look similar to that shown in Figure 13-8.
13.3.2 Database and stored procedures

The Interchange Server uses stored procedures to manage relationships. The following page describes how we can set up our development environment to compile DB2 stored procedures.


**Note:** This step has to be completed before you can create any new relationships.

13.4 Adapter configuration

This section will cover the default configuration of the adapter instances that we need. The employee and property synchronization scenarios require the following adapter instances. Refer to Table 13-1.

Table 13-1 Adapter instances

<table>
<thead>
<tr>
<th>#</th>
<th>EIS</th>
<th>Instance name</th>
<th>Adapter type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SAL301R</td>
<td>SAL301RConnector</td>
<td>JDBC</td>
</tr>
<tr>
<td>2</td>
<td>SAB404R</td>
<td>SAB404RConnector</td>
<td>JDBC</td>
</tr>
<tr>
<td>3</td>
<td>E-mail</td>
<td>EMailConnector</td>
<td>E-mail</td>
</tr>
<tr>
<td>4</td>
<td>Property publishing service</td>
<td>PropertyPublishingConnector</td>
<td>Web service</td>
</tr>
</tbody>
</table>

13.4.1 Creating new adapter instances

Follow the steps below to create new adapter instances according to Table 13-1.

1. Right-click the **SAB404RRealtyICL/Connectors** folder and select **Create New Connector...**, as shown in Figure 13-9 on page 478.
2. In the New Connector window, enter the name of your adapter instance in the Name field and select an adapter template, as illustrated in Figure 13-10 on page 479.
3. Click the **OK** button in the New Connector window to open the Connector Configuration Express tool with the default values for the adapter you have chosen, as displayed in Figure 13-11 on page 480.
4. Set standard property values used by all the adapter instances.

The list of standard properties in this section will show the same values for all the adapter instances in our sample project.

- **AgentTraceLevel**
  
  Level of trace messages for the Adapter Agent. The default value is 0. Change the value to 5.

- **BrokerType**
  
  Identifies the integration broker we are using, which is ICS.
- **CharacterEncoding**
  Specifies the character code set used to map from a character to a numeric value. Java-based adapters do not use this property.

- **ConCurrentEventTriggeredFlows**
  Determines how many Business Objects can be concurrently processed by the connector for event delivery. Use the default value.

- **ControllerStoreAndForwardMode**
  If this property is set to `true` and the destination Adapter Agent is unavailable when an event reaches ICS, the connector controller blocks the request. When the Adapter Agent becomes operational, the controller forwards the request to it. Use the default value.

- **ControllerTraceLevel**
  Level of trace messages for the Connector Controller. The default value is 0. Change the value to 5.

- **EnableOidForFlowMonitoring**
  If set to `true`, the adapter framework will mark the incoming ObjectEventId as a foreign key for the purpose of flow monitoring. Use the default value.

- **Jvm properties**
  Use the default values for all the Jvm-specific properties.

- **Locale**
  Specifies the language code, country and, optionally, the associated character code set. Use the default value (en_US).

- **LogAtInterChangeEnd**
  Turns on e-mail notification when set to `true`. Requires that an e-mail account be specified. Use the default value.

- **MaxEventCapacity**
  The maximum number of events in the controller buffer. Use the default value.

- **OAD properties**
  Use the default values for all the OAD properties.

- **PollEndTime**
  Time to stop polling the event queue. Use the default value.

- **PollFrequency**
  Amount of time between polling actions, in milliseconds. Use the default value.
- **PollStartTime**
  Time to start polling the event queue. Use the default value.

- **RepositoryDirectory**
  Location of the repository from which the adapter reads the Business Object definitions. Use the default value (remote).

- **RestartRetryCount**
  Specifies the number of times the connector attempts to restart itself. Use the default value.

- **RestartRetryInterval**
  Specifies the interval in minutes. Use the 1 minute default.

- **WireFormat**
  Message format on the transport. Use the default value.

- **DeliveryTransport**
  We decided to use JMS as a transport layer between the Adapter Agent and the Connector Controller. WebSphere MQ queues need to be created to support this decision. This chapter will provide the necessary queue names for each adapter. For instructions on how to implement the queues and other required MQ objects, please refer to “WebSphere MQ objects and settings” on page 628. The DeliveryTransport property specifies the transport mechanism for the delivery of events. Change the value from IDL to JMS.

**JMS-specific properties:**

- **ContainerManagedEvents**
  This property allows a JMS-enabled connector with a JMS event store to provide guaranteed event delivery. We will not be using this functionality in our sample. Leave the value field empty.

- **DuplicateEventElimination**
  When you set this property to true, a JMS-enabled connector can ensure that duplicate events are not delivered to the delivery queue. Use the default value.

- **jms.FactoryClassName**
  Specifies the class name to instantiate for a JMS provider. Use the default value.

- **jms.MessageBrokerName**
  Specifies the broker name to use for the JMS provider. Change the value to WebSphereICS.queue.manager.
– jms.NumConcurrentRequests.
  Specifies the maximum number of concurrent service call requests that can be sent to a connector at the same time. Use the default value.
– jms.Password
  Specifies the password for the JMS provider. Use the default value.
– jms.Username
  Specifies the user name for the JMS provider. Use the default value.

5. Set Standard Properties that will vary from instance to instance.
Look at the section describing the adapter you are working on (SAL301R, SAB404R, E-mail, PropertyPublisher) to find the instance-specific values.
– ApplicationName
  Name that uniquely identifies the adapter’s application.
– MessageFileName
  Name of the connector-specific message file.

**JMS-specific properties:**
– AdminInQueue
  Queue used by the integration broker to send administrative messages to the connector.
– AdminOutQueue
  Queue used by the connector to send administrative messages to the integration broker.
– DeliveryQueue
  Queue used by the connector to send Business Objects to the integration broker.
– FaultQueue
  If an error occurs while a message is being processed, the connector moves the message to this queue; included are a status indicator and a description of the problem.
– RequestQueue
  Queue used by the integration broker to send Business Objects to the connector.
– ResponseQueue
  Queue that delivers a response message from the connector framework to the integration broker.
- **SynchronousRequestQueue**
  Delivers request messages that require a synchronous response from the connector framework to the broker.

- **SynchronousResponseQueue**
  Delivers response messages sent in reply to a synchronous request from the broker to the connector framework.

6. Set the adapter-specific properties.

   Switch to the Connector-Specific Properties page in the Connector Configurator tool and set adapter-specific properties. Look at the section describing the adapter you are working on (SAL301R, SAB404R, E-mail, PropertyPublisher) to find the values that have to be set.

   ![Connector Configurator Express](image)

   **Figure 13-12 Adapter-specific properties**

7. Configure the logging/tracing output.

   Switch to the Trace/Log Files page and specify the name and path of the log/trace files to be used for this adapter instance.
8. Prepare the Adapter Agent runtime environment.

For more details, refer to the section that describes the adapter you are working on.

9. Test the configuration.

Test the adapter configuration by following the steps below:

a. Start InterChange Server by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Interchange Server Express → Interchange Server Express.

b. Deploy the adapter:

Connect to the server from the System Manager. Once you have connected, deploy the adapter by dragging the <adapterInstance>::SAB404RRealtyICL shortcut to the InterChange Server.

c. Restart the server. Stop the server by using the Shut down → Gracefully option; see Figure 13-14 on page 486.
d. Start the server again.

e. When the server is up and running again, start the Adapter Agent by double-clicking the shortcut you have created. If you can find the `appPolling` keyword in the Adapter Agent command window and there are no error messages, we can assume that the Adapter Agent is working properly. We will describe how to test the adapter later in this chapter.

13.4.2 SAL301RConnector

This section contains instance- and adapter-specific information for the JDBC adapter we will be using to connect to SAL301R.

**Instance-specific properties**

Open the Connector Configurator by double-clicking the **SAL301RConnector** instance and switch to the Standard Properties page. Where necessary, update the property values as specified below.
Table 13-2 Standard properties for the SAL301RConnector

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationName</td>
<td>SAL301RConnector</td>
</tr>
<tr>
<td>MessageFileName</td>
<td>SAL301RConnector.txt</td>
</tr>
<tr>
<td>AdminInQueue</td>
<td>SAL301RCONNECTOR/ADMININQUEUE</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>SAL301RCONNECTOR/ADMINOUTQUEUE</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>SAL301RCONNECTOR/DELIVERYQUEUE</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>SAL301RCONNECTOR/FAULTQUEUE</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>SAL301RCONNECTOR/REQUESTQUEUE</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>SAL301RCONNECTOR/RESPONSEQUEUE</td>
</tr>
<tr>
<td>SynchronousRequestQueue</td>
<td>SAL301RCONNECTOR/SYNCHRONOUSREQUESTQUEUE</td>
</tr>
<tr>
<td>SynchronousResponseQueue</td>
<td>SAL301RCONNECTOR/SYNCHRONOUSRESPONSEQUEUE</td>
</tr>
</tbody>
</table>

Connector-specific properties
Switch to the Connector-specific Properties page and, where necessary, update the property values as specified in Table 13-16 on page 548.
Table 13-3 provides a short description of the connector-specific properties. For a more detailed description, look at the JDBC adapter documentation in the WebSphere Business Integration Server Express infocenter.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationPassword</td>
<td>Password for the DB account</td>
</tr>
<tr>
<td>ArchiveProcessed</td>
<td>Set this property to true to cause events to be inserted into the archive table after they are deleted from the event table.</td>
</tr>
<tr>
<td>ArchiveTableName</td>
<td>Name of archive queue table. Events delete from the event table will be inserted into this table.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutoCommit</td>
<td>When set to <code>true</code>, all transactions are automatically committed.</td>
</tr>
<tr>
<td>CheckForEventTableInInit</td>
<td>Tells the adapter to check for the existence of the event and archive tables during initialization.</td>
</tr>
<tr>
<td>ChildUpdatePhyDelete</td>
<td>Set this property to <code>true</code> to cause the connector to physically delete the data record from the database.</td>
</tr>
<tr>
<td>CloseDBConnection</td>
<td>When set to <code>true</code> the database connection is closed after every service call.</td>
</tr>
<tr>
<td>ConnectorID</td>
<td>A unique ID for the connector. This ID is useful to retrieve events for a particular instance of the connector.</td>
</tr>
<tr>
<td>DatabaseURL</td>
<td>Identifies the database server to which the connector should connect.</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Specifies the date format that the connector expects to receive and return.</td>
</tr>
<tr>
<td>MaximumDatabaseConnections</td>
<td>Number of simultaneous database connection allowed.</td>
</tr>
<tr>
<td>PingQuery</td>
<td>Query used by the adapter to check database connectivity.</td>
</tr>
<tr>
<td>PreserveUIDSeq</td>
<td>Set to <code>true</code>, the unique ID is not committed until the Business Object is successfully processed in the destination application.</td>
</tr>
<tr>
<td>RDBMSVendor</td>
<td>Specifies which RDBMS the connector uses for special processing.</td>
</tr>
<tr>
<td>RetryCountAndInterval</td>
<td>Specifies the number of attempts and the interval in seconds that the connector should use when it is unable to lock data while performing an update operation.</td>
</tr>
<tr>
<td>UniqueIDTableName</td>
<td>Specifies the table that contains the latest value used for generation of a unique ID.</td>
</tr>
<tr>
<td>UseDefaults</td>
<td>Checks whether a valid value is provided for required Business Object attributes.</td>
</tr>
</tbody>
</table>
SAL301R Adapter Agent runtime environment

Each Adapter Agent instance runs as a standalone programs so we have to create a runtime environment for it. Follow the steps below to set up the runtime environment for the SAL301R Adapter Agent.

1. Create the runtime folder.
   Copy <ProductDir>\WebSphereServer\connectors\JDBC and rename the copy to <ProductDir>\WebSphereServer\connectors\SAL301R.

2. Change the name of the CW<adapter>.jar file.
   Rename the <ProductDir>\WebSphereServer\connectors\SAL301\CWJDBC.jar file to <ProductDir>\WebSphereServer\connectors\SAL301R\CWSAL301R.jar.

3. Edit the paths in the start_JDBC.bat file.
   Add the two lines below just above the `SET DATADIRECT` statement.

   **Example 13-1 Add DB2-specific information to the start_JDBC.bat file**

   ```
   SET JDBCDRIVERPATH=<your path>\db2java.zip
   SET DBEXECUTABLEPATH=<your path>\sqllib\bin
   ```

   Add the DBEXECUTABLE variable to the `java.library.path` statement at the end of the file.

Property | Description
--- | ---
UseDefaultsForCreatingChildBOs | Checks whether a valid value is provided for required Business Object attributes.
UseDefaultsForRetrieve | Sets default values in the BO before it is retrieved from the database.
EventKeyDel | Specifies the delimiter when the `object_key` column of the event table contains multiple attribute values.
EventOrderBy | Specifies whether to turn off the ordering of events.
EventQueryType | Specifies what type of query to use when retrieving events.
EventTableName | Name of event queue table, which is used by the connectors polling mechanism.
JDBCDriverClass | JDBC driver.
ApplicationUserName | Database account/username.
Example 13-2  Update java.library.path

From:
%CWJAVA% -mx128m -ms64m -Djava.ext.dirs="%MQ_LIB%;%JRE_EXT_DIRS%;
-Djava.library.path="%CROSSWORLDS%;%CONNDIR%;"%MQ_LIB%;%JRE_EXT_DIRS% 
%ORB_PROPERTY% -Duser.home="%CROSSWORLDS%" -cp %CLASSES%;%CONNDIR%;%CONNJAR%; AppEndWrapper -l%CONNPACKAGE% -n%CONNAME%Connector -s%SERVER% %3 %4 %5

To:
-Djava.library.path="%CROSSWORLDS%;%CONNDIR%;"%MQ_LIB%;%JRE_EXT_DIRS%;
-DBEXECUTABLEPATH% %ORB_PROPERTY% -Duser.home="%CROSSWORLDS%" -cp %CLASSES%;%CONNDIR%;%CONNJAR%; AppEndWrapper -l%CONNPACKAGE% -n%CONNAME%Connector -s%SERVER% %3 %4 %5

4. Create a Windows shortcut to start the SAL301R Adapter Agent.

   In the Target field of the shortcut, enter the following line.

Example 13-3  SAL301R shortcut

<ProductDir>\WebSphereServer\connectors\SAL301R\start_JDBC.bat SAL301R
WebSphereICS -cconnectors\SAL301R\SAL301RConnector.cfg

   Notice the three parameters SAL301R, WebSphereICS and the .cfg file. The
   first parameter specifies the adapter instance we want to start. The second
   parameter specifies the name of the interchange server. The last parameter
   contains the adapter configuration stored to file.

   From the Connector Configurator tool, click File → Save As → To File and
   store the file in the <ProductDir>/WebSphere/connectors/SAL301R folder.

Event notification mechanism
The SAL301R application requires Create, Update and Delete triggers on the
PROP database table. 7.3.2, “Setting up event notification” on page 175
describes how to set up the event notification for SAL301R.

13.4.3 SAB404RConnector

This section contains instance- and adapter-specific information for the JDBC
adapter we will be using to connect to SAB404R.

Instance-specific properties
Open the Connector Configurator by double-clicking the SAB404RConnector
instance and switch to the Standard Properties page. Where necessary, update
the property values as specified below.
**Table 13-4  Standard properties for the SAB404RConnector**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationName</td>
<td>SAB404RConnector</td>
</tr>
<tr>
<td>MessageFileName</td>
<td>SAB404RConnector.txt</td>
</tr>
<tr>
<td>AdminInQueue</td>
<td>SAB404RCONNECTOR/ADMININQUEUE</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>SAB404RCONNECTOR/ADMINOUTQUEUE</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>SAB404RCONNECTOR/DELIVERYQUEUE</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>SAB404RCONNECTOR/FAULTQUEUE</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>SAB404RCONNECTOR/REQUESTQUEUE</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>SAB404RCONNECTOR/RESPONSEQUEUE</td>
</tr>
<tr>
<td>SynchronousRequestQueue</td>
<td>SAB404RCONNECTOR/SYNCHRONOUSREQUESTQUEUE</td>
</tr>
<tr>
<td>SynchronousResponseQueue</td>
<td>SAB404RCONNECTOR/SYNCHRONOUSRESPONSEQUEUE</td>
</tr>
</tbody>
</table>

**Connector-specific properties**

Switch to the Connector-Specific Properties page and, where necessary, update the property values as specified in Figure 13-5 on page 494. Table 13-3 on page 488 provides a short description of the JDBC connector-specific properties. For a more detailed description, look at the JDBC adapter documentation in the WebSphere Business Integration Server Express infocenter.
Figure 13-17 Connector-specific properties for SAB404RConnector

### SAB404R Adapter Agent runtime environment

Follow the description in “SAL301R Adapter Agent runtime environment” on page 490 using SAB404R instead of SAL301R.

### Event notification mechanism

The SAB404R application requires Create, update and delete triggers on the EMPLOYEE database table. 7.3.2, “Setting up event notification” on page 175 describes how to set up the event notification for SAL301R. The SAB404R-Events.ddl contains the required SAB404R database triggers.
13.4.4 PropertyPublishingConnector

This section contains instance- and adapter-specific information for the WebServices adapter we will be using to connect to news service.

Instance-specific properties

Open the Connector Configurator by double-clicking the PropertyPublishingConnector instance and switch to the Standard Properties page. Where necessary, update the property values as specified below.

Table 13-5 Standard properties for the PropertyPublishingConnector

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationName</td>
<td>PropertyPublishingConnector</td>
</tr>
<tr>
<td>MessageFileName</td>
<td>PropertyPublishingConnector.txt</td>
</tr>
<tr>
<td>AdminInQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/ADMININQUEUE</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/ADMINOUTQUEUE</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/DELIVERYQUEUE</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/FAULTQUEUE</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/REQUESTQUEUE</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/RESPONSEQUEUE</td>
</tr>
<tr>
<td>SynchronousRequestQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/SYNCHREQUESTQ</td>
</tr>
<tr>
<td>SynchronousResponseQueue</td>
<td>PROPERTYPUBLISHINGCONNECTOR/SYNCHRESPONSEQ</td>
</tr>
</tbody>
</table>

Connector-Specific properties

Switch to the Connector-Specific Properties page and where necessary, update the property values as specified in Figure 13-18 on page 495. Table 13-6 on page 495 provides a short description of the WebService connector-specific properties. For a more detailed description, look at the WebService adapter documentation in the WebSphere Business Integration Server Express infocenter.
**Note:** For the PropertyPublishing adapter configuration, you can leave the JNDI element above as is.

### Table 13-6  Short description of the WebService Adapter specific properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConnectorType</strong></td>
<td>If this property is set to WebService, when binding the collaboration port, System Manager displays the connector as a Web Services connector.</td>
</tr>
<tr>
<td><strong>ProtocolHandlerFramework</strong></td>
<td>The Protocol Handler Framework uses this property to load and configure its protocol handlers.</td>
</tr>
<tr>
<td><strong>DataHandlerMetaObjectName</strong></td>
<td>This is the name of the meta-object that the data handler uses to set configuration properties.</td>
</tr>
</tbody>
</table>
PropertyPublishing Adapter Agent runtime environment

Follow the steps below to set up the runtime environment for the SAL301R Adapter Agent.

1. Create the runtime folder.
   Copy <ProductDir>\WebSphereServer\connectors\WebServices and rename the copy to <ProductDir>\WebSphereServer\connectors\PropertyPublishing.

2. Change the name of the CW<adapter>.jar file.
   Example 13-4   Rename the CWWebServices.jar file
   From:
   <ProductDir>\WebSphereServer\connectors\PropertyPublishing\CWWebServices.jar
   To:
   <ProductDir>\WebSphereServer\connectors\PropertyPublishing\CWPropertyPublishing.jar

   Find the SET AGENT=.. statement and change it as shown in Example 13-5.
   Example 13-5   Modify the start_WebServices.bat file
   From:
   set AGENT="%CONNDIR%\CWWebServices.jar"
   To:
   set AGENT="%CONNDIR%\CWPropertyPublishing.jar"

4. Create a Windows shortcut to start the PropertyPublishing Adapter Agent.
   In the Target field of the shortcut, enter the following line.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNDI</td>
<td>The connector maintains one set of JNDI provider properties that are used by the SOAP/JMS protocol handler.</td>
</tr>
<tr>
<td>ProtocolListenerFramework</td>
<td>The protocol listener framework uses this property to load protocol listeners.</td>
</tr>
<tr>
<td>UseDefaults</td>
<td>Tells the adapter to use the default values in the BO if they exist.</td>
</tr>
</tbody>
</table>

Property Description
Example 13-6  PropertyPublishing shortcut

C:\IBM\WebSphereServer\connectors\PropertyPublishing\start_WebServices.bat
PropertyPublishing WebSphereICS
-cconnectors/PropertyPublishing/PropertyPublishingConnector.cfg

Notice the three parameters PropertyPublishing, WebSphereICS and the .cfg file. The first parameter specifies the adapter instance we want to start. The second parameter specifies the name of the interchange server. The last parameter contains the adapter configuration stored to file.

From the Connector Configurator tool, click File → Save As → To File and store the file in the <ProductDir>/WebSphere/connectors/PropertyPublishing folder.

13.4.5 EMailConnector

This section contains instance- and adapter-specific information for the WebServices adapter we will be using to connect to the news service.

Instance-specific properties
Open the Connector Configurator by double-clicking the EMailConnector instance and switch to the Standard Properties page. Where necessary, update the property values as specified below.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationName</td>
<td>EMailConnector</td>
</tr>
<tr>
<td>MessageFileName</td>
<td>EMailConnector.txt</td>
</tr>
<tr>
<td>AdminInQueue</td>
<td>EMAILCONNECTOR/ADMININQUEUE</td>
</tr>
<tr>
<td>AdminOutQueue</td>
<td>EMAILCONNECTOR/ADMINOUTQUEUE</td>
</tr>
<tr>
<td>DeliveryQueue</td>
<td>EMAILCONNECTOR/DELIVERYQUEUE</td>
</tr>
<tr>
<td>FaultQueue</td>
<td>EMAILCONNECTOR/FAULTQUEUE</td>
</tr>
<tr>
<td>RequestQueue</td>
<td>EMAILCONNECTOR/REQUESTQUEUE</td>
</tr>
<tr>
<td>ResponseQueue</td>
<td>EMAILCONNECTOR/RESPONSEQUEUE</td>
</tr>
<tr>
<td>SynchronousRequestQueue</td>
<td>EMAILCONNECTOR/SYNCHREQUESTQ</td>
</tr>
<tr>
<td>SynchronousResponseQueue</td>
<td>EMAILCONNECTOR/SYNCHRESPONSEQ</td>
</tr>
</tbody>
</table>
Connector-specific properties

Switch to the Connector-Specific Properties page and where necessary, update the property values as specified in Figure 13-19. Table 13-5 on page 494 provides a short description of the EMailConnector-specific properties. For a more detailed description, look at the e-mail adapter documentation in the WebSphere Business Integration Server Express Infocenter.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Encrypt</th>
<th>Update Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SMTP_MailHost</td>
<td>&lt;your host&gt;</td>
<td>agent restart</td>
<td></td>
</tr>
<tr>
<td>2 DebugMode</td>
<td>false</td>
<td>agent restart</td>
<td></td>
</tr>
<tr>
<td>3 ApplicationUserName</td>
<td>&lt;your username&gt;</td>
<td>agent restart</td>
<td></td>
</tr>
<tr>
<td>4 PollConfigMO</td>
<td>MO_EMail_Default</td>
<td>agent restart</td>
<td></td>
</tr>
<tr>
<td>5 DataHandlerConfigMO</td>
<td>MO_DataHandler_Default</td>
<td>agent restart</td>
<td></td>
</tr>
<tr>
<td>6 MailsPerMailBox</td>
<td>1</td>
<td>agent restart</td>
<td></td>
</tr>
<tr>
<td>7 ApplicationPassword</td>
<td>&lt;your password&gt;</td>
<td>agent restart</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 13-19 Connector-specific properties for the EMailConnector**

**Table 13-8 Short description of the EMail Adapter-specific properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP_MailHost</td>
<td>The mail host to be contacted to send e-mail.</td>
</tr>
<tr>
<td>DebugMode</td>
<td>Specifies whether JavaMail debug messages are sent to STDOUT.</td>
</tr>
<tr>
<td>ApplicationUserName</td>
<td>Application user name used to connect to the mail host.</td>
</tr>
<tr>
<td>DataHandlerConfigMO</td>
<td>Name of the meta-object that the Data Handler uses to set configuration properties.</td>
</tr>
<tr>
<td>MailsPerMailBox</td>
<td>Specifies the number of messages processed in each mailbox before the adapter proceeds to process the next mailbox.</td>
</tr>
<tr>
<td>ApplicationPassword</td>
<td>Application password used to connect to the mail host.</td>
</tr>
</tbody>
</table>

**E-mail Adapter Agent runtime environment**

Follow the steps below to set up the runtime environment for the E-mail Adapter Agent.
1. Create a Windows shortcut to start the E-mail Adapter Agent.

   In the Target field of the shortcut, enter the following line.

   **Example 13-7  EMail shortcut**
   
   ```
   C:\IBM\WebSphereServer\connectors\EMail\start_Email.bat EMail WebSphereICS -cEMailConnector.cfg
   ```
   
   Notice the three parameters E-mail, WebSphereICS and the .cfg file. The first parameter specifies the adapter instance we want to start. The second parameter specifies the name of the interchange server. The last parameter contains the adapter configuration stored to file.

2. From the Connector Configurator tool click **File → Save As → To File** and store the file in the `<ProductDir>/WebSphere/connectors/EMail` folder.

### 13.5 SAB404R to SAL301R: employee synchronization

This task was allocated to two of the developers, making use of other elements developed by others in the team. These developers split the tasks in the following manner:

- Developer 1 developed the Business Objects and the collaboration
- Developer 2 developed the maps and relationship.

The team reused the adapters created in 13.4.1, “Creating new adapter instances” on page 477 and added support for the employee-related Business Objects.

#### 13.5.1 Business Objects

In this section, we will describe the development of the Business Objects specified in 8.3.1, “Overview of Business Object Designer Express” on page 223:

- Generic Business Objects for Employee synchronization
- Application Specific Business Objects for SAB404R
- Application Specific Business Objects for SAL301R

Table 13-9 on page 500 lists the Business Objects we will create in this chapter and how these will be created.
<table>
<thead>
<tr>
<th>Business Object</th>
<th>Type</th>
<th>Create using</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Generic</td>
<td>Re-use existing</td>
<td>This Business Object is part of the base installation</td>
</tr>
<tr>
<td>EmployeeAddress</td>
<td>Generic, child of Employee</td>
<td>Re-use existing</td>
<td>This Business Object is part of the base installation</td>
</tr>
<tr>
<td>EmployeeHR_Info</td>
<td>Generic, child of Employee</td>
<td>Re-use existing</td>
<td>This Business Object is part of the base installation</td>
</tr>
<tr>
<td>EmployeeHR_Misc</td>
<td>Generic, child of Employee</td>
<td>Re-use existing</td>
<td>This Business Object is part of the base installation</td>
</tr>
<tr>
<td>EmployeeJob</td>
<td>Generic, child of Employee</td>
<td>Re-use existing</td>
<td>This Business Object is part of the base installation</td>
</tr>
<tr>
<td>SAB404R_EMPLOYEE</td>
<td>Application-specific</td>
<td>JDBC ODA</td>
<td>Represents the EMPLOYEE table</td>
</tr>
<tr>
<td>SAB404R_OFFICE</td>
<td>Application-specific, child of SAB404R_EMPLOYEE</td>
<td>JDBC ODA</td>
<td>Represents the OFFICE table</td>
</tr>
<tr>
<td>SAB404R_ADDRESS</td>
<td>Application-specific, child of SAB404R_EMPLOYEE</td>
<td>JDBC ODA</td>
<td>Represents the ADDRESS table. This can also be a child of SAB404R_EMPLOYEE</td>
</tr>
<tr>
<td>SAL301R_USR</td>
<td>Application-specific</td>
<td>JDBC ODA</td>
<td>Represents the USR table</td>
</tr>
<tr>
<td>SAL301R_USR_ADDR</td>
<td>Application-specific, child of SAL301R_USR</td>
<td>JDBC ODA</td>
<td>Represents the USR_ADDR table</td>
</tr>
</tbody>
</table>

**Generic Business Objects**

Instead of creating the Generic Business Objects from scratch, we will use the Employee and its related child objects from the base installation of WebSphere Business Integration Server Express.
Follow these steps to enable the Generic Business Objects in SAB404RRealtyICL:

1. In the System Manager, select the following Business Objects in the WBIExpressLibrary project (use the Ctrl key to select multiple objects):
   - Employee
   - EmployeeAddress
   - EmployeeHR_Info
   - EmployeeHR_Misc
   - EmployeeJob

2. Select Edit → Copy from the System Manager menu.
3. Select the Business Objects folder in the SAB404RRealtyICL project.
4. Select Edit → Paste to paste the Business Objects into the project.
   You can also copy objects using Ctrl+C and Ctrl+V, or by dragging the objects to the destination project folder.
5. Double-click the Employee Business Object to open it in Business Object Designer Express.
6. Remove all default values in the Employee Business Object and all its children. Figure 13-20 on page 502 shows the Employee Business Objects with its child Business Objects.
7. Select File → Save to save the Business Object.

Note: If the Employee Generic Business Object does not contain all the child Business Objects listed in Table 13-9, you must import the BIA_BO_BaseCollabBOs.jar repository file.
### Figure 13-20  Attributes for Employee Generic Business Object

<table>
<thead>
<tr>
<th>Pos</th>
<th>Name</th>
<th>Type</th>
<th>key</th>
<th>Foreign Key</th>
<th>Required</th>
<th>Cardinality</th>
<th>Max</th>
<th>Def</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EmployeeId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>EmployeeNumber</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>EmpEffectiveCnDate</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>EmpEffectiveTillDate</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Prefix</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LastName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>FirstName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>MiddleName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Suffix</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SocialNumber</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>WorkPhone</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>WorkPhoneExt</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>HomePhone</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Fax1</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Fax2</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Pager</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Cellular</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Telex</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Telephone1</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Telephone2</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>CompanyId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>EmployeeHR_Info</td>
<td>EmployeeHR_Info</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>OriginatingSystemId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>EmployeeAddress</td>
<td>EmployeeAddress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>EmailAddress</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>EmployeeJob</td>
<td>EmployeeJob</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>LoginId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>ExemplarB, Miss</td>
<td>ExemplarB, Miss</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SAB404R Application Specific Business Objects**

Application Specific Business Objects can be created using an Object Discovery Agent (ODA) if this is provided with the specific adapter. WebSphere Business Integration Server Express supplies five ODAs for different technology adapters: COM, JDBC, Portal Infranet, Web services and XML.

SAB404R is a relational database in DB2, and we can therefore use the JDBC ODA to create the Business Objects based on the EMPLOYEE, OFFICE, and ADDRESS tables.
Before you can use an ODA to create the Business Objects, you must start the agent by selecting Start → Programs → IBM WebSphere Business Integration Server Express → Adapters → Object Discovery Agents → JDBC.

Creating Business Object definitions using JDBC ODA

Open Business Object Designer Express and select File → New Using ODA from the menu to open the Business Object Wizard.

1. Select the Agent:
   a. Click Find Agents.
   b. Select the JDBCODA agent from the list of located agents.
   c. Click Next.

2. Configure the Agent:
   a. Fill in the following details:
      
      Current Profile: SAB404R
      UserName: <db_username>
      Password: <db_password>
      DatabaseUrl: jdbc:db2:SAB404R
      DatabaseDriver: COM.ibm.db2.jdbc.app.DB2Driver
      DefaultBOPrefix: SAB404R_
   b. Click Save to save the profile. This enables you to retrieve the information you typed in if you wish to use it later.
   c. Click Next.

3. Select the Source:
   a. Expand the DB2ADMIN schema and tables.
   b. Select the ADDRESS, EMPLOYEE and OFFICE tables.
   c. Click Next.

4. Confirm source nodes for Business Objects. Click Next.

5. Generate Business Object definitions:
   a. In the list of verbs, deselect the RetrieveByContent option.
   b. Select No to Add stored procedures.
   c. Click OK.

6. Save Business Object definitions:
   a. Choose to save the Business Objects to the SAB404RRealtyICL project.
   b. Elect to open the Business Objects.
   c. Do not select to shut down the ODA agent, since we will use this later.
   d. Click Finish.
The JDBC ODA agent has created three Business Objects based on the EMPLOYEE, OFFICE and ADDRESS tables. It has added the following application-specific information:

- On the Business Object level: TN=<table name>; SCN=
- On the attributes: CN=<column name>

**Modifying application-specific information**

Modify the Business Objects as follows:

1. For each Business Object, remove the SCN= attribute. The application-specific information on the Business Object level should be TN=<table name>. If you are using the soft deletion attribute on the JDBC adapter configuration, the SCN attribute specifies the column to update if the object/row is deleted.

2. Add SAB404R_OFFICE as a child attribute to SAB404R_EMPLOYEE:
   a. Right-click attribute EMPOFFICEKEY in SAB404R_EMPLOYEE and select Insert Below.
   b. Specify the following attributes:
      - Name: OFFICE
      - Type: SAB404R_OFFICE
      - Cardinality: 1

   An employee can only work in one office.

3. Add SAB404R_ADDRESS as a child attribute to SAB404R_OFFICE:
   a. Right-click attribute OFFICEADDRKEY in SAB404R_OFFICE and select Insert Below.
   b. Specify the following attributes:
      - Name: ADDRESS
      - Type: SAB404R_ADDRESS
      - Cardinality: 1

   An office can only be located at one address.

4. Append the following to the application-specific properties for attributes in SAB404R_EMPLOYEE:
   - EMPKEY: UID=AUTO
     UID=AUTO specifies that the key is auto-generated
   - EMPOFFICEKEY:
     FK=SAB404R_OFFICE.OFFICEKEY; CONTAINMENT=NO_OWNERSHIP
     FK=SAB404R_OFFICE.OFFICEKEY specifies that the parent/child relationship between employee and office is stored in the parent Business Object.
• CONTAINMENT=NO_OWNERSHIP specifies that the office is not owned by one employee, but shared among several.

5. Append the following to the application-specific properties for attributes in SAB404R_OFFICE:
   – OFFICEKEY: UID=AUTO
   – OFFICEADDRKEY:
     FK=SAB404R_ADDRESS.ADDRKEY:CONTAINMENT=OWNERSHIP
     CONTAINMENT=OWNERSHIP specifies that the address is owned by the office.

6. Append the following to the application-specific properties for attributes in SAB404R_ADDRESS:
   ADDRKEY: UID=AUTO

7. Set the Foreign Key attribute property to the following attributes:
   – SAB404R_EMPLOYEE.EMPOFFICEKEY
   – SAB404R_OFFICE.OFFICEADDRKEY

8. Save and close the Business Object definitions.

Figure 13-21 on page 506 shows the attributes for SAB404R_EMPLOYEE.
As with SAB404R, SAL301R is a relational database in DB2. We will use the JDBC ODA to create the Business Objects based on the USR and USR_ADDR tables.

**Create Business Object definitions using JDBC ODA**

Open Business Object Designer Express and select File → New Using ODA from the menu to open the Business Object Wizard.

1. Select the Agent:

   If JDBCODA is not in the list of located agents, perform these steps:

   a. Click **Find Agents**.

   b. Select the JDBCODA agent from the list of located agents.

   c. Click **Next**.
2. Configure the Agent:
   a. Fill in the following details:
      - Current Profile: SAL301R
      - UserName: <db_username>
      - Password: <db_password>
      - DatabaseUrl: jdbc:db2:SAL301R
      - DatabaseDriver: COM.ibm.db2.jdbc.app.DB2Driver
      - DefaultBOPrefix: SAL301R_
   b. Click Save to save the profile.
   c. Click Next.

3. Select the Source:
   a. Expand the DB2ADMIN schema and tables.
   b. Select the USR and USR_ADDR tables.
   c. Click Next.

4. Confirm source nodes for Business Objects. Click Next.

5. Generating Business Object definitions
   a. In the list of verbs, de-select the RetrieveByContent option.
   a. Select No for Add stored procedures.
   a. Click OK.

6. Save Business Object definitions:
   a. Save the Business Objects to the SAB404RRealtyICL project.
   b. Open the Business Objects.
   c. Shut down the ODA agent.
   d. Click Finish.

The JDBC ODA agent has created two Business Objects based on the USR and USR_ADDR tables.

**Modify application-specific information**
Modify the Business Objects as follows:

1. For each Business Object, remove the SCN= attribute. The application-specific information on the Business Object level should be TN=<table name>.

2. Add SAL301R_USR_ADDR as a child attribute to SAL301R_USR:
   a. Right-click attribute USRCRDON in SAL301R_USR and select Insert Below.
b. Specify the following attributes:
   - Name: ADDRESS
   - Type: SAL301R_USR_ADDR
   - Cardinality: 1

   An employee will only have one address, the office address from SAB404R. In “SAL301R: user with multiple addresses” on page 547 we describe how the Business Object structure would be if multiple addresses were supported.

3. Append the following to the application-specific properties for attributes in SAB301R_USR:
   USRID: UID=AUTO

4. Append the following to the application-specific properties for attributes in SAL301R_USR_ADDR:
   - USRADDRID: UID=AUTO
   - USRID: FK=SAL301R_USR.USRID

   In SAL301R, the employee/address relationship is stored in the child Business Object.

5. Set the Foreign Key attribute property for SAL301R_USR_ADDR.USRID.

6. Save and close the Business Object definitions.

Figure 13-22 on page 509 shows the attributes for SAL301R_USR.
Adding support for Business Object in connectors

In order to use the Business Objects in a collaboration, you need to add support for the Business Objects in a connector. In this example we have two connectors, one for the SAL301R application, and one for SAB404R. These were created in “Adapter configuration” on page 477.

Modify SAB404RConnector

Perform the following steps to add support for our Business Objects in the SAB404RConnector:

1. Double-click the SAB404RConnector in the SAB404RRealtyICL project to open it in Connector Configurator Express.
2. Select the Supported Business Object tab.
3. In line 3, select the Employee Generic Business Object.
4. In line 4, select the SAB404R_EMPLOYEE Application Specific Business Object. Select agent support for this Business Object.

5. Select File → Save → To Project.


7. Select File → Exit.

Modify SAL301RConnector
Repeat the process above to add support for Employee and SAL301R_USR to the SAL301RConnector. Save the configuration to file by replacing the existing SAL301RConnector.cfg file.

Testing the SAL301R Business Objects
In this chapter we will describe a way to test if the Business Object structure and meta-data works as expected. In a Business Object test we want to send the SAL301R Business Object directly to the source, without any maps involved. The process requires us to:

- Add agent support for the Business Object to the SourceConnector.
- Create a new collaboration template
- Create a new collaboration object
- Deploy the test objects
- Perform the test
- Delete the deployed test objects

Add Business Objects to SourceConnector
Follow these steps to add agent support for the SAL301R Business Objects to the SourceConnector:

1. Copy the SourceConnector from the WBIExpressLibrary project to the SAB404RRealtyICL project.

2. Double-click the connector to open it in Connector Configurator Express.

3. Select the Supported Business Objects tab.

4. Delete the line containing the Customer Business Object by selecting it and press Delete.

5. Select SAL301R_USR from the list of objects, and select agent support.
   We will use the SourceConnector to send a SAL301R_USR Business Object to the SAL301R adapter.

6. Select File → Save → To Project.

7. Select File → Close.
Create the BOTest collaboration template

The collaboration template we want to use should only pass the Business Object to the destination application. Since we want to test the Create, Update, and Delete verbs we must use the Collaboration Foundation template in WBIExpressLibraryICL as a base for our template.

Perform these steps create the collaboration template:

1. Copy the Collaboration Foundation template from WBIExpressLibrary to SAB404RRealtyICL.
2. Double-click the template to open it in Process Designer Express.
3. Double-click the Definitions icon in the left pane. You will receive warning messages stating that the Controller Business Object does not exist, and a question if you want to delete the ports. Click No, since we want to use these ports.
4. Select the Ports and Triggering Events tab.
5. Select SAL301R_USR as the BO Type for all ports.
6. Select the following options for the From port:
   - Event Create: Main
   - Event Delete: Main
   - Event Retrieve: Retrieve
   - Event Update: Main

Figure 13-23 on page 512 shows the Ports and Triggering Events tab.
7. Click Apply and Close.
8. Select File → Save As → To Project
9. Enter BOTest as the name for the collaboration template, select the SAB404RRealtyICL project, and click Save.
10. Select File → Compile to compile the collaboration template. The template must be compiled in order to deploy it.
11. Select File → Exit.
12. In System Manager, select the Collaboration Foundation template and select Edit → Delete. Confirm the deletion.
Create BOTest_Source_to_SAL301R collaboration object

In order to run the collaboration template we need to create a collaboration object that binds the defined ports to the SourceConnector and SAL301RConnector.

Perform these steps to create the collaboration object.

1. Right-click the Collaboration Objects folder in SAB404RRealtyICL and select Create New Collaboration Object.

2. Select BOTest from the list of collaboration templates. Enter the following collaboration object name: BOTest_Source_to_SAL301R.

3. Click Next.

4. Bind the ports to the connectors by selecting them from the BindWith column:
   - DestinationAppRetrieve: SAL301RConnector
   - From: SourceConnector
   - To: SAL301RConnector

5. Click Finish.

Deploy test objects

Instead of deploying the test objects using a user project, we will deploy them by dragging them directly to the server. Perform the following steps:

1. Make sure the Interchange Server is running and that the System Manager is connected to it.

2. Stop the SAB404RConnector, SAL301RConnector and SourceConnector if these are running.

3. Select all the Business Objects we have created and drag them to the Business Object folder in SAB404RRealtyICS user project.

4. Right-click the user project and select Deploy user project.

5. Select the Interchange Server from the list of servers.
6. Select the Business Objects we have created, the SAB404RConnector and the SAL301RConnector.

7. Click **Next**.

8. Select to overwrite all objects on the server and click **Finish**.

   **Note:** For testing purposes, we will deploy the rest of the objects without going through a user project.

9. Drag the and SourceConnector to the server.

10. Drag the BOTest collaboration template to the server.

11. Drag the BOTest_Source_to_SAL301R collaboration object to the server.

12. Right-click the server and select **Shut Down → Gracefully**.

13. Start the Interchange Server and connect to it in System Manager.

14. Start the BOTest_Source_to_SAL301R collaboration if it is not running.

**Perform the test**

At this point, we have an environment where we can test if the synchronization of employee data to SAL301R will work, given that the maps and collaboration we will create later, work correctly.

During this test will set up the test environment and test the operations we support; Create, Update, Retrieve and Delete.

1. Follow these steps to set up the test environment and the test Business Object:

   a. Select **Start → Programs → IBM WebSphere Business Integration Server Express → Adapters → Connectors → SAL301R** to start the SAL301RConnector.

   b. Create a new instance of the Test Connector by selecting **Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Test Connector**.

   c. In the Test Connector, select **File → Create/Select Profile**.

   d. Select the SourceConnector and click **OK**.

   e. Select **File → Connect** to start the connector.

   f. Select **SAL301R_USR** from the list of Business Object types.

   g. Click **Create** and enter a name for the Business Object instance.

   h. Right-click the ADDRESS[n] and click **Add Instance**. Supply the data as specified in Figure 13-24 on page 514.
2. Perform these steps to test the Create action:
   a. Select **Create** from the list of verbs.
   b. Select **Request** → **Send**. To send the data to SAL301R.
   c. Switch to the console for the SAL301RConnector adapter. Observe that the Business Object is inserted into the database. Example 13-8 shows the description of the Business Object that has been inserted into the database. Note that the USRID and USRADDRID attributes have received values, since these were created automatically (the IDs may vary in your test).

   **Example 13-8  Print-out of the SAL301R_USR Business Object**

```
<Version = 3.0>
<EndHeader>
<StartBO:SAL301R_USR>
   BusinessObject = SAL301R_USR
   Verb = Create
   Locale = en_US
   USRID = 74
   USRUSERNAME = myusername
   USRPASSWORD = password
```
USRROLE = Agents
USRFIRSTNAME = John
USRLASTNAME = Doe
USREMAILADDR = CxIgnore
USRWEBURL = CxIgnore
USRPHNNUM = 919 555 2342
USRACTIVE = Y
USRCRDON = 07/01/2004 11:11:11

<StartChild>
ADDRESS = 1
<StartBO:SAL301R_USR_ADDR>
   BusinessObject = SAL301R_USR_ADDR
   Verb = CxBlank
   Locale = en_US
   USRADDRID = 102
   USRID = 74
   USRADDRNAME = OFFICE
   USRADDRCNTRYCD = US
   USRADDRPOSTCD = 27713
   USRADDRRGN = North Carolina
   USRADDRMUNI = Durham
   USRADDRSTRTNAME = Sean Francis Way
   USRADDRBLDGNUM = CxIgnore
   USRADDRUNITNUM = CxIgnore
   USRADDRPOBOX = CxIgnore
   USRADDRADDLTEXT = CxIgnore
   USRCRDON = 07/01/2004 11:11:11
   ObjectEventId = CxIgnore
<EndBO:SAL301R_USR_ADDR>
<EndChild>
   ObjectEventId = SourceConnector_1089395862373_2
<EndBO:SAL301R_USR>

d. Verify that the data has been successfully committed to the database.

3. Perform these steps to test the Update action:
   a. Switch back to the Test Connector.
   b. Select Update from the list of verbs.
   c. Change the USRLASTNAME attribute to Smith.
   d. Add the USRID values you received from the previous test and add these to the USRID attributes.
   e. Select Request → Send.
   f. Switch to the SAL301RConnector console. Observe that the Business Object is being processed.
g. Verify that the changes has been committed to the database. Take note of the new address ID.

4. Perform these steps to test the Retrieve action:
   a. Switch back to the Test Connector.
   b. Select Retrieve from the list of verbs.
   c. Select Request → Send.
   d. Switch to the console for the SAL301RConnector adapter. Verify that the address ID is the same as the one committed during the update.

5. Perform these steps to test the Delete action:
   a. Switch back to the Test Connector.
   b. Select Delete from the list of verbs.
   c. Select Request → Send.
   d. Verify that the data has been deleted from the database.

**Delete deployed test objects**

Some of the objects we have created for the test are not supposed to be present in production. We will remove the following objects:

- SourceConnector
- BOTest collaboration template
- BOTest_Source_to_SAL301R collaboration object

Delete the objects in the following manner:

1. Stop the SAL301RConnector agent by closing the console.
2. Exit the Test Connector for SourceConnector.
3. Choose to stop the BOTest_Source_to_SAL301R collaboration object.
4. Right-click the collaboration and select Delete. Accept the defaults and click OK.

**Note:** The adapter processes the child address object by first performing an SQL DELETE and then an INSERT. It does not perform an UPDATE of the child object.

In the application-specific information on the Business Object you can specify a status column to update instead of performing a physical delete. This is specified by the SCN meta-attribute. You also need to set the connector-specific `ChildUpdatePhyDelete` property to `False`. 

Note:
5. Right-click the **BOTest** collaboration template and select **Delete**. Accept the defaults and click **OK**.

6. Stop the **SourceConnector**.

7. Right-click the **SourceConnector** and select **Delete**. Accept the defaults and click **OK**.

8. Delete the same objects in the same order from the SAB404RRealtyICL project.

### 13.5.2 Relationships

In order to cross-reference unique identifiers between applications, you can create relationship tables in the Interchange Server repository that map these IDs. Between SAB404R and SAL301R we will need three relationships:

- **Employee**
  
  This dynamic relationship will map the employee IDs between the EMPLOYEE.EMP_KEY column in SAB404R and USR.USR_ID column in SAL301R.

- **Country**
  
  This relationship will contain static entries mapping the three character abbreviation for countries in SAB404R with the two character abbreviation in SAL301R, for example, from USA to US.

- **State**
  
  This static relationship will map the two-character state abbreviation in SAB404R with the full state name in SAL301R, for example, from NC to North Carolina.

**Employee dynamic relationship**

Follow these steps to create the Employee dynamic relationship:

1. Right-click the **Relationships** folder in the SAB404RRealtyICL project and select **Relationship Designer**.

2. In the Relationship Designer, replace the text NewRIt with **Employee** and press **Enter**. This is the relationship name.

3. Right-click the **Employee** relationship and select **Add Participant Definition**. Enter the name Emp301R and press **Enter**. This is the employee participant for SAL301R.

4. Repeat step 3 for Emp404R. This is the employee participant for SAB404R.

5. Repeat step 3 for Employee. This is the Interchange Server participant.

6. Right-click the **Employee** relationship and select **Advanced Settings**.
7. Set the following information as displayed in Figure 13-25:
   - Relationship type: Identity
   - URL: jdbc:db2:SMB_USER
   - Login: <db_username>
   - Password: <db_password>
   - Type: DB2

![Advanced Settings](image)

**Figure 13-25**  Employee relationship advanced settings

8. Select the **Emp301R** participant and enter the following information:
   - Table name: EMP301R
   - Stored procedure name: SP_EMP301R

9. Select the **Emp404R** participant and enter the following information:
   - Table name: EMP404R
   - Stored procedure name: SP_EMP404R

10. Select the **Employee** participant and enter the following information:
    - Table name: EMPLOYEE
    - Stored procedure name: SP_EMPLOYEE
    - Select this participant to be IBM WBI Managed.

11. Click **OK**.

12. Right-click the **Emp301R** participant and select **Participant Types**. Now we associate the participants with the Business Object attributes we wish to cross-reference.
13. Do the following to associate the SAL301R_USR.USRID attribute with the Emp301R participant.
   a. Expand the SAL301R_USR Business Object in the Participant Types window.
   b. Drag SAL301R_USR in Participant Types to Emp301R in the Relationship Designer.
   c. Drag the USRID attribute in Participant Types to SAL301R_USR in the Relationship Designer.

14. Repeat step 13 to associate SAB404R_EMPLOYEE.EMPKEY with the Emp404R participant.

15. Repeat step 13 to associate Employee.EmployeeId with the Employee participant.

16. Close the Participant Types window. The relationship should be as displayed in Figure 13-26 on page 521.

17. Select File → Save All to save the relationship.

**Country static relationship**

Follow these steps to create the Country static relationship:

1. In Relationship Designer, right-click SAB404RRealtyICL and select New Relationship.

2. Enter Country as the relationship name and press Enter.

3. Add the following participants as you did for the Employee relationship:
   - Cnt301R, which holds the country abbreviations for SAL301R
   - Cnt404R, which holds the country abbreviations for SAB404R
   - Country, which cross-references SAL301R and SAB404R

4. Right-click the Country relationship and select Advanced Settings. Set the relationship type to Static.

5. Enter the following participant information:
   - Cnt301R
     - Table name: CNT301R
     - Store procedure name: SP_CNT301R
   - Cnt404R
     - Table name: CNT404R
     - Store procedure name: SP_CNT404R
   - Country
     - Table name: COUNTRY
Store procedure name: SP_COUNTRY
This table will be managed by the Interchange Server

6. Click OK.
7. Right-click Cnt301R and select Participant Types.
8. Drag the Data symbol to all the participants. This specifies that the data loaded manually.
9. Select File → Save All to save the relationship.

**State static relationship**
Create the static relationship for states in the same manner as with the Country relationship. Enter the State-specific information as follows:

- Relationship name: State
- Participant information:
  - St301R
    - Table name: ST301R
    - Store procedure name: SP_ST301R
  - St404R
    - Table name: ST301R
    - Store procedure name: SP_ST301R
  - State
    - Table name: STATE
    - Store procedure name: SP_STATE
    - This table will be managed by the Interchange Server.

Figure 13-26 on page 521 shows the Employee, Country and State relationships, their participants, and their participant types.
Deploying relationships

Deploy the relationships in the following way:

1. Right-click the SAB404RRealtyICS user project and select Update Project.

2. In the Update project dialog box, select the newly created relationships in the SAB404RRealtyICL project.

3. Click Finish.

4. Start the Interchange Server and connect to it.

5. Right-click the SAB404RRealtyICS user project and select Deploy user project.

Note: A new relationship called NewRIt may appear in the list of dynamic relationships in System Manager. Delete this relationship.
6. Select the Interchange Server to deploy to.

7. Select the **Employee**, **Country**, and **State** relationships. Note that the **Create Schema** option is automatically selected. This means that the deployment procedure will create the following tables:
   - Employee relationship: EMP301R and EMP404R
   - Country relationship: CNT301R, CNT404R, and COUNTRY
   - State relationship: ST301R, ST404R, and STATE

8. Click **Finish**. Verify that the tables are created.

The Employee relationship is populated with data when employees are synchronized from SAB404R to SAL301R.

Contrary to the Employee relationship, the Country and State relationships need to be populated with data prior to running the synchronization. We will fill the tables with data using a database script. Proceed as follows:

1. Locate the file populateStaticRelationship.ddl.
2. Start a command line window.
3. Navigate to the folder where the DDL is located.
4. Enter the following command to run the script:
   
   ```
   db2 -td@ -vf populateStaticRelationship.ddl
   ```
5. Exit the command line window.

**Note:** If DB2 is not in PATH, you will get the following error message:

```
DB21061E Command line environment not initialized.
```

You can either add DB2 to PATH, or you can run the script in the following way:

1. Select **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Line Processor**.
2. Type `quit` and press **Enter**.
3. Follow the directions above from step 3.

Verify that the data has been populated by following these steps:

1. Restart the Interchange Server and connect to it.
2. Start the relationships in the Interchange Server Component Management outline.
3. Select **Start** → **Programs** → **IBM WebSphere Business Integration Server Express** → **Toolset Express** → **Administrative** → **Relationship Manager**.
4. In Relationship Manager, select Server → Connect.
5. Select the Interchange Server, type in the user name and password and click Connect.
6. Select File → Open.
7. Select Country from the list of relationships and click OK.
8. Choose to retrieve all instances and click Get Instances.
9. Expand all the participants and view the data. Notice how the values USA and CAN in SAB404R relate to US and CA in SAL301R.
10. Repeat steps 5 to 8 to examine the State relationship.

![Relationship Manager - Country](image)

**Figure 13-27 Country relationship data**

### 13.5.3 Maps

Maps are necessary to convert Application Specific Business Objects to Generic Business Objects, and vice versa. For the employee synchronization process we have defined the following maps:

- **SAB404R_EMPLOYEE_Employee**
  
  This specifies the mapping from the SAB404R_EMPLOYEE Application Specific Business Object to the Employee Generic Business Object. The map uses the submap Sub_SAB404R_ADDRESS_EmployeeAddress to map the child Business Objects SAB404R_ADDRESS and EmployeeAddress.

- **Employee_SAL301R_USR**
This specifies the mapping from the Employee Generic Business Object to the SAL301R_USR Application Specific Business Object. The map uses the submap Sub_EmployeeAddress_SAL301R_USR_ADDR to map the child Business Objects EmployeeAddress and SAL301R_USR_ADDR.

**SAL301R_USR_Employee**

This specifies the mapping from SAL301R_USR to Employee. The map uses the submap Sub_SAL301R_USR_ADDR_EmployeeAddress to map the child Business Objects.

**Sub_SAB404R_ADDRESS_EmployeeAddress**

This is a submap specifying the mapping from the SAB404R_ADDRESS Application Specific Business Object to the EmployeeAddress Generic Business Object.

Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select **Create New Map**.
2. Select the **SAB404RRealtyICL** project and click **Next**.
3. Select **SAB404R_ADDRESS** as the source Business Object and click **Next**.
4. Select **EmployeeAddress** as the destination Business Object and click **Next**.
5. Enter the map name: **Sub_SAB404R_ADDRESS_EmployeeAddress**.
6. The map direction is Application Specific to Generic. Click **Finish**.
7. Map the attributes as described in Table 13-10. Drag the attribute from SAB404R_ADDRESS to the attribute in EmployeeAddress to map it.

| Table 13-10  Sub_SAB404R_ADDRESS_EmployeeAddress mapping |
|--------------|-------------|-------------|
| SAB404R_ADDRESS | EmployeeAddress | Rule       |
| ADDRTYPE      | AddressType  | Move       |
| ADDRLINE1     | AddressLine1 | Move       |
| ADDRLINE2     | AddressLine2 | Move       |
| ADDRCITY      | City         | Move       |
| ADDRZIP       | PostalCode   | Move       |
| ADDRSTATE     | State        | Custom     |
| ADDRCOUNTRY   | Country      | Custom     |
8. The two-letter state abbreviation in SAB404R needs to be mapped to the full state name in SAL301R. We have created the State static relationship for this mapping:
   a. Double-click the **Custom** rule for the state mapping to open the Activity Editor.
   b. Drag the Static Lookup function from the library to the main window. The function is in the General\Relationship folder.
      This function looks up a static value in a specified relationship and takes the following parameters:
      - **relationship name**: The relationship to use for lookup. This will be a constant with value State.
      - **participant name**: The name of the participant that represents the Application Specific Business Object in the relationship. This will be a constant with value St404R.
      - **inbound?**: This value is true if the map is inbound to the Interchange Server. This will be a constant with value true.
      - **source value**: The value to use for lookup. This value will be the value from the ADDRSTATE attribute in SAB404R_ADDRESS.
   c. To create a new constant, right-click the main window and select **New Constant**. Enter a value in the yellow box, for example, State. You can select the constant type in the Properties outline.
      Create the following constants:
      - State, type string
      - St404R, type string
      - true, type boolean
   d. Expand the folder Variables and the ObjSAB404R_ADDRESS Business Object to see its attributes.
   e. Drag ADDRSTATE to the main window.
   f. Click and drag the arrow in the State constant to the arrow next to **relationship name**. This specifies that the string State is the value of the relationship name parameter in the Static Lookup function.
   g. Specify St404R as the **participant name** parameter.
   h. Specify true as the **inbound?** parameter.
   i. Specify ObjSAB404R_ADDRESS.ADDRSTATE as the **source value** parameter.
   j. Drag the arrow next to **lookup value** to ObjEmployeeAddress.State. This specifies that the value from the lookup function will be applied to the State attribute in EmployeeAddress.
k. Select **File → Save.**
l. Select **File → Close.**

Figure 13-28 shows the function that looks up the state value in the State relationship.

![Diagram of State relationship](image)

**Figure 13-28  Function for looking up the state value**

9. Repeat the steps in step 8 to retrieve the country value using the Country relationship. The parameters to the Static Lookup function are:

- relationship name: Country, string constant
- participant name: Cnt404R, string constant
- inbound?: true, boolean constant
- source value: ObjSAB404R_ADDRESS.ADDRCOUNTRY

Figure 13-29 on page 526 shows the function for retrieving the country.

![Diagram of Country relationship](image)

**Figure 13-29  Function for looking up the country value**
10. In Map Designer Express, select **File** → **Save** → **To Project**. Click **Yes** to compile the map.

11. Select **File** → **Exit**.

**Sub_EmployeeAddress_SAL301R_USR_ADDR**

This is a submap specifying the mapping from the EmployeeAddress Generic Business Object to the SAL301R_USR_ADDR Application Specific Business Object.

Follow these steps to create the map:

1. Select to create a new map:
   - **Source Business Object**: EmployeeAddress
   - **Destination Business Object**: SAL301R_USR_ADDR
   - **Name**: Sub_EmployeeAddress_SAL301R_USR_ADDR
   - **Mapping direction**: Generic to Application Specific

2. Map the attributes as described in Table 13-11.

<table>
<thead>
<tr>
<th>EmployeeAddress</th>
<th>SAL301R_USR_ADDR</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressType</td>
<td>USRADDRNAME</td>
<td>Move</td>
</tr>
<tr>
<td>AddressLine1</td>
<td>USRADDRSTRTN</td>
<td>Move</td>
</tr>
<tr>
<td>AddressLine3</td>
<td>USRADDRBLDGN</td>
<td>Move</td>
</tr>
<tr>
<td>AddressLine4</td>
<td>USRADDRUNITN</td>
<td>Move</td>
</tr>
<tr>
<td>City</td>
<td>USRADDRMUNI</td>
<td>Move</td>
</tr>
<tr>
<td>State</td>
<td>USRADDRRGN</td>
<td>Custom</td>
</tr>
<tr>
<td>PostalCode</td>
<td>USRADDRPOSTCD</td>
<td>Move</td>
</tr>
<tr>
<td>Country</td>
<td>USRADDRCNTRYCD</td>
<td>Custom</td>
</tr>
</tbody>
</table>

3. Create the function that retrieves the state from the State static relationship. The parameters to the Static Lookup function are:
   - **relationship name**: State, string constant
   - **participant name**: St301R, string constant
   - **inbound?**: false, boolean constant (this is outbound of the Interchange Server)
   - **source value**: ObjEmployeeAddress.State
4. Create the function that retrieves the country from the Country static relationship. The parameters to the Static Lookup function are:
   - relationship name: Country, String constant
   - participant name: Cnt301R, String constant
   - inbound?: false, boolean constant (this is outbound of the Interchange Server)
   - source value: ObjEmployeeAddress.Country

5. USRCRDON in SAL301R_USR_ADDR specifies when the address was created in SAL301R. This attribute is required. Since we do not have the corresponding attribute in SAB404R, we need to specify the value for it. The value will be the time at which the mapping is performed.

   Follow these steps to create the function for setting the current time:
   a. Select **Custom** from the list of rules for the attribute USRCRDON in SAL301R_USR_ADDR.
   b. Double-click the **Custom** rule to open the Activity Editor.
   c. Drag the **Now** function from the folder General\Date to the main window. The **Now** function returns a String representation of the current time in the specified date/time format. The date/time format for SAL301R is MM/dd/yyyy HH:mm:ss.
   d. Create a new constant with value **MM/dd/yyyy HH:mm:ss** and make it the input parameter to the function.
   e. Attach the return value of the function to ObjSAL301R_USR_ADDR.USRCRDON.
   f. Save the function and exit the Activity Editor.

6. Save, compile, and exit the map.

   Figure 13-30 shows the Diagram tab of the Sub_EmployeeAddress_SAL301R_USR_ADDR submap.
Sub_SAL301R_USR_ADDR_EmployeeAddress

This is a submap specifying the mapping from the SAL301R_USR_ADDR Application Specific Business Object to the EmployeeAddress Generic Business Object.

Follow these steps to create the map:

1. Create a new map:
   - Source Business Object: SAL301R_USR_ADDR
   - Destination Business Object: EmployeeAddress
   - Name: Sub_SAL301R_USR_ADDR_EmployeeAddress
   - Mapping direction: Application Specific to Generic

2. Map the attributes as described in Table 13-12.

Table 13-12  Attribute mappings for Sub_SAL301R_USR_ADDR_EmployeeAddress

<table>
<thead>
<tr>
<th>SAL301R_USR_ADDR</th>
<th>EmployeeAddress</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRADDRNAME</td>
<td>AddressType</td>
<td>Move</td>
</tr>
<tr>
<td>USRADDRCNTRYCD</td>
<td>Country</td>
<td>Custom</td>
</tr>
<tr>
<td>USRADDRPOSTCD</td>
<td>PostalCode</td>
<td>Move</td>
</tr>
<tr>
<td>USRADDRRGN</td>
<td>State</td>
<td>Custom</td>
</tr>
<tr>
<td>USRADDRMUNI</td>
<td>City</td>
<td>Move</td>
</tr>
<tr>
<td>USRADDRSTRRTNAME</td>
<td>AddressLine1</td>
<td>Move</td>
</tr>
<tr>
<td>USRADDRBLDGNUM</td>
<td>AddressLine3</td>
<td>Move</td>
</tr>
</tbody>
</table>
3. Create the function that retrieves the state from the State static relationship. The parameters to the Static Lookup function are:
   - relationship name: State, String constant
   - participant name: St301R, String constant
   - inbound?: true, boolean constant
   - source value: ObjSAL301R_USR_ADDR.USRADDRRGN

4. Create the function that retrieves the country from the Country static relationship. The parameters to the Static Lookup function are:
   - relationship name: Country, String constant
   - participant name: Cnt301R, String constant
   - inbound?: true, boolean constant
   - source value: ObjSAL301R_USR_ADDR.USRADDRCNTRYCD

5. Save, compile, and exit the map.

**SAB404R_EMPLOYEE_Employee**

This map specifies the mapping from the SAB404R_EMPLOYEE Application Specific Business Object to the Employee Generic Business Object. The map will use the Sub_SAB404R_ADDRESS_EmployeeAddress submap to map the address child objects.

1. Create a new map:
   - Source Business Object: SAB404R_EMPLOYEE
   - Destination Business Object: Employee
   - Name: SAB404R_EMPLOYEE_Employee
   - Mapping direction: Application Specific to Generic

2. We have created a dynamic relationship for cross-referencing employee identifiers between SAB404R and SAL301R. We want the map to use this relationship to map between EMPKEY in SAB404R_EMPLOYEE and Employeeld in Employee:
   a. Drag ObjSAB404R_EMPLOYEE to ObjEmployee.
   b. Select **Cross reference** from the list of rules.
   c. Select **Employee** from the list of relationships.
   d. Select **Emp404R** as the application-specific participant.
   e. Click **OK**.

3. Map the attributes as described in Table 13-13.

<table>
<thead>
<tr>
<th>SAL301R_USR_ADDR</th>
<th>EmployeeAddress</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRADDRUNITNUM</td>
<td>AddressLine4</td>
<td>Move</td>
</tr>
</tbody>
</table>
Table 13-13  Attribute mappings for SAB404R_EMPLOYEE_Employee

<table>
<thead>
<tr>
<th>SAB404R_EMPLOYEE</th>
<th>Employee</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Verb] for ObjSAB404R_EMPLOYEE</td>
<td>[Verb] for ObjEmployee</td>
<td>Move</td>
</tr>
<tr>
<td>EMPLASTNAME</td>
<td>LastName</td>
<td>Move</td>
</tr>
<tr>
<td>EMPFIRSTNAME</td>
<td>FirstName</td>
<td>Move</td>
</tr>
<tr>
<td>EMPID</td>
<td>EmployeeNumber</td>
<td>Move</td>
</tr>
<tr>
<td>EMPTAXID</td>
<td>SocialNumber</td>
<td>Move</td>
</tr>
<tr>
<td>EMPDOB</td>
<td>DateOfBirth in EmployeeHR_Info</td>
<td>Move</td>
</tr>
<tr>
<td>EMPSEX</td>
<td>Gender in EmployeeHR_Info</td>
<td>Move</td>
</tr>
<tr>
<td>EMPTYPE</td>
<td>JobTitle in EmployeeJob</td>
<td>Custom</td>
</tr>
<tr>
<td>EMPDATEHIRED</td>
<td>HireDate in EmployeeJob</td>
<td>Move</td>
</tr>
</tbody>
</table>

4. SAL301R is a role based application, meaning that the user can perform different operations depending on his or her role. SAB404R has equivalent roles specified in the EMPTYPE attribute. We will create custom code in the JobTitle attribute in EmployeeJob for mapping these roles.

a. Double-click the Custom role in the EmployeeJob.JobTitle attribute to open the Activity Editor.

b. Use the following functions and variables to the main window and specify their input parameters:
   - General\String\Trim Text (Trims white spaces before and after the string):
     in string: ObjSAB404R_EMPLOYEE.EMPTYPE
   - General\String\Text Equal (Compares two strings and returns true if they are equal)
     in string 1: Trim Text.trimmed string
     in string 2: constant value AGENT
   - General\String\If (Compares two strings to a condition value. If it is equal to the first string the function returns true)
     condition: Text Equal.are they equal?
     value 1: constant value Agents
value 2: constant value Administrators

- ObjEmployee.EmployeeJob[0].JobTitle

in value: If.result

This function specifies that if EMPTYPE is equal to AGENT, JobTitle should be Agents. If not, the JobTitle should be Administrators. The function is shown in Figure 13-31.

Figure 13-31  Function for mapping employee type in SAB404R to role in SAL301R

c. Save and exit the Activity Editor.

5. Opposed to SAB404R, the SAL301R application requires user to log in with a username and password. The Employee Business Object contains a LogonId attribute where we can specify the username. We will set the username as the first letter of the employees first name, and the full last name. The password will be set manually in the map between Employee and SAL301R_USR. Follow these steps to create the function:

a. Select the attributes EMPFIRSTNAME and EMPLASTNAME and drag them to the LogonId attribute in Employee (hold the Ctrl key to select multiple attributes).

b. Click OK in the Join dialog box.

c. Select Custom as the rule for LogonId and double-click the rule to open the Activity Editor.

d. Use the following functions and variables to the main window and specify their input parameters:

- General\String\Substring by position (returns a portion of a string based on start and end positions):
  
  **string:** ObjSAB404R_EMPLOYEE.EMPFIRSTNAME
  
  **start position:** constant int 0
end position: constant int 1

- General\String\Append Text (concatenates two strings)
  in string 1: ObjSAB404R_EMPLOYEE.EMPLASTNAME
  in string 2: Substring by position.substring

- General\String\Lower Case (converts a string to lowercase)
  fromString: Append Text.result

- ObjEmployee.LoginId
  in value: Lower Case.toString

The function is shown in Figure 13-32.

---

e. Save and exit the Activity Editor.

6. The ADDRESS attribute in SAB404R_EMPLOYEE will map to the EmployeeAddress attribute in Employee using the Sub_SAB404R_ADDRESS_EmployeeAddress submap. Perform these steps to specify the submap:

a. Drag the ADDRESS attribute to the EmployeeAddress attribute.

b. Select Submap from the list of rules.

c. Select Sub_SAB404R_ADDRESS_EmployeeAddress from the list of submaps and click **OK**.

7. Save, compile and exit the map.
**Employee_SAL301R_USR**

This map specifies the mapping from the Employee Generic Business Object to the SAL301R_USR Application Specific Business Object. It is used during a service call request from the employee collaboration. The map will use the Sub_EmployeeAddress_SAL301R_USR_ADDR submap to map the address child objects.

Follow these steps to create the map:

1. Select to create a new map:
   - Source Business Object: Employee
   - Destination Business Object: SAL301R_USR
   - Name: Employee_SAL301R_USR
   - Mapping direction: Generic to Application Specific

2. We will use the Employee dynamic relationship to map between EmployeeId in Employee to USRID in SAL301R_USR. Specify the cross-reference mapping with the following properties:
   - Relationship: Employee
   - Participant: Emp301R

3. Map the attributes as described in Table 13-14.

*Table 13-14  Attribute mappings for Employee_SAL301R_USR*

<table>
<thead>
<tr>
<th>Employee</th>
<th>SAL301R_USR</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Verb] for ObjEmployee</td>
<td>[Verb] for ObjSAL301R_USR</td>
<td>Move</td>
</tr>
<tr>
<td>FirstName</td>
<td>USRFIRSTNAME</td>
<td>Move</td>
</tr>
<tr>
<td>LastName</td>
<td>USRLASTNAME</td>
<td>Move</td>
</tr>
<tr>
<td>Telephone1</td>
<td>USRPHNNUM</td>
<td>Move</td>
</tr>
<tr>
<td>EmailAddress</td>
<td>USREMAILADDR</td>
<td>Move</td>
</tr>
<tr>
<td>LoginId</td>
<td>USRUSERNAME</td>
<td>Move</td>
</tr>
<tr>
<td>EmployeeAddress</td>
<td>ADDRESS Submap</td>
<td>Submap</td>
</tr>
<tr>
<td></td>
<td>Sub_Employee_SAL301R_USR_ADDR</td>
<td></td>
</tr>
<tr>
<td>JobTitle in EmployeeJob</td>
<td>USRROLE</td>
<td>Move</td>
</tr>
</tbody>
</table>
4. We have specified the username as the first character of the first name and the full last name. In order to log on to SAL301R the user still needs a password. For example purposes we will specify this as password:
   a. Select **Set Value** from the list of rules for USRPASSWORD.
   b. Enter the value **password** and click **OK**.
5. Set the value of USRACTIVE to **Y** (for Yes).
6. USRCRDON specifies the date and time when the user was created. Create the function for this attribute in the same manner as we did for USRCRDON in “Sub_EmployeeAddress_SAL301R_USR_ADDR” on page 527.
7. In order for the adapter to be able to insert the address into SAL301R, the USRID must be specified in the ADDRESS child object. The steps below show how to retrieve the SAL301R user ID from the Employee dynamic relationship.

   **Note:** During a Create operation, the USRID attribute will be set automatically by the adapter. The adapter does not do this during an Update.

   a. Select **Custom** from the list rules for ADDRESS.USRID and open the Activity Editor.
   b. Use the following functions and variables for the main window and specify their input parameters:
      - General\Math\String to Number (converts a string to number). Specify int as output type.
        - **string**: ObjEmployee.EmployeeId
      - General\APIs\Relationship\Retrieve Participants (retrieves participants from a relationship):
        - **relDefName**: constant string Employee
        - **partDefName**: constant string Emp301R
        - **instanceId**: String to Number.number
      - General\APIs\Participant\Array\Get Participant At (retrieves the participant at a given position in an array of participants)
        - **array**: Retrieve Participantsparticipant instances
        - **index**: constant int 0
      - General\APIs\Participant\Get Business Object Data (retrieves the Business Object associated with the participant)
        - **participant**: Get Participant Atparticipant
- General\APIs\Business Object\Get String (retrieves an attribute from the Business Object):
  Business Object: Get Business Object Data.data
  attribute: constant value USRID
- ObjSAL301R_USR.ADDRESS.USRID
  in value: Get String.value
- General\Utilities\Catch Error (catches all exceptions in the current activity).

The Get Participant At function will throw a java.lang.ArrayIndexOutOfBoundsException when there are no participant instances retrieved from the lookup. This will occur during a Create. We will not do anything with the exception.

c. The function is shown in Figure 13-33.

![Figure 13-33 Function for retrieving USRID from Employee relationship](image)

d. Save and exit the Activity Editor.

8. Save, compile and exit the map.

Figure 13-34 on page 537 shows the Table tab for the Employee_SAL301R_USR map.
Figure 13-34 Table tab of Employee_SAL301R_USR map

**SAL301R_USR_Employee**

This map specifies the mapping from the SAL301R_USR Application Specific Business Object to the Employee Generic Business Object. It is used during a service call response from the adapter to the collaboration. The map will use the Sub_SAL301R_USR_ADDR.EmployeeAddress submap to map the address child objects.

Follow these steps to create the map:

1. Select to create a new map:
   - Source Business Object: SAL301R_USR
   - Destination Business Object: Employee
   - Name: SAL301R_USR_Employee
   - Mapping direction: Application Specific to Generic

2. We will use the Employee dynamic relationship to map between USRID in SAL301R_USR to EmployeeId in Employee. Specify the cross-reference mapping with the following properties:
   - Relationship: Employee
   - Participant: Emp301R

3. Map the attributes as described in Table 13-15.

*Table 13-15 Attribute mappings for SAL301R_USR_Employee*

<table>
<thead>
<tr>
<th>Source Attribute</th>
<th>Destination Attribute</th>
<th>Transformation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjEmployee</td>
<td>ObjSAL301R_USR</td>
<td>Cross Reference</td>
</tr>
<tr>
<td>..._firstName</td>
<td>_USRFIRSTNAME</td>
<td>Move</td>
</tr>
<tr>
<td>..._lastName</td>
<td>_USR_LASTNAME</td>
<td>Move</td>
</tr>
<tr>
<td>..._emailAddress</td>
<td>_USREMAILADDR</td>
<td>Move</td>
</tr>
<tr>
<td>..._loginId</td>
<td>_USR_USERNAME</td>
<td>Move</td>
</tr>
<tr>
<td>..._telephone1</td>
<td>_USR_PHNNUM</td>
<td>Move</td>
</tr>
<tr>
<td>..._jobTitle</td>
<td>_USR_ROLE</td>
<td>Move</td>
</tr>
<tr>
<td>..._usrPassword</td>
<td>_USR_PASSWORD</td>
<td>Set value (&quot;password&quot;)</td>
</tr>
<tr>
<td>_USR_ACTIVE</td>
<td>_USR_RDOCNUM</td>
<td>Set value (&quot;Y&quot;)</td>
</tr>
<tr>
<td>_USR_ID</td>
<td>_USR_ID</td>
<td>Custom</td>
</tr>
<tr>
<td>..._employeeAddr</td>
<td>ADDRESS</td>
<td>Submap (Sub_Employee)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Attribute</th>
<th>Destination Attribute</th>
<th>Transformation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAL301R_USR</td>
<td>Employee</td>
<td>Move</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Attribute</th>
<th>Destination Attribute</th>
<th>Transformation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRFIRSTNAME</td>
<td>FirstName</td>
<td>Move</td>
</tr>
<tr>
<td>USRLASTNAME</td>
<td>LastName</td>
<td>Move</td>
</tr>
</tbody>
</table>
4. Save, compile and exit the map.

**Deploy the maps**
Follow these steps to deploy the maps we have created:
1. Right-click the SAB404RRealtyICS user project and select **Update project**.
2. Expand the SAB404RRealtyICL project and select the Maps folder.
3. Click **Finish**.
4. Right-click the **SAB404RRealtyICS** user project and select **Deploy user project**.
5. Select the **Interchange Server** as the destination.
6. Select the **Maps** folder from the objects you wish to deploy.
7. Click **Finish** to deploy the maps.

**Test the maps**
We will perform the following tests:
- A **EVENT_DELIVERY** using the SAB404R_EMPLOYEE_Employee map
- A **SERVICE_CALL_REQUEST** using the Employee_SAL301R_USR map
- A **SERVICE_CALL_RESPONSE** using the SAL301R_USR_Employee map

**Testing SAB404R_EMPLOYEE_Employee**
Follow these steps to test the SAB404R_EMPLOYEE_Employee map.
1. Connect to the Interchange Server.
2. Open the SAB404R_Employee map and select the **Test** tab.
3. Select **EVENT_DELIVERY** as the calling context.
4. Right-click the **OFFICE** attribute and select **Add Instance**.

---

<table>
<thead>
<tr>
<th>SAL301R_USR</th>
<th>Employee</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRPHNNUM</td>
<td>Telephone1</td>
<td>Move</td>
</tr>
<tr>
<td>USREMAILADDR</td>
<td>EmailAddress</td>
<td>Move</td>
</tr>
<tr>
<td>USRUSERNAME</td>
<td>LoginId</td>
<td>Move</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>EmployeeAddress</td>
<td>Submap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sub_SAL301R_USR_AD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DR_EmployeeAddress</td>
</tr>
<tr>
<td>USRROLE</td>
<td>JobTitle in EmployeeJob</td>
<td>Move</td>
</tr>
</tbody>
</table>
5. Expand the OFFICE attribute, right-click the ADDRESS attribute and select Add Instance.

6. Type in the following sample data for SAB404R_EMPLOYEE:
   - EMPKEY: 1001
   - EMPOFFICEKEY: 1234
   - OFFICE.OFFICEKEY: 1234
   - OFFICE.OFFICEADDRKEY: 2001
   - OFFICE.ADDRESS.ADDRKEY: 2001
   - OFFICE.ADDRESS.ADDRTYPE: OFFICE
   - OFFICE.ADDRESS.ADDROWNER: 1
   - OFFICE.ADDRESS.ADDRLINE1: Some street 433
   - OFFICE.ADDRESS.ADDRCITY: Durham
   - OFFICE.ADDRESS.ADDRZIP: 27713
   - OFFICE.ADDRESS.ADDRSTATE: NC
   - OFFICE.ADDRESS.ADDRCOUNTRY: USA
   - EMPTYPE: AGENT
   - EMPFIRSTNAME: John
   - EMPLASTNAME: Doe
   - EMPDATEHIRED: 01/01/2000
   - EMPID: EMP001
   - EMPDOB: 01/30/1970
   - EMPSEX: M
   - EMPTAXID: 4325322

7. Click Save To and save the Business Object to the file SAB404R_EMPLOYEE.bo.

8. Select Debug → Run Test.

9. Click Connect in the dialog box. When the test has finished, the following text appears in the console:

   Starting test run...
   Test run finished.

10. Observe the mapped data in the Employee Business Object. Notice how an EmployeeId has been generated using the relationship. Notice also that the State and Country attributes in EmployeeAddress have both been mapped to the value 1. This specifies the participant instance number in the static relationships.

   Examine the results of the custom functions in the map.

11. Click Save To on the Employee object and save it to file Employee.bo. We will use this Business Object in the test of the Employee_SAL301R_USR map.

12. Exit the map.
Testing Employee_SAL301R_USR
Follow these steps to test the Employee_SAL301R_USR map.
1. Open the Employee_SAL301R_USR map and select the Test tab.
2. Select SERVICE_CALL_REQUEST as the calling context.
3. Click Load From, select the Employee.bo file we created in the previous test and click OK.
4. Run the test.
5. Examine the results from the mapping.
6. Save the destination SAL301R_USR Business Object to SAL301R_USR.bo. We will use this file in the next test.

Testing SAL301R_USR_Employee
Follow these steps to test the SAL301R_USR_Employee map.
1. Open the SAL301R_USR_Employee map and select the Test tab.
2. Select SERVICE_CALL_RESPONSE as the calling context.
3. Select Employee from the Generic Business Object list.
4. Select ObjSAL301R_USR and click Load From. Select the file SAL301R_USR.bo from the previous test.
5. Select Employee and click Load From. Select the Employee.bo file from the SAB404R_EMPLOYEE_Employee test.
6. The test simulates the adapter having created the entries in the SAL301R database and returns the identifiers to Interchange Server for future cross-reference. Add the following information to the SAL301R_USR object:
   – USRID: 11
   – ADDRESS.USRADDRID: 22
   – ADDRESS.USRID: 11
7. Run the test.
8. At this point, the Interchange Server has mapped the employee identifiers between SAB404R and SAL301R. Verify this in the Relationship Manager:
   a. Select Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → Relationship Manager.
   b. Select Server → Connect.
   c. Select the Interchange Server, enter the user name and password, and click Connect.
   d. Select File → Open.
Chapter 13. Sample development

e. Select the **Employee** relationship and click **OK**.

f. Select to retrieve all instances and click **Get Instances**.

g. Expand the participant instance and note how the 0th instance is mapped. Figure 13-35 shows that Emp404R.EMPKEY=1001 is mapped to Employee.EmployeeId=701, which in turn is mapped to Emp301R.USRID=11.

h. If you are not performing the extra tests listed below, you need to delete the participant instance from the Relationship Manager. Right-click the instance and select **Delete Relationship Instance**.

![Relationship Manager - Employee](image)

Figure 13-35  Employee relationship with one participant instance

**Extra tests**

To test that the Update and Delete functions work properly, you can perform the tests above in the same sequence. Change the verbs to Update or Delete, respectively.

Note that during an Update, the employee ID is resolved from the relationship, and that the function retrieving the SAL301R-specific USRID works properly.

**Getting relationship values using Java**

Example 13-9 shows the equivalent Java code to Figure 13-33 on page 536 for getting the relationship values.

```java
try {
    // Get participants
    Participant[] participants =
```
```java
RelationShip.retrieveParticipants(
"Employee",
"Emp301R",
ObjEmployee.getInt("EmployeeId");
// Get the value in USRID attribute
int userId =
    participants[0].getBusObj().getInt("USRID");
// Set the attribute value
ObjSAL301R_USR.set(
    "ADDRESS.USRID",
    userId);
} catch (Exception e) {
    // Do nothing
}
```

13.5.4 Collaboration

The synchronization of employee data from SAB404R to SAL301R will support the Create, Retrieve, Update and Delete operations. The Collaboration Foundation template delivered with WebSphere Business Integration Server Express in the WebSphere BI Express Library supports these operations. Instead of creating the template ourselves, we will make a copy of the Collaboration Foundation and have it suit our requirements.

In this chapter, we will perform the following tasks:

- Create the EmployeeSync collaboration template based on the Collaboration Foundation.
- Create the EmployeeSync_SAB404R_to_SAL301R collaboration object and bind the ports to the connectors SAB404RConnector and SAL301RConnector.
- Test the collaboration using the Integrated Test Environment.

Create EmployeeSync

Perform these steps create the EmployeeSync collaboration template:

1. Copy the Collaboration Foundation template from WBIExpressLibrary to SAB404RRealtyICL.
2. Double-click the template to open it in Process Designer Express.
3. Double-click the Definitions icon in the left pane. You will receive warning messages stating that the Controller Business Object does not exist, and a question if you want to delete the ports. Click No, since we want to use these ports.
4. Select the Ports and Triggering Events tab.
5. Select **Employee** as the BO Type for all ports.

6. Select the following options for the From port:
   - Event Create: Main
   - Event Delete: Main
   - Event Retrieve: Retrieve
   - Event Update: Main

   Only the application specified through the From port can trigger the collaboration.

   Figure 13-36 shows the Ports and Triggering Events tab.

7. Click **Apply** and **Close**.

8. Select **File → Save As → To Project**.

9. Enter **EmployeeSync** as the name for the collaboration template, select the **SAB404RRealtyICL** project, and click **Save**.

10. Select **File → Compile** to compile the collaboration template. The template must be compiled in order to deploy it.

11. Select **File → Exit**.

12. In System Manager, select the **Collaboration Foundation** template and select **Edit → Delete**. Confirm the deletion.

---

**Creating EmployeeSync_SAB404R_to_SAL301R**

A collaboration template specifies the collaboration’s interfaces to the outside world, but it does not specify which applications it interfaces. This is done through a collaboration object, where you bind a connector to each port specified in the template. The EmployeeSync template has three ports:

- **DestinationApp**

    The Collaboration Foundation template has a USE_RETRIEVE property which specifies that the Business Object can be retrieved from the destination...
application before it is synchronized. This is useful when performing compensation processing and when setting the verb based on whether the Business Object already exists in the destination application.

Even though we will not be using this property, we will bind the port to SAL301R.

▶ From
This specifies the application in which Business Objects will trigger the collaboration. This port will be bound to SAB404R.

▶ To
This specifies the application to which the data will be synchronized. This port will be bound to SAL301R.

Perform these steps to create the collaboration object.
1. Right-click the Collaboration Objects folder in SAB404RRealtyICL and select Create New Collaboration Object.
2. Select EmployeeSync from the list of collaboration templates. Enter the following collaboration object name: EmployeeSync_SAB404R_to_SAL301R.
3. Click Next.
4. Bind the ports to the connectors by selecting them from the BindWith column:
   – DestinationAppRetrieve: SAL301RConnector
   – From: SAB404RConnector
   – To: SAL301RConnector
5. Click Finish.
6. Right-click the Business Objects and select Show Maps to see more details.

Deploying the collaboration
Follow these steps to deploy the collaboration:
1. Drag the EmployeeSync template to the Collaboration Templates folder in the SAB404RRealtyICL user project.
2. Drag the EmployeeSync_SAB404R_to_SAL301R object to the Collaboration Objects folder in the SAB404RRealtyICL user project.
3. Right-click the user project and select Deploy user project.
4. Select the Interchange Server from the list of servers.
5. Select the newly created collaboration template and object and click Finish.
6. Restart the Interchange Server and connect to it.
7. Start the EmployeeSync_SAB404R_to_SAL301R collaboration object if it is not running.
Testing the collaboration
We will test the collaboration using the Integrated Test Environment in
WebSphere Business Integration Server Express. This enables us to test the
collaboration and the maps by sending a Business Object from SAB404R to
SAL301R and view the results.

Follow these steps to set up the Integrated Test Environment:

1. Register test server:
   a. Right-click Interchange Server Instances in the Interchange Server
      Component Management outline, and select Register Server.
   b. Click Browse and select your Interchange Server.
   c. Enter your username and password and save them.
   d. Select the Test Server option and enter the Interchange Server install
      path, for example, C:\IBM\WebSphereServer.
   e. Click OK. Note that (Test) appears next to the Interchange Server name.
   f. Connect to the Interchange Server.

2. Right-click the EmployeeSync_SAB404R_to_SAL301R collaboration object
   and select Debug In Integrated Test Environment.

3. In the dialog box, accept the defaults and click Finish.

4. Select Integrated Test Environment → Run to start the environment. Note
   that you get two Test Connector instances to the left of the collaboration, one
   for SAB404RConnector and one for SAL301RConnector.
   a. In the SAL301RConnector, click the Connect button.
   b. Select the configuration file for the SAL301RConnector as specified in
      13.4.2, “SAL301RConnector” on page 486.
   c. Click Open.
   d. Repeat the process for the SAB404RConnector.

Figure 13-37 on page 546 shows an example view of the Integrated Test
Environment with the EmployeeSync_SAB404R_to_SAL301R collaboration
object.
Follow these steps to test the Create verb for the collaboration:

1. In the SAB404RConnector instance, select SAB404R_EMPLOYEE as the BO Type.

   **Tip:** It can be difficult to view the information in the Integrated Test Environment. Double-click the window headers to open them in full screen mode. Double-click them again to restore them.

2. Click Create to create a new Business Object definition. Enter the name Test.

3. Select Edit → Import BO from the menu (the menu is opened by clicking the arrow pointing down).

4. Select the SAB404R_EMPLOYEE.bo file you created in the map tests.

5. Select Server → Send.
6. Double-click the Business Object in the result view of SAL301RConnector.

7. Modify the following information:
   - USRID: 101
   - ADDRESS.USRADDRID: 10
   - ADDRESS.USRID: 101

8. Click OK.

9. Select Reply → Success to complete the collaboration.

10. View the relationship instance that has been created in the Relationship Manager.

    Repeat the step using the Update verb. Notice how the employee identifiers have been automatically cross-referenced using relationship.

    Repeat the step using the Delete verb.

### 13.5.5 Additional information

**SAL301R: user with multiple addresses**

In our example solution, a user may only have one address in the SAL301R application, even though the database schema supports multiple addresses for a user.

In this chapter we will describe the changes we must perform to the Application Specific Business Objects SAL301R_USR, and SAL301R_USR_ADDR to be able to support multiple addresses:

- The first step is to change the cardinality of the ADDRESS attribute to N in SAL301R_USR. This specifies that a user can have multiple addresses.
- Set the application-specific information to KEEP_RELATIONSHIP=TRUE. This specifies that connector is supposed to delete existing addresses in SAL301R before inserting the addresses represented in the Business Object. This is acceptable since users will not be able to add addresses in the SAL301R application. If we specified this attribute to TRUE, a new address would be added each time the employee is synchronized.

**Note:** You can also remove the code in the Employee_SAL301R_USR map that gets the USRID from the dynamic relationship and puts it in the child.

According to the Online Help the adapter is supposed to set all foreign-key values in each child that stores the parent/child relationship in the child. This only works for multiple cardinality, even though it is also supposed to work for single cardinality.
13.6 SAL301R to SAB404R: property synchronization

This task was allocated to three of the developers, making use of other elements developed by others in the team. The developers doing this synchronization further split the tasks to Business Objects, collaborations, maps and configuring the adapters.

13.6.1 Business Objects

In this section, we will describe the development of the Business Objects used in the Property Synchronization scenario.

- Generic Business Objects for Property synchronization
- Application Specific Business Objects for SAB404R
- Application Specific Business Objects for SAL301R
- Application Specific Business Objects for PropertyPublishing
- Application Specific Business Objects for E-mail

Table 13-16 lists the Business Objects we will create in this chapter.

<table>
<thead>
<tr>
<th>Business Object</th>
<th>Type</th>
<th>Create using</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>GBO</td>
<td>Manually</td>
<td>Custom Business Object.</td>
</tr>
<tr>
<td>Address</td>
<td>GBO</td>
<td>Re-use</td>
<td>Generic Business Object that comes with the base product.</td>
</tr>
<tr>
<td>SAL301R_PROP</td>
<td>ASBO</td>
<td>JDBC ODA</td>
<td>Represents the PROP table in the SAL301R database</td>
</tr>
<tr>
<td>SAL301R_PROP_STS</td>
<td>ASBO</td>
<td>JDBC ODA</td>
<td>Represents the PROP_STS table in the SAL301R database</td>
</tr>
<tr>
<td>SAL301R_PROP_TYP</td>
<td>ASBO</td>
<td>JDBC ODA</td>
<td>Represents the PROP_TYPE table in the SAL301R database</td>
</tr>
<tr>
<td>SAB404R_INVENTORY</td>
<td>ASBO</td>
<td>JDBC ODA</td>
<td>Represents the INVENTORY table in the SAB404R database</td>
</tr>
<tr>
<td>SAB404R_ADDRESS</td>
<td>ASBO</td>
<td>JDBC ODA</td>
<td>Represents the ADDRESS table in the SAB404R database</td>
</tr>
</tbody>
</table>
This chapter describes how we created the Generic Business Objects Address and Property.

**Address**

The WBIExpressLibrary contains an address definition that we can use.

Copy the Address object to the SAB404RRealtyICL project. Right-click the Address definition in the WBIExpressLibrary\Business Objects folder and select Copy. Paste it into the SAB404RRealtyICL\Business Objects folder.

**Property**

Follow the steps below to create the Generic Business Object Property.

1. Open the Business Object Designer Express tool.

   Click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Business Object Designer Express.

---

### Generic Business Objects for the property synchronization

<table>
<thead>
<tr>
<th>Business Object</th>
<th>Type</th>
<th>Create using</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP_publishProperty_N1525933095_TLO</td>
<td>ASBO</td>
<td>WS ODA</td>
<td>Represents the parameters required by the Web service Property Publishing. This object contains multiple child objects not listed here.</td>
</tr>
<tr>
<td>LocalProtocolAttr</td>
<td>ASBO</td>
<td>Manually</td>
<td>Child of the SOAP_publishProperty object. Contains meta-data.</td>
</tr>
<tr>
<td>LocalProtocolAttr</td>
<td>ASBO</td>
<td>Manually</td>
<td>Child of the SOAP_publishProperty object. Contains meta-data.</td>
</tr>
<tr>
<td>LocalTransRule</td>
<td>ASBO</td>
<td>Manually</td>
<td>Child of the SOAP_publishProperty object. Contains meta-data.</td>
</tr>
<tr>
<td>EMailNotification</td>
<td>ASBO</td>
<td>Modify base version.</td>
<td>Represents the parameters required by the e-mail application</td>
</tr>
<tr>
<td>XML__Property</td>
<td>ASBO</td>
<td>XML ODA</td>
<td>Child of the modified EMailNotification object</td>
</tr>
</tbody>
</table>
2. Create a new object.

Click **File → New**. In the New Business Object window, enter **Property** in the field Business Object Name. Select **SAB404RRealtyICL** from the project drop-down list.

3. Specify Object level details

In the Business Object Designer Express tool, switch to the General page. In the field Business Object Level application-specific information, enter the text **GBO**. Make sure that the Supported Verbs tables contains Create, Delete, Retrieve and Update.

4. Specify object attributes.

Switch to the Attributes page. Right-click **row 1** and select **Insert Above**. See Figure 13-38.

Add attributes to your object definition until it looks like Figure 13-39.
5. Save to project.
Save the Property object definition to the SAB404RRealtyICL project by clicking Ctrl+S.

6. Copy object definition to the SAB404RRealtyICS project.
Right-click the Property definition in the System Manager. Select Copy and paste it in the SAB404RRealtyICS\Business Object folder.

Application Specific Business Objects for SAB404R
This section describes how to create the Application Specific Business Objects SAB404R_INVENTORY and SAB404R_ADDRESS using the JDBCODA.

1. Prepare the JDBC ODA.
For details on how to set up the JDBC ODA, look at “Creating the ASBOs” on page 188.

2. Start JDBC ODA runtime component.
Double-click the start_JDBCODA.bat file in the <ProductDir>\WebSphereServer\ODA\JDBC directory.

3. Start the Business Object Designer Express.
Click Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Business Object Designer Express.

4. Start the ODA Wizard.
In the Business Object Designer, click File → New Using ODA.... In the next window, click the Find Agents button and select the JDBCODA entry in the Located Agents field.

5. Connect to the SAB404R data source.
In the Configure Agent window, enter the values as shown in Table 13-17.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserName</td>
<td>db2admin</td>
</tr>
<tr>
<td>Password</td>
<td>smbP4$$</td>
</tr>
<tr>
<td>DatabaseURL</td>
<td>jdbc:db2:SAB404R</td>
</tr>
<tr>
<td>DatabaseDriver</td>
<td>COM.ibm.db2.jdbc.app.DB2Driver</td>
</tr>
<tr>
<td>DefaultBOPrefix</td>
<td>SAB404R_</td>
</tr>
</tbody>
</table>

6. Click Next.
7. Database tables:
   Expand the entry **DB2ADMIN** → **Tables**. Select the tables **ADDRESS** and **INVENTORY** then click the **Next** button twice.

8. Set Business Object properties.
   In the window **BO Properties - For all the Tables selected** change the value for the property **Add Stored Procedure Attributes** from **yes** to **no**. Click **OK**.

   In the last window that appears make sure you save the definition to the SAB404RRealtyICL project. Select the options **Open the new Business Object in new window** and **Shut down JDBCODA**. Click **Finish**.

10. Add an address attribute to the inventory definition.
    Change the name of the INVADDRKEY attribute to INVADDR_FK. Modify the value of the application-specific information column to look like **Example 13-10**.

    **Example 13-10 INVADDR_FK application-specific information**
    
    | CN=INV_ADDR_KEY:FK=SAB404R_ADDRESS.ADDRKEY:CONTAINMENT=OWNERSHIP |

    Right-click the INVADDR_FK row and select **Insert below**. In the empty row that appears, enter the values as displayed in Table 13-18.

    **Table 13-18 Values for the ADDRKEY attribute.**
    
    | Column                             | Value                     |
    |------------------------------------|---------------------------|
    | Name                               | ADDRKEY                   |
    | Type                               | SAB404R_ADDRESS           |
    | Cardinality                        | 1                         |
    | Application-specific information   | CONTAINMENT=OWNERSHIP     |

11. Modify the SAB404R_INVENTORY.INVKEY attribute.
    Change the application-specific information for this attribute. Modify the text to look like **Example 13-11**.

    **Example 13-11 INVKEY application-specific information**
    
    | CN=INV_KEY:UID=AUTO |

12. Modify the SAB404R_ADDRESS.ADDRKEY attribute.
    Change the application-specific information for this attribute. Modify the text to look similar to **Example 13-12 on page 553**.
Chapter 13. Sample development

13. Save the SAB404R_INVENTORY and SAB404R_ADDRESS definitions

   Use Ctrl+S to save the object definitions to the SAB404RRRLRealtyICL project. Copy and paste the object definitions to the SAB40RRRLRealtyICS user project.


   “Testing the SAL301R Business Objects” on page 510 describes how to test the SAL301R adapter and the Business Object SAL301R_USR. Follow that description to create a test collaboration named BOTest_Source_to_SAB404R and use that to test the SAB404R adapter and the Business Object SAB404R_INVENTORY.

   **Note:** Remember to test the creation, deletion and updating of an object.

**Application Specific Business Objects for SAL301R**

This chapter describes how to create the Application Specific Business Objects SAL301R_PROP, SAL301R_PROP_STS and SAL301R_TYPE using the JDBCODA.

1. Create the initial definitions using the JDBCODA.

   Please look at 7.3.4, “Creating Business Objects” on page 188 for instruction on how to create the initial definitions.

2. Modify the SAL301R_PROP definition

   Please look at 7.3.5, “Modifying the ASBOs” on page 190 for instructions on how to include SAL301R_PROP_STS and SAL301R_PROP_TYPE as child objects.

   Change the application-specific information for the attributes displayed in Table 13-19.

**Table 13-19  Application specification for the SAL301R_PROP definition**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Application-specific information</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPID</td>
<td>CN=PROP_ID:UID=AUTO</td>
</tr>
<tr>
<td>PROPTYPEID_FK</td>
<td>CN=PROP_TYPE_ID:CONTAINMENT=NO_OWNERSHIP:FK=SAL301R_PROP_TYPE.PROPTYPEID</td>
</tr>
<tr>
<td>PROPSTSID_FK</td>
<td>CN=PROP_STATUS_ID:CONTAINMENT=NO_OWNERSHIP:FK=SAL301R_PROP_STS.PROPSTSID</td>
</tr>
</tbody>
</table>

3. Modify the SAL301R_PROP_TYPE definition
Change the application-specific information for the attributes displayed in Table 13-20.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Application-specific information</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPTYPEID</td>
<td>CN=PROP_TYPE_ID:UID=AUTO</td>
</tr>
</tbody>
</table>

4. Modify the SAL301R_PROP_STS definition

Change the application-specific information for the attributes displayed in Table 13-21.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Application-specific information</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPSTSID</td>
<td>CN=PROP_STS_ID:UID=AUTO</td>
</tr>
</tbody>
</table>

5. Save the object definitions.

Use **Ctrl+s** to save the object definitions to the SAB404RRealtyICL project. Copy and paste the object definitions to the SAB40RRealtyICS user project.

6. Testing the Business Objects and the adapter.

“Testing the SAL301R Business Objects” on page 510 describes how to test the SAL301R adapter and the Business Object SAL301R_USR. Follow that description to create a test collaboration named BOTest_Source_to_SAL31R_PROP and use it to test the SAL301R adapter and the Business Object SAL301R_PROP.

**Note:** Remember to test the creation, deletion and updating of an object.

**Application Specific Business Objects for PropertyPublishing**

This section describes how to create the Application Specific Business Objects for the PropertyPublishing Web service. The initial version of the SOAP_publishProperty_N1525933095_TLO object will be created using the WSDL ODA.

1. Copy re-usable meta objects to the SAB404RRealtyICL

   From the WBIExpressLibrary\Business Objects folder copy all the object definitions beginning with MO_ to the SAB404RRealtyICL\Business Objects folder. Make them available in the user project by copying them from SAB404RRealtyICL\Business Objects to SAB404RRealtyICS\Business Objects.

2. Start the WSDL ODA runtime component.
Double-click the `start_WSDLODA.bat` file in the `<ProductDir>\WebSphereServer\ODA\WSDL` folder.

3. Open the Business Object Designer Express.

   ![Start ➔ Programs ➔ IBM WebSphere Business Integration Server Express ➔ Toolset Express ➔ Development ➔ Business Object Designer Express.](image)

4. Start the ODA Wizard.

   In the Business Object Designer, click **File ➔ New Using ODA...** In the next window, click the **Find Agents** button and select the **WSDLODA** entry in the Located Agents field.

5. Connect to the WSDL source file.

   In the Configure Agent window, enter the values as shown in Table 13-22.

   ![Table 13-22 Configure Agent values](image)

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSDL_URL</td>
<td><code>C:\&lt;your path&gt;\SAL404R\SAB404RAgencyWeb\WebContent\wsdl\com\ibm\itso\newsservice\PropertyPublisher.wsdl</code></td>
</tr>
<tr>
<td>WebService</td>
<td><code>propertyPublisher</code></td>
</tr>
</tbody>
</table>

6. Click **Next**.

7. Select `webservice`.

   In the Select Source window expand **PropertyPublisherService ➔ PropertyPublisher ➔ PropertyPublisherSoapBinding**. Select the publishProperty node and click **Next** and **Next** again.

8. Save generated Business Objects.

   In the Save Business Object definitions window select the SAB404RRealtyICL project in the Save to project dropdown list. Select the options **Open new Business Object definitions** and **Shutdown the WSDL ODA**. Click **Finish**. Copy the generated object definitions to the SAB404RRealtyICS project.

9. Modify the SOAP_publishProperty_N1525933095_TLO object definition.

   Select the Key column for the Handler attribute and save the object definition by clicking **Ctrl+S**. Copy the object definition to the SAB404RRealtyICS project.

In the Business Object Designer Express click **File** and **New**. In the Business Object Name field, enter the text `LocalTransRule`. Modify the object definition to look like Figure 13-40.

<table>
<thead>
<tr>
<th>General</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos</td>
<td>Name</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>ContentTy</td>
</tr>
<tr>
<td>2</td>
<td>MimeTy</td>
</tr>
<tr>
<td>3</td>
<td>CharSet</td>
</tr>
<tr>
<td>4</td>
<td>ObjectEventId</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 13-40   LocalTransRule Business Object definition.*

Copy the object definition to the SAB404RRealtyICS project.

11. Create the Business Object `LocalTransMap`.

In the Business Object Designer Express click **File** and **New**. In the Business Object Name field enter the text `LocalTransRule`. Modify the object definition to look like Figure 13-41.

<table>
<thead>
<tr>
<th>General</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos</td>
<td>Name</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>TransformationRule</td>
</tr>
<tr>
<td>2</td>
<td>ObjectEventId</td>
</tr>
</tbody>
</table>

*Figure 13-41   LocalTransMap Business Object definition.*

Copy the object definition to the SAB404RRealtyICS project.

12. Create the Business Object `LocalProtocolAttr`

In the Business Object Designer Express, click **File** and **New**. In the Business Object Name field, enter the text `LocalProtocolAttr`. Modify the object definition to look similar to Figure 13-42.

<table>
<thead>
<tr>
<th>General</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos</td>
<td>Name</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>userProperty1</td>
</tr>
<tr>
<td>2</td>
<td>ObjectEventId</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 13-42   LocalProtocolAttr Business Object definition.*
13. Copy the object definition to the SAB404RRealtyICS project.

14. Modify the SOAP_publishProperty_N1525933095_Request_ConfigMO object definition.

   Open the object definition from the SAB404RRRealtyICL\Busines Objects folder in the System Manager. Add a default value to the attributes in

\[Table 13-23 \hspace{1em} \text{Default values for the Request_ConfigMO object definition}\]

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeInfo</td>
<td>false</td>
</tr>
<tr>
<td>TypeCheck</td>
<td>none</td>
</tr>
</tbody>
</table>

15. Save the object definition by clicking \texttt{Ctrl+s}.

16. Modify the SOAP_publishProperty_N1525933095_Protocol_Req_ConfigMO object definition.

   Open the object definition from the SAB404RRRealtyICL\Busines Objects folder in the System Manager. Add a default value to the attribute in

\[Table 13-24 \hspace{1em} \text{Default value for the Protocol_Req_ConfigMO object definition}\]

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Default values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>text/xml</td>
</tr>
</tbody>
</table>

Add the two attributes displayed in Table 13-25 to the object definition.

\[Table 13-25 \hspace{1em} \text{Add new attributes to the Protocol_Req_ConfigMO object definition}\]

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserDefinedProperties</td>
<td>LocalProtocolAttr</td>
<td>1</td>
</tr>
<tr>
<td>MessageTransformationMap</td>
<td>LocalTransMap</td>
<td>1</td>
</tr>
</tbody>
</table>

17. Testing the Business Objects and the adapter.

   “Testing the SAL301R Business Objects” on page 510 describes how to test the SAL301R adapter and the Business Object SAL301R_USR. Follow that description to create a test collaboration named BTest_Source_to_PropertyPublishing and use that to test the PropertyPublishing adapter and the Business Object SOAP_publishProperty_N1525933095_TLO.
**Application Specific Business Objects for E-mail**

This chapter describes how to create the Application Specific Business Objects EMailNotification and XML__Property for the E-mail service.

1. **Copy from WBIExpressLibrary**
   
   Copy the existing Business Object EMailNotification from the WBIExpressLibrary\Business Objects to the SAB404RRealtyICL\Business Objects.

2. **Connect to the Property source file**
   
   Double-click the Property object definition in System Manager. Select File and Save Copy to File. Remember where you store the copy.

3. **Start the XML ODA**
   
   Double-click the start_XMLODA.bat file in the <ProductDir>\WebSphereServer\ODA\XML directory.

4. **Start the ODA Wizard.**
   
   In the Business Object Designer click File → New Using ODA... On the next window click the Find Agents button and select the XMLODA entry in the Located Agents field.

5. **Connect to the source file**
   
   In the Configure Agent window, enter the values as displayed in Table 13-26.

   **Table 13-26  Connect to Property.xsd source file**

<table>
<thead>
<tr>
<th>Property name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>C:&lt;your path&gt;\Property.xsd</td>
</tr>
<tr>
<td>Root</td>
<td>Property</td>
</tr>
<tr>
<td>BOPrefix</td>
<td>XML__</td>
</tr>
</tbody>
</table>

   Click Next, Next, Next and OK.

6. **Save to project.**
   
   In the Save Business Object definitions window, select the SAB404RRealtyICL project in the Save to project drop-down list. Select the options Open new Business Object definitions and Shutdown the XML.
ODA. Click Finish. Copy the generated object definitions to the SAB404RRealtyICS project.

7. Modify the EMailNotification Business Object definition.

Open the object definition from the SAB4040RRRealtyICl\Business Objects folder in the System Manager and add the attributes displayed in Table 13-27.

Table 13-27 Add attributes to the EMailNotification Business Object definition

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Required attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML_Property</td>
<td>XML_Property</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>AttachmentExtension</td>
<td>String</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mime_Type</td>
<td>String</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

8. Save the object definition by clicking Ctrl+s.


“Testing the SAL301R Business Objects” on page 510 describes how to test the SAL301R adapter and the Business Object SAL301R_USR. Follow that description to create a test collaboration named BOTest_Source_to_EMail and use that to test the E-mail adapter and the Business Object EMailNotification.

Note: An e-mail account must be available before you can run this test. A successful result should send and e-mail with an attachment to the e-mail account you specified.

13.6.2 Maps

For the property synchronization process we have defined the following maps:

- SAL301R_PROP_Property
  This specifies the mapping from the Application Specific Business Object SAL301R_PROP to the Generic Business Object Property.

- SAB404R_Property_Inventory
  This specifies the mapping from the Generic Business Object Property to the Application Specific Business Object SAB404R_INVENTORY.

- SAB404R_Inventory_Property
  This specifies the mapping from the Application Specific Business Object SAL301R_PROP to the Generic Business Object Property.
- Property_EMailNotification
  This specifies the mapping from the Generic Business Object Property to the Application Specific Business Object EMailNotification.
- EMailNotification_Property
  This specifies the mapping from the Application Specific Business Object EMailNotification to the Generic Business Object Property.
- Property_PublishingService
  This specifies the mapping from the Generic Business Object Property to the Application Specific Business Object SOAP_publishProperty_N1525933095_TLO.
- PublishingService_Property
  This specifies the mapping from the Application Specific Business Object SOAP_publishProperty_N1525933095_TLO to the Generic Business Object Property.

**SAL301R_PROP_Property**
Follow these steps to create the map:
1. Right-click the Maps folder in SAB404RRealtyICL and select **Create New Map**.
2. Select the SAB404RRealtyICL project and click **Next**.
3. Select **SAL301R_PROP** as the source Business Object and click **Next**.
4. Select **Property** as the destination Business Object and click **Next**.
5. Enter the map name: **SAL301R_PROP_Property**
6. The map direction is **Application Specific to Generic**. Click **Finish**.
7. Map the attributes as described in Figure 13-43 on page 561. Drag the attribute from SAL301R_PROP to the corresponding attribute into Property.
Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select **Create New Map**.
2. Select the SAB404RRealtyICL project and click **Next**.
3. Select **Property** as the source Business Object and click **Next**.
4. Select SAB404R_INVENTORY as the destination Business Object and click **Next**.
5. Enter the map name: SAB404R_Property_Inventory
6. The map direction is **Generic to Application Specific**. Click **Finish**.

Map the attributes as described in Figure 13-44 on page 562. Drag the attributes from Property to the corresponding attribute in SAB404R_INVENTORY.
This map has two custom rules:

- Property.INV_DATE_SOLD to SAB404R_INVENTORY.INVDATESOLD

- Property.INV_DATE_LISTED to SAB404R_INVENTORY.INVDATELISTED
SAB404R_Inventory_Property
Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select Create New Map.
2. Select the SAB404RRealtyICL project and click Next.
3. Select SAB404R_INVENTORY as the source Business Object and click Next.
4. Select Property as the destination Business Object and click Next.
5. Enter the map name: SAB404R_Inventory_Property
6. The map direction is Application Specific to Generic. Click Finish.

Map the attributes as described in Figure 13-47. Drag the attributes from SAB404R_INVENTORY to the corresponding attribute in Property.

Figure 13-47  SAB404R_INVENTORY to Property
Property_EMailNotification
Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select Create New Map.
2. Select the SAB404RRealtyICL project and click Next.
3. Select Property as the source Business Object and click Next.
4. Select EMailNotification as the destination Business Object and click Next.
5. Enter the map name: Property_EMailNotification
6. The map direction is Generic to Application Specific. Click Finish.

Map the attributes as described in Figure 13-48 and Figure 13-49 on page 565. Drag the attributes from Property to the corresponding attribute in EMailNotification.

![Property to EMailNotification part 1 diagram]

Figure 13-48  Property to EMailNotification part 1
Note: This e-mail adapter instance in the figure above will send an e-mail to the account specified in the RecipientName field. Change the rule for this attribute to suite your environment.

**EMailNotification_Property**

Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select **Create New Map**.
2. Select the SAB404RRealtyICL project and click **Next**.
3. Select EMailNotification as the source Business Object and click **Next**.
4. Select Property as the destination Business Object and click **Next**.
5. Enter the map name: EMailNotification_Property

6. The map direction is Application Specific to Generic. Click Finish.

Map the attributes as described in Figure 13-50 and Figure 13-51 on page 567. Drag the attributes from Property to the corresponding attribute in EMailNotification.

![Figure 13-50  EMailNotification to Property part 1](image-url)
Property_PublishingService

Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select Create New Map.

2. Select the SAB404RRealtyICL project and click Next.

3. Select Property as the source Business Object and click Next.

4. Select SOAP_publishProperty_N1525933095_TLO as the destination Business Object and click Next.

5. Enter the map name: Property_PublishingService.

6. The map direction is Generic to Application Specific. Click Finish.
Map the attributes as described in Figure 13-52. Drag the attributes from Property to the corresponding attribute in SOAP_publishProperty_N1525933095_TLO.

**PublishingService_Property**

Follow these steps to create the map:

1. Right-click the Maps folder in SAB404RRealtyICL and select **Create New Map**.
2. Select the SAB404RRealtyICL project and click **Next**.
3. Select **SOAP_publishProperty_N1525933095_TLO** as the source Business Object and click **Next**.
4. Select **Property** as the destination Business Object and click **Next**.
5. Enter the map name: **Property_PublishingService**.
6. The map direction is Generic to Application Specific. Click **Finish**.

Map the attributes. Drag the attributes from SOAP_publishProperty_N1525933095_TLO to the corresponding attribute in Property.
13.6.3 Updating the adapter configuration

The Business Objects and maps have to be added to the adapter configuration. Follow the steps below to update the configuration for e-mail, SAL301R, SAB404R and PropertyPublishing adapters.

1. Open the Connector Configuration tool.
   Double-click the instances in the System Manager tool.

2. Open the Supported Business Objects page.
   Add a Business Object by clicking an empty row and selecting a Business Object definition. For all Application Specific Business Objects, select the Agent Support option. Use Ctrl+S to save to the project.

3. Associate maps to Business Objects.
   Switch to the Associated Maps page. The Business Objects you entered in the previous step will be visible in the Business Object Name column. Select the map in the Associated Map column that you want your Business Object to use. If there is more than one alternative, select the Explicit Binding option. Use Ctrl+S to save to the project.

4. Update the SAL301R Connector instance.
   Follow steps 1 to 3 for the SAL301R connector instance. On the Business Object page add the Property and SAL301R_PROP. On the Associated Maps page select SAL301R_PROP_Property.

5. Update the SAB404R Connector instance.
   Follow step 1 to 3 for the SAB404R connector instance. On the Business Object page add the Property and SAB404R_INVENTORY. Select Agent Support for the SAB404R_INVENTORY object. On the Associated Maps page, select SAB404R_Property_Inventory for the Property object. Select the Explicit Binding option. For the SAB404R_INVENTORY object, select the map SAB404R_Inventory_Property. Select the Explicit Binding option.
6. Update the EMailConnector instance.

Follow steps 1 to 3 for the EMailConnector instance. On the Business Object page, add Property, MO_EMail_Default, MO_DataHandler_Default and EMailNotification. Select the Agent Support option for all the objects except Property. On the Associated Maps page, select Property_EmailNotification for the Property object and EMailNotification_Property for the EMailNotification object. Select the Explicit Binding option for both.

7. Update the PropertyPublishingConnector.

Follow steps 1 to 3 for the PropertyPublishingConnector instance. On the Business Object page, add Property, MO_DataHandler_Default and SOAP_publishProperty_N1525933095_TLO. Select the Agent Support option for all the objects except Property. On the Associated Maps page, select Property_PublishingService for the Property object. Select the Explicit Binding option for both.

### 13.6.4 Relationship

Between SAB404R and SAL301R, the property synchronization scenario will need one relationship.

**Note:** 13.3.2, “Database and stored procedures” on page 477 must be completed before you can create any new relationships.

- **Property**

  This dynamic relationship will map the property IDs between the SAB404R_INVENTORY.INVKEY column in SAB404R and the SAL301R_PROP.PROPID column in SAL301R.

**Property dynamic relationship**

Follow these steps to create the Property dynamic relationship:

1. Right-click the **Relationships** folder in the SAB404RRealtyICL project and select **Relationship Designer**.

2. In the Relationship Designer, replace the text NewRIt with Property and press Enter. This is the relationship name.

3. Right-click the Property relationship and select **Add Participant Definition**. Enter the name Prop301R and press Enter. This is the prop participant for SAL301R.

4. Repeat step 3 for Inv404R. This is the inventory participant for SAB404R.

5. Repeat step 3 for Property. This is the Interchange Server participant.
6. Right-click the **Property** relationship and select **Advanced Settings**.

7. Set the following information as displayed in Figure 13-54:
   - Relationship type: *Identity*
   - URL: *jdbc:db2:SMB_USER*
   - Login: <db_username>
   - Password: <db_password>
   - Type: *DB2*

8. In the Advanced settings window, select the **Prop301R** participant and enter the following information:
   - Table name: *PROP*
   - Stored procedure name: *SPPROP*

9. In the Advanced settings window, select the **Invp404R** participant and enter the following information:
   - Table name: *INVENTORY*
   - Stored procedure name: *SP_INVENTORY*

10. Select the **Property** participant and enter the following information:
    - Table name: *PROPERTY*
    - Stored procedure name: *SPPROPERTY*
    - Select this participant to be **IBM WBI Managed**.

11. Click **OK** to close the Advanced settings window.
12. Right-click the **Prop301R** participant and select **Participant Types**. Now we associate the participants with the Business Object attributes we wish to cross-reference.

13. Do the following to associate the SAL301R_PROP.ID attribute with the Prop301R participant.
   a. Expand the SAL301R_PROP Business Object in the Participant Types window.
   b. Drag SAL301R_PROP in Participant Types to Prop301R in the Relationship Designer.
   c. Drag the PROPID attribute in Participant Types to SAL301R_PROP in the Relationship Designer.

14. Repeat step 13 to associate SAB404R_INVENTORY.INVKEY with the Inv404R participant.

15. Repeat step 13 to associate PROPERTY.INV_KEY with the Property participant.

16. Close the Participant Types window. The relationship should look like Figure 13-55.

17. Select **File** → **Save All** to save the relationship.

**Deploying the relationship**

Follow the steps in “Deploying relationships” on page 521.
13.6.5 Collaborations

This collaboration template and object will be used to connect the connectors, maps, and BOs accordingly using triggers to create, update, and delete a property in SAL301R will either create, update, or delete inventory in SAB404R, the Web service or send an e-mail about the property.

1. Open the System Manager by navigating to Start → Programs → WebSphere Business Integration Server Express → Toolset Express → Administrative → System Manager.

2. In the WBIExpressLibrary, expand the collaboration template and open InventorySynchronization by double-clicking the template.

3. In the Process Designer, navigate to File → Save As → To Project and select SAB404RRealtyICL then name the collaboration template InventorySync.

4. Now the editing will be done to the new template InventorySync in the SAB404RRealtyICL project.

5. In the left window, open the template definitions by double-clicking Definitions. Click the Ports and Triggering Events tab.

6. Change the BO Type to Property and change the relevant triggers to correspond to the trigger verbs as in Figure 13-56.

   In this collaboration port, To refers to SAB404R, port To1 refers to Webservices, port To2 refers to E-mail, and port From refers to SAL301R.

7. Open the Send Verb scenario by double-clicking the scenario.

Note: The Property relationship is a dynamic relationship. The database tables INVENTORY, PROP and PROPERTY will be populated/updated when a property is registered in the SAL301R application.
8. Navigate to **View → UIDs**. This will display the IDs for each node.

9. In the Send Verb scenario, change the Label of action node 183 by double-clicking the node and change the name to **Create SAB404R**. Click **Apply** then **Close**. The scenario appears as shown in Figure 13-57.

   a. Hold down the **Ctrl** key on your keyboard and select **action node 183** and **service node 212** then go to **Edit → copy then Edit → paste**.

   b. Change the newly pasted action node's Label to **Create Webservice** and the code Fragment to `To1BusObj = processingBusObj.duplicate();` click **Apply** then **Close**.

   c. Then double-click the newly pasted service node to open the properties and change the port to **To1** and the triggering Business Object to **To1BusObj** and keep the verb as **Create**. Click **Apply** then **Close**.

   d. Then click **Edit → paste** and this time change the newly pasted action node's Label to **Create E-mail** and the code Fragment to `To2BusObj = processingBusObj.duplicate();` click **Apply** then **Close**.

   e. Then, for the newly pasted service node, change the port to **To2** and the triggering Business Object to **To2BusObj** and keep the verb as **Create**. Click **Apply** then **Close**.

10. Continue by changing the name of action node 186 to **Update SAB404R**. Click **Apply** then **Close**. The scenario is similar to that shown in Figure 13-58.
a. Then hold down the Ctrl key on your keyboard and select action node 186 and service node 408 and go to Edit → copy then Edit → paste.

b. Change the newly pasted action node's Label to Update WebService and the code Fragment to To1BusObj = processingBusObj.duplicate(); click Apply then Close.

c. Then double-click the newly pasted service node to open the properties and change the port to To1 and the triggering Business Object to To1BusObj and keep the verb as Update. Click Apply then Close.

d. Then click Edit → paste and this time change the newly pasted action node's Label to Update E-mail and the code Fragment to To2BusObj = processingBusObj.duplicate(); click Apply then Close.

e. Then, for the newly pasted service node, change the port to To2 and the triggering Business Object to To2BusObj and keep the verb as Update. Click Apply then Close.

Figure 13-58  Add action nodes and services nodes for Update 1

11. In the Send Verb scenario, change the Label of action node 189 by double-clicking the node and change the name to Create SAB404R. Click Apply then Close. The scenario appears as shown in Figure 13-59.
a. Hold down the Ctrl key on your keyboard and select **action node 189** and **service node 420** then go to **Edit → copy** then **Edit → paste**.

b. Change the newly pasted action node’s Label to **Create Webservice** and the code Fragment to `To1BusObj = processingBusObj.duplicate();` click **Apply then Close**.

c. Double-click the newly pasted service node to open the properties and change the port to `To1` and the triggering Business Object to `To1BusObj` and keep the verb as **Create**. Click **Apply then Close**.

d. Then click **Edit → paste** and this time change the newly pasted action node’s Label to **Create E-mail** and the code Fragment to `To2BusObj = processingBusObj.duplicate();` click **Apply then Close**.

e. Then, for the newly pasted service node, change the port to `To2` and the triggering Business Object to `To2BusObj` and keep the verb as **Create**. Click **Apply then Close**.

![Figure 13-59  Add action nodes and services nodes for Create 2](image)

12. Then save the collaboration template to a file by navigating to **File → Save → To File**. Name the file `InventorySync.cwt`.

13. Then save the collaboration template to the project by navigating to **File → Save → To project**. Use the Project SAB404RRealtyICL as a collaboration template; the name should be `InventorySync`.

14. Then compile the collaboration template by navigating to **File → Compile**. The collaboration template should compile successfully. If not then in the
message window in the Process Designer, there will be a location listed that refers to the UID in the collaboration scenario where the compile failed. If all the nodes are not connected or you missed changing a code fragment or port, the compile may be unsuccessful. Once the compile is successful, you may close the Process Designer.

15. Next, create a collaboration object that uses the collaboration template to connect all the parts of the solution. In SAB404RRRealtyICL, right-click **Collaboration Objects** and select **Create a new collaboration object**.

   a. This step starts the collaboration object wizard. Select **InventorySync** as the collaboration template and name the collaboration object **InventorySync_SAL301R_to_SAB404R_Webservice_Email** and click **Next**. This is shown in Figure 13-60.

   ![Figure 13-60: New collaboration object window](image)

   **Figure 13-60**  New collaboration object window

   b. Then, select the correct connectors in which to bind each port to in the **InventorySync** collaboration. This is shown in Figure 13-61 on page 578. Then click **Next**.
c. The next step sets the general properties of the collaboration object change:
   - Effective transactional level to Best Effort
   - System Trace Level to 5 - Add detailed message contents
   - Collaboration trace level to 5

d. Click **Next**.

e. The next step displays more properties of the collaboration object, click **Next** then **Finish**. The new collaboration object should open and appear similar to the one shown in Figure 13-62.
16. The final step is to check whether the collaboration works by creating a new property in SAL301R and determining whether the information is passed to the SAB404R, Webservice and E-mail adapters.

17. Publish the collaboration template and the collaboration object to the WebSphereICS. First, stop all connectors used by the collaboration and all maps. Then drag the InventorySync collaboration template and drop it in the WebSphereICS. Do the same with InventorySync_SAL301R_to_SAB404R_Webservices_Email.

18. In the System Manager, use the Shutdown → Gracefully option.

13.6.6 Testing the property synchronization scenario

This chapter will describe how you can deploy and test the property synchronization scenario.

1. Deploy InterChange server artefacts

In the System Manager, right-click the SAL404RRealtyICS user project and select Deploy User Project.... In the Deployable Components window, select the destination server and components to deploy.

*Note: Even though the dropdown box is gray, you can select from it.*

Follow the description above and deploy the objects that we have created in the following order:

- Templates
- Relationships
- Business Objects
- Maps
- Collaboration Objects
- Connectors

Pay attention to the InterChange server log statements to make sure that all the elements were deployed successfully. When all the objects have been deployed, stop the server (click System Manager → Shutdown → Gracefully).

2. Restart the server by navigating to Start → Programs → WebSphere Business Integration Server Express → InterChange Server Express. Connect to the server from the System Manager and verify that the Relationships, Maps, Connectors and Collaboration objects are running. If not, start them.

3. Start all the adapter agents.

Use the shortcuts we created earlier to start the adapter agents.
Example 13-13  Path to adapter shortcuts

<your path>\WebSphereServer\connectors\EMail\  
<your path>\WebSphereServer\connectors\PropertyPublishing  
<your path>\WebSphereServer\connectors\SAB404R  
<your path>\WebSphereServer\connectors\SAL301R  

When you complete this step, you should have four adapter agents polling for events. This is shown in Figure 13-63.

![Figure 13-63  Adapter Agents started](image)

4. Start SAL301R and SAB404R

Publish and start the SAL301R and SAB404R applications in your WebSphere Studio development environment.

**Note:** Run the application server on port 7080. If you use another port, remember to update the destination attribute in the SOAP_publisProperty_N1525933095_Protocol_Req_ConfigMO object.

5. Test the Web service

Use the WebServices Explorer tool in WebSphere Studio to test the PropertyPublishing Web service. A successful test will add an entry in the log file specified in the log4j.xml file.

**Note:** SAB404RAgencyWeb\properties\log4j.xml folder specifies the location of the log file.
6. Test the scenario using a create event. Register a new property in the SAL301R application.

**Note:** You will have to log on as an agent to be able register a new Property.

Pay attention to the Adapter Agent logs. They will show you how the test progresses. A successful test means the following:

- A new row should be created in the INVENTORY and ADDRESS tables in the SAB404R database.
- An e-mail should be sent to your e-mail account.
- A new line should be added to the log file used by the PropertyPublishing Web service.
- The Property relationship should be updated.

Open the Relationship manager by clicking **Start → Program → WBI Server Express → Toolset Express → Administrative → Relationship Manager**. Connect to it and get all instances in the Property relationship. Verify that the IDs in the relationship are the same as the IDs in the SAL301R and SAB404R databases. Refer to Figure 13-64.

**Figure 13-64  Relationship details**
13.7 Server Access Interface

This chapter will introduce the Interchange Server Express Server Access Interface, discuss scenarios where it is useful and provide an easy, worked example of how to set it up and use it.

After working through this document, the reader will be able to configure a collaboration for Server Access Interface calls, write an access client and diagnose some of the more common errors.

The goal of this chapter is to allow you to get a simple example up and running with a minimum of effort. This is driven by the observation that the first step is always the hardest.

**Important:** This sample assumes that you are working with existing Business Objects, connectors, and maps from the proceeding chapters. A working solution is included in the GettingStartedICL and SAB404R_Server_Access_Interface projects.

13.7.1 Server Access Interface overview

The Server Access Interface (SAI) is a mechanism to allow collaborations to be synchronously invoked by programs running externally to the Interchange Server Express.

This contrasts the regular collaboration execution mechanism where connectors detect events and publish them asynchronously to the Interchange Server Express. That is: in the usual environment, a connector publishes an event, possibly archives it and then moves on to the next task; it does not wait for the collaboration to complete, and does not base any future processing on the outcome of each collaboration's execution.

By contrast, if an application wishes to call a collaboration as part of its processing (perhaps in order to make a decision or to obtain data from another system), then the Server Access Interface is the technology of choice.

The Server Access Interface is a collection of programming language interfaces, defined using CORBA Interface Definition Language (IDL). However, the most common binding is the Java one. This allows Java programs to be written that can call a collaboration almost as simply as calling a method.

The communication with the InterChange Server is performed behind the scenes using IIOP (Internet Inter-ORB Protocol). Configuring this part of the system is often the most complicated.
It is important to realize the following things about the Server Access Interface when used out of the box.

- Transaction context cannot be flowed with calls to the SAI.

Users of J2EE Application Servers are accustomed to declarative management of transactions when calling resources such as Enterprise JavaBeans (EJB). No such support is provided in the current implementation of the Server Access Interface. This means that the work of the collaboration will be performed as soon as the call to execute the collaboration is made and is not part of the Application Server’s transaction.

**Important:** Note that the Interchange Server model is that a single collaboration includes several independent transactions. The compensation technology is designed to allow some degree of control over what happens if one or more of these transactions fails.

- There is no load-balancing or fail-over support.

As part of the setup of an SAI client, a particular InterChange Server name must be given. If that server goes down or is busy, there is no automatic routing to an alternative server.

**Important:** If load-balancing or fail-over support is required, this must be added separately into the stack

- You must commit to block indefinitely.

The documentation implies there is no way to specify a time-out on a call to a collaboration. Therefore, it seems possible that an indefinite wait can occur.

**Important:** If a fixed wait must be obtained, an alternative would be to use an asynchronous transport such as MQSeries and configure the collaboration to write the result to a given queue. The driving application could perform a get-with-wait for the response.

If a more generic and standardized way for accessing collaborations is desired, the WebServices connector would be the technology of choice.

### 13.7.2 Server Access Interface sample overview

Since the purpose of this chapter is to enable the reader to get an exceedingly trivial SAI example running, the solution you will develop in this chapter is very simple.
The user specifies a CustomerID (integer) on the command line.

A request Business Object Customer is sent to CrossWorlds® using the Server Access Interface with the 'Retrieve' verb set.

The collaboration retrieves the Business Object using the SAL301 connector.

The Business Object is returned to the original user.

The Business Object is displayed on the user's screen.

This demonstration allows you to develop a simple SAI access client and collaboration. Once this step is complete, you are free to add extra functionality as appropriate.

### 13.7.3 Collaboration

Create a collaboration based on Collaboration Foundation and name it SAI_Retrieve. We are using for this example the Retrieve scenario. To configure the collaboration, change the Business Object type for each port to Customer and connect the Retrieve verb of the From port with the Retrieve scenario. See Figure 13-65.

![Process Designer Express - SAI_Retrieve - GettingStartedICL - [Template Definitions]](image)

**Figure 13-65 Set collaboration properties**

Open the Retrieve scenario and double-click the To.Retrieve service call icon to access it's properties. For the service call, specify the triggeringBusObj Java Reference to be the BOVariable. The result of the service call will be to update the...
Business Object referenced by triggeringBusObj with the values provided in the connector. See Figure 13-66.

**Tip:** *triggeringBusObj* is a special reference: changes to this Business Object will be sent as the result of the SAI call-back to the client application.

![ServiceCall Properties: ServiceCall_3](image)

*Figure 13-66  Set ServiceCall properties*

### 13.7.4 Business Objects

The Business Objects that you will use during this example are:

- Customer Generic Business Object.
- SAL301R_USR Application Specific Business Object

Both Business Objects are part of the GettingSamplesICL.

### 13.7.5 Maps

You will be using the existing Customer_SAL301R_USR and SAL301R_USR_Customer maps.
13.7.6 Connector configuration

Start the Connector Configurator Express and open the SAL301RConnector.

Make sure that:

- The SAL301R_USR and Customer Business Objects are listed under **Supported Business Objects**.
- Agent Support is selected for SAL301R_USR.
- See Figure 13-71 on page 594.
- You may need to redeploy the connector to Interchange Server Express.

![Connector Configuration](image)

**Figure 13-67 Connector configuration**

13.7.7 Creating the collaboration object

Now that the SAL30RConnector has been configured to support the Customer and the SAL301RUSR Business Objects, it is possible to directly create a collaboration object based on the template you have created (rather than having to create it and then bind the ports later).

1. Go to the **GettingStartedICL** project.
2. Select the **Collaboration Objects** folder.
3. Right-click and select **Create New Collaboration Object**.
4. Select the **SAI_Retrieve** template and provide a name for the collaboration object, for instance **SAI_Retrieve_Customer**.
5. Click **Next**.
6. Select the **SAL301RConnector** to bind with the **DestinationAppRetrieve** and the **To** port.

7. Leave the **From** port empty.

8. Click **Next**.

9. You may change the trace level if you want to see additional trace messages.

10. Click **Finish**.

11. The collaboration object is opened in graphical view of the System Manager.

12. Right-click the **From** port and select **Bind Port**.

13. A **Configure Port** window opens up.

14. Select **Incoming** as type.

15. Select the **SAL301R_USR** Business Object and drag it onto the Configure Port window.

16. Select **Destination Business Object** in the Business Object Type dialog and click **OK**.
17. Click OK to close the window.

18. You are now ready to deploy the collaboration to your Interchange Server Express.

### 13.7.8 Server Access Interface client development

The Server Access Interface allows programs running externally to the Interchange Server Express to synchronously request execution of collaborations. This means that to use the Server Access Interface, one must write some code!

It is usual to use an Integrated Development Environment (such as WebSphere Studio Application Developer) to develop such applications but this is not a
requirement. All that is required is that the correct resources are available on the Java Virtual Machine’s classpath and that some particular resources are present on the local machine. Fortunately the System Manager is based on WebSphere Business Integration Workbench and provides a Java development perspective.

You need to include following Java Archive (JAR) in your Java build path:

C:\IBM\WebSphereServer\lib\crossworlds.jar

**What environmental information is required?**

In order to connect to the Interchange Server Express, an access client needs to know the following information:

- Username and password (e.g. admin/null)
- Name of InterChange server (e.g. CrossWorlds)
- Location of ‘ior’ file. This file (which is usually recreated each time the Interchange Server is restarted - but this problem can be fixed; see the SAI Manual) is required by the client to locate the CORBA resources it needs
- Collaboration Name and Port Name

When examining the sample source code for the client, you will need to adapt these settings to match your environment.

1. Switch to the Java development perspective of your System Manager.
   a. Click **Select a Perspective** icon of the left icon bar.
   b. If **Java** is on the list, select it.
   c. Otherwise, click **Others**, select **Java** in the next dialog and click **OK**.

2. Create a new Java project.
   a. Select **File → New → Project**.
   b. Select **Java and Java Project**.
   c. Click **Next**.
   d. Provide a name, for Instance **SAB404R_Server_Access_Interface**.
   e. Click **Next**.
   f. Select the **Libraries** tab.
   g. Click **Add External JARs**.
   h. Browse to C:\IBM\WebSphereServer\lib and select **crossworlds.jar**.
   i. Click **OK** and **Finish** to finish the creation of the project.
   j. Import the sample Java program into your project.
   i. Right-click the **SAB404R_Server_Access_Interface** project.
   ii. Select **Import**.
   iii. Select **Filesytem**.
iv. Browse to the location where you had extracted the additional files.

v. Select `SAITest.java`.

vi. Extend the destination folder name with `/com/sab404r/sai`.

vii. Click Finish

3. Review the Java code.

4. Make sure that your Interchange Server Express and the connector `SAL301RConnector` are up and running.

5. Click the running man icon to configure and start your Java program.

6. Provide an numerical argument, for instance 1.

7. Click Run.

Tip: If you click the running man icon a second time, the execution starts with the last configuration.
8. Reviewing the console output, you will see the sent and received Business Object.

### 13.7.9 Extending the sample

Since the SAI provides synchronous execution of collaborations, it is common to allow collaboration logic to be exposed over the Web.

That is: a servlet (or a JSP) - as part of its execution - calls into the SAI to do some work on its behalf. Thus, to implement a Web page which can provide a list of outstanding invoices for a customer, one would write a servlet that calls a collaboration which retrieves the data from SAP.

In the sample code above, we called the `accessSession.createComponentObject("Customer")` method. This creates an empty Business Object which we can then populate with the necessary data before sending it into Interchange Server Express.
This is a perfectly valid way to proceed. However, once we accept the possibility of exposing a collaboration on the Web via a Web Application Server, the possibility for more than just human interaction exists.

It then seems natural to allow a collaboration to be triggered based on incoming XML, for example.

One could write code in the servlet to parse the XML and turn it into a Business Object but code already exists to do that in the CrossWorlds system: the XML data handler. More generally, the use of DataHandlers allows text strings to be converted to Business Objects.

The Server Access Interface allows DataHandlers to be invoked. Since the primary method for declaring a byte stream's format is to specify its MIME-type, the Server Access Interface allows a MIME type to be used to determine which DataHandler gets called.

Some configuration is required on the server end (to ensure the Server's Data Handler meta-object is aware of the mappings) but it really is no more complicated than that.

13.8 Additional material, tips and tricks

The remainder of this chapter is a collection of tips, observations and tricks.

13.8.1 Triggers, Events, Archive and other control tables

We are using the JDBC connector for both the SAL301R Web application and the AGENCY application. For both of these applications, we will need to configure the JDBC connector.

We could use the same database and tables for both applications, but we will use discreet tables in each of the applications.

In the JDBC connector documentation (jdbc.pdf), in the chapter Installing and configuring the connector, there is a section about SQL scripts. The documentation points to a set of template scripts.

To prepare for the JDBC connector, it was necessary to customize these scripts. For convenience, we combined these scripts into a single script for each of the application databases.
13.8.2 Setting up queues for JMS

With the WebSphere Business Integration Server Express Plus product, there is a file included that contains template definitions for creating MQ queues used by the installation `crossworlds_mq.tst`. This file is found in the folder `C:\IBM\WebSphereServer\mqseries` in a normal install.

The part of the file that is important to us is the block of local queues used for the adaptors when configured to use JMS for transport. The queue list in the template file looks like this:

```
DEFINE QLOCAL(AdapterName/AdminInQueue)
DEFINE QLOCAL(AdapterName/AdminOutQueue)
DEFINE QLOCAL(AdapterName/DeliveryQueue)
DEFINE QLOCAL(AdapterName/RequestQueue)
DEFINE QLOCAL(AdapterName/ResponseQueue)
DEFINE QLOCAL(AdapterName/FaultQueue)
DEFINE QLOCAL(AdapterName/SynchronousRequestQueue)
DEFINE QLOCAL(AdapterName/SynchronousResponseQueue)
```

Not all queues are used in all contexts and for a production installation, you would want to determine which queues are used. For example, there are different queues defined for synchronous versus asynchronous message processing.

**Tip:** With WebSphere MQ 5.3 and later, you can determine easily if a queue is in use in MQSeries Explorer from the Queues view by right-clicking the queue and clicking the **status...** menu item. In the pop-up window, WebSphere MQ will show what processes are attached to the queue if it is in use. If there are no processes attached to the queue, it is not in use. You can also check the number of Opens for input or output.

When the connector is configured to use JMS, the tool will add all the queue entries and default some names. We used the standard names as generated by the configuration and created entries for the WebSphere MQ `runmqsc` script to match.

The connector configuration with JMS queues looks like the window shown in Figure 13-71 on page 594.

**Important:** WebSphere MQ queue names are case sensitive. All queue names configured in the Connector configuration must match those created in WebSphere MQ exactly.
Our script for the source connector had the following values:

```
DEFINE QLOCAL(SOURCECONNECTOR/ADMININQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/ADMINOUTQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/DELIVERYQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/REQUESTQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/RESPONSEQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/FAULTQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/SYNCHRONOUSREQUESTQUEUE)
DEFINE QLOCAL(SOURCECONNECTOR/SYNCHRONOUSRESPONSEQUEUE)
```

These values saved in a file can be used to create the queues in WebSphere MQ by opening a command prompt and clicking **Start → Programs → Accessories → Command Prompt**.

Change to the directory where the MQSeries script is and run the following command:

```
runmqsc < yourfile.txt
```
This should run for a short period and after each queue there should be the line:
AMQ8006: WebSphere MQ queue created

If you run the script again, the queue will already exist and you will get the message for each queue:
AMQ8150: WebSphere MQ object already exists.

Consult your MQSeries documentation if other messages appear.

13.8.3 Changing the verification connectors to use JMS

As part of understanding how the JMS connectivity works and what configuration changes were necessary to use JMS, we converted the verification SourceConnector and DestinationConnector to use JMS. This required a number of changes to the connectors and the creation of profiles for the TestConnector to use so that these connectors could be tested.

Setting up the sample connectors for JMS

For the SourceConnector and DestinationConnector shipped as part of the standard product and used for installation verification, to be converted to use JMS as the transport, a number of changes are necessary.

These changes are made from System Manager. Start System Manager from the Start menu by clicking Start → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → System Manager. From the System Manager perspective, in the top left pane, expand the WBIExpressLibrary option and the connectors option by clicking the plus sign. Double-click the SourceConnector and DestinationConnector in turn and perform the following actions:

1. Find the row with the property DeliveryTransport, click the value in the property column and click JMS in the drop-down list. This will cause the number and values of the properties in the grid to change.

2. Find the row with the property jms.MessageBrokerName and update it to the value of the queue manager created during the install. The default value created as part of the installation is WebSphereICS.queue.manager.

3. Save the updates to a new file and to the project. Note where this file is since it will be necessary for using the TestConnector.

4. Deploy the changes to the server. The connectors need to stopped on the server prior to deploying the updated connectors. Expand the server instance in the lower left hand pane, as well as the connectors, right-click each connector and click the stop connectorname menu item. Also stop any instances of the TestConnector. The simplest way to deploy the connectors is
to click the connectors in the top right hand pane, holding down the Ctrl key and dragging and dropping onto the Connectors value in the lower left hand pane. A warning message should appear.

5. Stop the Interchange Server Instance and restart. Then connect the instance again for further monitoring.

**Performing the verification with the reconfigured connectors**

Once the changes are deployed, the initial installation verification can be performed again. The scenario used is described in the document README_SystemTest_Sample.rtf. This document can be accessed by clicking Start → IBM WebSphere Business Integration Server Express → Samples → System Test → Readme System Test Sample.

The differences are that to use the new SourceConnector and DestinationConnectors configured to use JMS, we need to create a new profile for each connector. Start the TestConnector twice, once for the source connector and once for the destination connector: click Start → IBM WebSphere Business Integration Server Express → Toolset Express → Development → Test Connector.

Using the source Test Connector dialog, click File → Create/Select Profile. When the Connector Profile dialog appears, click the Create New Profile button with the plus symbol in the top left hand side as seen in Figure 13-72.

![Connector Profile dialog](image)

*Figure 13-72 Selecting the TestConnector Profile*

After clicking the Create New Profile button, a new dialog will appear where the details of the profile are entered. Figure 13-73 on page 597 shows the values we used for this test.
Figure 13-73  New Profile for the Source TestConnector

The values in the dialog are:

► The file name of the connector file as saved in “Performing the verification with the reconfigured connectors” on page 596.

► A name for the connector. Since this is standing in for the real connector, we used the original name and appended JMS to the end. This value is not intrinsic to the values in the server.

► Click the Broker Type drop-down list and click the Interchange Server value.

► For JMS, the values of Server and Password are not important. Use a dummy value for both.

Once configured, the verification as described in the installation document can be followed with JMS being used as the delivery transport.

Peeking at the JMS messages

By stopping the Destination connector and sending a request from the source connector using asynchronous mode, it is possible to view the WebSphere MQ message sent to the destination connector on the queue. For example, using our configuration, we had a message on the queue:

DESTINATIONCONNECTOR/REQUESTQUEUE

The message had the following interesting properties:

► MQMD properties:
  – MQ Message format - MQHRF2. WebSphere MQ uses MQHRF2 headers to implement JMS.
  – Persistent - true. As documented, the WebSphere Business Integration uses persistent messages for messaging.
  – Message Type - Request.
- Reply to queue and queue manager populated. To work in a distributed environment, these values need to be set.
- Expiry - value of -1, never to expire. Messages will stay on the queue until read or deleted and with the Persistent flag set, will survive a reboot of the host server.
- Code page 819 instead of the default of 437 on the standard platform.

- The RFH2 header has a JMS folder containing Destination and Reply To information.
- The RFH2 header has a user folder containing XML information about the request type.
- The data is mixed ASCII and binary data.

**Tip:** The utility RFHUTIL.EXE available from the IBM Web site is a useful tool for interacting with messages on WebSphere MQ queues.

Support pacs can be found at:

http://ibm.com/software/integration/support/supportpacs/
- Category: Category 2 - Freeware
- Product: WebSphere Business Integration Message Broker

The Current Tool number is IH03 and titled WBI Message Broker V5 - Message display, test and performance utilities.

### 13.8.4 Using CVS with System Manager

In this chapter, we will provide a brief overview of CVS and how to use it with System Manager.

**Overview of CVS**

CVS (Concurrent Versions System) is an open-source version control system (see [http://www.cvshome.org](http://www.cvshome.org)). There are many advantages of having version control in projects, for example:

- It stores the files in a central repository, eliminating the vulnerability of having the code only on each developer’s workstation.
- It is an easy way of distributing code between developers.
- It keeps a history of changes made to the files. This also makes it possible to retrieve earlier versions of code.
- Keeping files in a central repository that always has the latest code makes it easier for a deployment manager to build and deploy applications.
There are two types of version control systems:

- **Pessimistic version control**
  
  In pessimistic version control systems, you must check out files before you can edit them. The system keeps the file locked for everyone else until the person that has checked it out, checks it in again. This ensures that no files are in conflict.

  Rational ClearCase is a pessimistic version control system.

- **Optimistic version control**
  
  In optimistic version control systems, you do not check out code. There are no safeguards preventing two developers working on the same file. When the file is committed to the server, the developer receives a conflict warning stating that someone else has edited the file since you retrieved it from the repository. The developer is then required to resolve the conflict by merging the changes.

  CVS is an optimistic version control system.

There are advantages and disadvantages to both pessimistic and optimistic version control systems. The most important thing is that you use a version control system.

**CVS and System Manager**

System Manager is based on the Eclipse open-source IDE platform (see [http://www.eclipse.org](http://www.eclipse.org)). Eclipse supports team development through a CVS plug-in, and you can therefore use CVS in System Manager. In this chapter, we will describe how to use CVS in System Manager, and some of the limitations as opposed to standard Eclipse CVS functionality.

**Note:** In this chapter, we only cover the basic functions of CVS. Refer to the CVS help for more information about using CVS in Eclipse.

**Setting up a CVS location**

Here, we assume that a CVS server is available. Follow these steps to set up a CVS location:

1. Open the CVS perspective by selecting **Window → Open Perspective → CVS Repository Exploring**.

2. Right-click within the CVS Repositories outline, and select **New → Repository Location**.

3. Provide the CVS information (if you do not know this information, ask the person that set up the CVS server).
4. Click **Finish**.

**Sharing projects**

Follow these steps to put a project into CVS:

1. Right-click the project you wish to share and select **Team → Share Project**.
2. Select the repository location you have created.
3. Click **Next**.
4. By default, the module name in CVS will be the project name. However, you may enter a different module name. Click **Finish** to create the module in CVS.
5. The Synchronize outline will appear in Outgoing Mode. The project has not been shared properly until you have committed all the resources to the CVS server.

   Right-click the project in the outline, and select **Commit**.
6. You are asked if you wish to add the resources to version control. Click **Yes**.
7. Enter a commit comment, for example, **Initial commit**. Click **OK**.

The project is now available for sharing in the CVS server. Other team members may now check out the project by following these steps:

1. Open the CVS Repository Exploring perspective.
2. Expand the repository location, and expand HEAD.
3. Right-click the project to check out and select **Check Out As Project**.
4. Go back to the System Manager perspective to view the project.

**Note:** You may have to restart System Manager in order to view the project.

**Synchronizing resources**

Follow these steps to synchronize outgoing and incoming changes in a project:

1. Right-click the project (or folder, or resource) you wish to synchronize, and select **Team → Synchronize with Repository**.
2. In the Synchronize outline, you can view incoming and outgoing changes using the buttons as shown in Figure 13-74 on page 601.
3. To commit changes, right-click the folder or resource, and select **Commit**. If it is a new resource, accept to add it to version control. Provide a commit comment.

You can only commit changes in outgoing or incoming/outgoing mode.

**Note:** It is good practice to *always* provide a commit comment. This helps to track changes.

**Note:** If you have deleted a resource, you *must* commit the folder that contained the resource.

4. To update from the repository, right-click the folder or resource you wish to retrieve, and select **Update from Repository**.

You can only update from the repository in incoming or incoming/outgoing mode.

5. If there are conflicts, the status bar will notify you, specifying the number of conflicts. The text in the status bar also becomes red. Click the **Show only conflicts** and the **Incoming/Outgoing mode** button to view all conflicts.

6. Follow these steps if you wish to merge the file:

   a. Double-click the resource to compare your changes with the version in CVS. The local file is on the left, while the repository file is on the right. Merge the changes by using the merge tool bar.

   b. To save the local file, right-click within it and select **Save**. The file will appear with an asterisk next to it. This means that it has been merged.

   c. However, you have to mark the file as merged before you can commit it to CVS. Right-click the file and select **Mark as Merged**. This will make the file an outgoing change, and will not be visible if you selected the **Show only conflicts** option.
7. You can override the file in CVS with your local version by doing the following:
   a. Right-click the file and select **Override and Commit**.
   b. Select to release the changes.
   c. Enter a commit comment and click **OK**.

8. If you wish to override the local version of your file with the file in CVS, right-click the file and select **Override and Update**.

**Limitations of using CVS in System Manager**

System Manager is based on Eclipse, and should therefore have the same CVS functionality as Eclipse. However, this is not the case. We discovered that the System Manager lacked a lot of the basic CVS functions that are provided in Eclipse.

**Note:** We used WebSphere Studio Workbench V2.0.3, which is based on Eclipse V2.0.3. The limitations we have listed may have been fixed in later releases.

We found the following limitations:

- **File content type**

  System Manager ignores the file content type preferences. This means that you cannot tell the CVS plug-in to regard files with a specific extension as ASCII (or binary). This is a problem, since the CVS plug-in regards for example Business Object files (.xsd) as binary, when they are actually ASCII files (XML).

  The end result is that you cannot merge these files, since they cannot be specified as ASCII. However, you may see the contents of previous versions in ASCII format if you choose to view them in the resource history.

- **Ignored resources**

  Eclipse provides you with the ability to set up which files to ignore when synchronizing with CVS. You can specify file extensions, full file names, or file name patterns.

  In System Manager, this functionality has been removed. For example, you cannot specify that class files are to be ignored.

  You also cannot add files and file patterns to the .cvsignore file.

- **Label decorations**

  In Eclipse, you can specify CVS label decorations. This enables you to view which resources are in CVS, and the current version number of the resource you have locally. CVS label decorations are not available in System Manager.

  You may specify them, but the settings will be ignored.
Deploying the redbook sample code

This appendix describes the steps needed to install and configure the sample application discussed in this book and discusses and provides information about all the necessary prerequisites.

The process described is complete. It includes some of the material discussed throughout the book and is the final position of the solution development described in Chapter 13, “Sample development” on page 469.
Prerequisites

This sample application was designed to illustrate the development of an integration solution between two existing applications.

WebSphere Business Integration Server Express Plus and WebSphere Application Server - Express were used to develop the samples and must be installed to successfully install and execute our integration sample. For more information, see Chapter 4, “Installation and configuration” on page 59.

The installation of WebSphere Business Integration Server Express Plus is discussed in this book. Installing WebSphere Business Integration Server Express Plus will install DB2 and MQSeries, which are required.

The goal of the sample is to show WebSphere Business Integration Server Express Plus being used to do integration when neither existing application required a change.

A small number of additional products, components, or samples are required for the sample to be fully functional.

SAB404R material

Our redbook sample code and other required material can be downloaded from the Web. See Appendix B, “Additional material” on page 643 for instructions.

After downloading the zipped file, extract the contents to the c:\ directory or another directory. This will create a subdirectory called SAB404R that contains sample and configuration files and other subdirectories. This description will assume the material has been extracted to the base c:\SAB404R directory.

The extracted directory contents should include the material described in Table A-1.

Table A-1   SAB404R Package Contents

<table>
<thead>
<tr>
<th>File/Directory</th>
<th>Location and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAB404RAgencyEAR.ear</td>
<td>c:\SAB404R SAB404R WebSphere Application Server package for the Legacy SAB404R application.</td>
</tr>
<tr>
<td>SAB404R.ddl</td>
<td>c:\SAB404R\DDL Database scripts to create and populate the sample databases for SAL301R and SAB404R with data at a common starting point</td>
</tr>
</tbody>
</table>
To leverage existing sample applications, the sample for WebSphere Application Server - Express, a development example, was used as one of the sample applications (SAL301R). The SAL301R sample application can be downloaded from:

ftp://www.redbooks.ibm.com/redbooks/SG246301/SG246301.zip

The SAL301R application must be installed and verified prior to installing any of the material for this redbook.

**SAL301R sample application**

To leverage existing sample applications, the sample for WebSphere Application Server - Express, a development example, was used as one of the sample applications (SAL301R). The SAL301R sample application can be downloaded from:

ftp://www.redbooks.ibm.com/redbooks/SG246301/SG246301.zip

The SAL301R application must be installed and verified prior to installing any of the material for this redbook.
For the sample solution supplied in this book, the data in SAL301R is refreshed. This is purely to provide a known starting point for the two applications SAL301R and SAB404R. The database structure of SAL301R is unchanged. The data is refreshed as part of “Creating the legacy sample database - SAB404R” on page 608.

**C compiler**

This sample demonstrates relationships used in collaborations. To make these functional, there must be a C compiler available to compile DB2 stored procedures.

**How to obtain a C compiler**

The GCC free open-source C compiler is available and has been used successfully to compile DB2 stored procedures. Installation and configuration instructions can be found at:


**Preparing to compile stored procedures**

If you have installed the GCC compiler discussed above and followed the procedures described, then you do not need to read this section. If you are using another C compiler you will need to:

- Make sure that you have a system wide environment variable DB2PATH that points to your DB2 base path. You will probably have to reboot after setting this value.

- Prepare a Windows batch file (for example, DB2CL.BAT), that compiles and links the C file prepared by DB2 into a windows DLL. The first parameter, %1, will be the name of the C file without an extension. For example, this will be something like Pnnnnnnn.C. Use this in your batch file to access the name of the source file. DB2 will also have prepared a DEF file with the name of the entry point. This file will need to be used by the linker of your C compiler. You will need the DB2 include and library files to compile and link these C programs.

- Inform DB2 of what the command is for compiling DB2 stored procedures by running the command db2set. The command will be similar to:

  ```
  db2set DB2_SQLROUTINE_COMPILE_COMMAND="c:\somepath\DB2CL.bat SQLROUTINE_FILENAME"
  ```

  The GCC link referred to in “How to obtain a C compiler” on page 606 describes the background to compiling stored procedures for DB2.
SMTP service

The sample in this book shows integration with e-mail. An SMTP server must be available for the e-mail integration to function correctly.

Prerequisite checklist

Table A-2 lists a summary of all the steps that need to be performed prior to the loading of our sample.

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Business Integration Server Express Plus</td>
<td></td>
</tr>
<tr>
<td>WebSphere Application Server - Express</td>
<td></td>
</tr>
<tr>
<td>IBM Redbook sample for WebShere Application Server - Express (SAL301R)</td>
<td></td>
</tr>
<tr>
<td>C Compiler and DB2 stored procedure configuration</td>
<td></td>
</tr>
<tr>
<td>SMTP server</td>
<td></td>
</tr>
<tr>
<td>Our redbook sample - SAB404R, downloaded and unzipped</td>
<td></td>
</tr>
</tbody>
</table>
Components of our sample application

This book contains sample application code as well as the components of WebSphere Business Integration Server Express Plus developed to implement the integration solution.

The components required are:

- The sample legacy database SAB404R. See “Creating the legacy sample database - SAB404R” on page 608.
- The legacy application SAB404R. See “Deploying the legacy agency system - SAB404R” on page 609.
- Database triggers and event tables. See “Creating the database triggers and event tables” on page 625.
- Repository information for relationships. See “Importing the relationships” on page 627.
- WebSphere MQ objects used by the JMS adapters. See “WebSphere MQ objects and settings” on page 628.
- The WebSphere Business Integration Server Express Plus sample solution for SAB404R. See “Testing our WebSphere Business Integration sample” on page 630.

Creating the legacy sample database - SAB404R

To install the working example successfully, you must begin by running the database creation script for the legacy sample database. Running the script will:

- Delete the database SAB404R if it already exists.
- Create the SAB404R database.
- Create the database tables.
- Populate the tables with the data used in the working example.

This script is located in folder c:\SAB404R\DDL.

**Important:** Check the database script file SAB404R.DDL. Using Notepad or a similar editor, open this file and change the user ID and password to one suitable for your installation if required. The scripts are set up to use db2admin with the default password defined during the installation of WebSphere Business Integration Server Express Plus.

This applies to all database scripts supplied with this sample.
To create and populate the database, open a DB2 command window by clicking **Start → Programs → IBM DB2 → Command Line Tools → Command Window**.

After the command window opens, type the commands as shown below.

```bash
cd \sab404r\ddl
db2 -td@ -vf sab404r.ddl
db2 -td@ -vf sal301r.ddl
```

The commands should process and create the databases SAB404R and SAL301R, with the required tables, as well as insert sample database records for each of the applications.

You can verify the contents of the newly created database SAB404R by clicking **Start → Programs → IBM DB2 → General Administrative Tools → Control Center** and when the program starts, expand the options until Databases are displayed. Expand SAB404R and click **Tables**. Right-click one of the SAB404R application tables such as Employee and click the **Sample Contents** menu option. This should display a dialog box with a grid containing data as shown in Figure A-1.

![Sample Contents - Employee](image)

**Figure A-1  Sample database contents - verified**

### Deploying the legacy agency system - SAB404R

The SAB404R Web application contains enough functionality to add new employees to the SAB404R Employee table. This application was built using WebSphere Application Server - Express.

The motive for providing this functionality for this application is to show synchronization of employees from the SAB404R application to the SAL301R application with WebSphere Business Integration Server Express Plus product.
The SAB404R system is packaged as an EAR file and must be loaded into the WebSphere Application Server.

This is a two-step process that involves loading the application into WebSphere Application Server and setting up the DataSource for the application to interact with the database.

**Loading the application into WebSphere Application Server**

Configuration of WebSphere Application Server is performed with WebSphere Application Server Administrative Console. This is a Web application that is started by clicking **Start → Programs → IBM WebSphere Application Server - Express 5.1 → Administrative Console (Local)**.

An initial logon window will appear. Put in an appropriate user ID or leave it blank and click **OK**.

The main console Web page will display. Click the plus (+) symbol beside to expand the Applications file in the panel to the left and display its options, one of which is Install New Application as shown in Figure A-17 on page 625.
Click **Install New Application**, which will display a Web page to perform the action of installing the new application.

Click **Browse** for the local path setting and using the choose file dialog, navigate to the location of where the package for this sample has been unzipped. Select **SAB404RAgencyEAR.ear** and click **Open**. The window should look like Figure A-3 on page 612.
Click **Next** to begin the installation process. On the next Web page, which is the generate default bindings page, leave all the defaults and click **Next**.

The next page is Step 1: Provide options to perform the installation. Leave all settings with the default values and click **Next**.

The next page is Step 2: Map virtual hosts for Web modules. Leave all settings with the default values and click **Next**.

The next page is Step 3: Map modules to application servers. Click the row with Server1, click the check box next to AgencyWeb, and click **Apply**. Your Web page should be similar to that illustrated in Figure A-4 on page 613, depending on what servers you have defined. Click **Next** to continue the process.
Figure A-4  Select server1 and the module AgencyWeb

The next page is a summary of the information gathered by the installation process. Click the **Finish** button.

When the processing is complete, if everything has been successful, then a Web page similar to Figure A-5 on page 614 will be displayed.
Click **Save to Master Configuration** to update the WebSphere Application Server. A confirmation Web page will appear. Click the Save link and the Save button in the Save to Master Configuration panel. After a short time, the default Administration Console Web page will display.

Installation of the SAB404R Web application is complete. Now the DataSources must be configured.

**Configuring the SAB404R DataSource**

There are a number of steps involved in configuring the DataSource. The first is to configure security for the DataSource, which is used later in setting up the JDBC providers. Then the configuration of the JDBC providers needs to be performed.

**JAAS configuration**

In the left hand panel of the Administrative Console Web page, click the plus (+) symbol next to **Security** to expand the security options. Then, expand JAAS
Configuration, and click the **J2C Authentication Data** link. The Web page illustrated in Figure A-6 will be displayed.

![J2C Authentication Data initial window](image.png)

**Figure A-6  J2C Authentication Data initial window**

Click **New** on the J2C Authentication Data Entries panel. The New J2C Authentication Data Entries Web page will display.

Enter values appropriate for the database configuration in use. We have been using the values from the default install:

- db2adminAlias
- db2admin
- password (substitute actual value)

After entering appropriate values, the window should resemble that in Figure A-7 on page 616. Click **OK** to confirm the settings and return to the J2C Authentication Data Entries Web page. Save the configuration at this point with the Save link and confirm with the Save button.
The JAAS configuration is complete. The JDBC provider information now needs to be completed.

**JDBC Provider configuration**

Click the plus (+) symbol beside Resources in the left panel of the Administration Console Web page and click the JDBC Providers link.

When the JDBC Providers Web page is displayed, make sure that an appropriate server is selected as shown in Figure A-8 on page 617, clicking Apply if a new value is selected. Then click New.
In the JDBC Providers New Web page, click the JDBC Providers list and select **DB2 Universal JDBC Driver Provider** as shown in Figure A-9 on page 618. Click **OK**. When the new Web page is displayed, click **Apply** to accept the rest of the defaults for the selected driver.
After **Apply** is clicked, the Web page will refresh, and at the bottom will be an Additional Properties table. The important option is the DataSources link as shown in Figure A-10 on page 619.
Click **Data Sources** to open the DataSources Web page as seen in Figure A-11 on page 620. From this page, you can create the DataSources used by SAB404R.
From the DataSource Web page, click the **New** button. When the DataSources New page is displayed, enter these values:

- **Name**: enter **AGENCY**
- **JNDI Name**: enter **jdbc/agency**
- **Component-managed Authentication Alias**: click **DefaultNode/db2AdminAlias** (the alias created earlier).
- **Container-managed Authentication Alias**: click **DefaultNode/db2AdminAlias**.

Click **Apply** to set the values. An Additional Properties frame will appear at the bottom of the Configuration frame. Click the **Custom Properties** link to set the specific database information. A Web page that resembles the one shown in Figure A-12 on page 621 should be displayed.
Figure A-12  Data Source Custom Properties

There are a number of values that need to be updated from this page. Click each of these settings and update with the supplied value, clicking **OK** on each page when finished:

- **databaseName** - SAB404R (as shown in figure Figure A-13 on page 622)
- **serverName** - localhost
There is one more value to be set. This is the value of the environment variable DB2UNIVERSAL_JDBC_DRIVER_PATH used by the JDBC Provider.

On the left pane, click **Environment** to expand the environment options, and then click the **Manage WebSphere Variables** link. If the Server field has a value, clear it and click **Apply**.

Find the DB2UNIVERSAL_JDBC_DRIVER_PATH link in the table and click it. Enter the path where the file db2jcc.jar is found. With the WebSphere Business Integration Server Express Plus installation, this will be C:\IBM\SQLLIB\java as shown in Figure A-14 on page 623.
Click the **Save** link to apply changes to the master configuration, and click the **Save** button as seen in Figure A-15.

Exit the WebSphere Application Server Administrative Console and restart the WebSphere Application Server to ensure that the changes have been applied.
SAB404R Web service sample requirements

As part of showing the capabilities of the WebSphere Business Integration Server Express Plus product, the Interchange Server will write to a Web service. The SAB404R application includes a Web service to simulate the external Newspaper interface. The sample Newspaper Web Service needs a log file to function. The code is expecting the file newsservice.log to exist in C:\SAB404R, which would have been created if the packaged solution was unzipped to C:\. If you extracted the packaged solution to another location, you must copy the sample file or create an empty file in C:\SAB404R.

Running and verifying the SAB404R application

To confirm that the SAB404R Web application is functional, assuming the WebSphere Application Server is installed locally on the default port settings, open this link:

http://localhost:7080/AgencyWeb/about_us.html

The Web page displayed will look like that illustrated in Figure A-16.

![Figure A-16   The SAB404R Web application- about page](image)
Click **Add Employee** to bring up the Employee data entry Web page. Add sample data as shown in Figure A-17 and click **Insert Employee**. Use a database tool to verify that the employee information is inserted into the database.

![Figure A-17](image)

**Creating the database triggers and event tables**

A key aspect of WebSphere Business Integration is the data triggers and tables used by the agents. The database updates required for these purposes are performed with different scripts. The scripts will:
- Drop the events and archive events tables if they exist
- Drop the triggers if they exist
- Create the events and archive events database tables
- Create the triggers required for each table and operation

The scripts for each database are located in folder c:\SAB404R\DDL. Check the SAB404R-Events.DLL and SAL301R-Events files. Using Notepad or a similar editor, open these files and change the user ID to one suitable for your installation if required.

To create the tables and triggers in the databases, open a DB2 command window by clicking Start → Programs → IBM DB2 → Command Line Tools → Command Window.

**Important:** Check the DDL files. Using notepad or a similar editor, open the files and change the user ID and password to one suitable for your installation if required. The scripts are set up to use db2admin with the default password defined during the installation of WebSphere Business Integration Server Express Plus.

After the command window opens, type the commands as shown in Example A-1.

**Example: A-1   Commands for processing and creating tables and triggers**

```
cd \sab404r\ddl
db2 -td@ -vf sab404r-events.ddl
db2 -td@ -vf sal301r-events.ddl
db2 -td@ -vf smb_user.ddl
```

The commands should process and create the tables and triggers into the SAB404R and SAL301R databases respectively.

Next, create and load the tables and other database objects needed by the relationships component using the commands in Example A-2.

**Example: A-2   Commands for loading the tables and other database objects**

```
db2 -td@ -vf SMB_User-Relationships.ddl
db2 -td@ -vf SMB_User-RelationshipsData.ddl
db2 -td@ -vf SMB_User-StoredProcs.ddl
```
Importing the relationships

Repos_copy is a command line interface for working with integration components and InterChange Server repositories. It allows you to deploy a collection of integration components to a server repository or to export components from the repository to a package.

Repos_copy was used to export the relationship components that must be imported to form part of the sample solution. Inside the directory c:\sab404r\Relationships there is a command file LoadRelationships.cmd that calls Repos_copy with the appropriate parameters. Open this file using notepad or another editor and verify that -u (user) and -p (password) are appropriate for your installation.

Important: Your Interchange Server instance must running before you run this command.

Open a normal command window: click Start → Run and in the Run dialog, type cmd and click OK.

In the command window that opens, type the following:

cd \sab404r\Relationships
LoadRelationships.cmd

The command should run and the output of the command window should resemble that shown in Figure A-18.

Reboot Interchange Server as requested by the batch file. In System Manager, in the Server Component window, click the active server instance and right-click Shut Down → Gracefully. The start the Interchange Server again by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Interchange Server Express → Interchange Server Express

![Figure A-18  Repos command output](image-url)
If you open System Manager, navigate to the Interchange Server Component pane, connect to the Interchange Server instance, and expand the tree to show Relationships and then Static, you will see the newly imported relationship. See Figure A-19 for an example.

![Figure A-19 Newly imported Static Relationship](image)

**Note:** To redo the load of the relationships command, you must stop the relationship and delete it. Then restart the server.

### WebSphere MQ objects and settings

All the connectors used in the SAB404R sample solution have been configured to use JMS as the transport. This means that WebSphere MQ objects need to be created for each connector instance.

First, make sure that WebSphere MQ has been started and that the queue manager is running. The queue manager installed as part of WebSphere Business Integration Server Express Plus is `WebSphereICS.queue.manager`
and is not configured to start automatically. To check the status of WebSphere MQ and start any components not running, click **Start → Programs → IBM WebSphere MQ → WebSphere MQ Services.**

When the WebSphere MQ MQServices console application starts, right-click **WebSphereICS.queue.manager** in window to the left, and click **All tasks → Start.** The process will take a short while. When it is complete, check the services listed in window to the right. If the status of any of these entities is not Running, right-click the entry, and select **All tasks → Start.** If the services still fail to start, consult an WebSphere MQ expert or the WebSphere MQ documentation.

In addition to starting the MQSeries queue manager, you may also be required to define a listener for the queue manager. There are two easy ways to define a listener:

1. Via the GUI by running MQSeries services - Click **Start → Programs → IBM WebSphere MQ → WebSphere MQ Services.** When the MQServices snap in starts, right-click the default queue manager and select **New → Listener.** The defaults are generally acceptable, so click **OK.**

2. Via the command line - Execute the following command from a command window:

   ```
   amqmdain crt1sr WebSphereICS.queue.manager -t tcp
   ```

Both of these create a listener with a TCP configuration that is listening on the default port 1414. The listener will be set to start automatically when the queue manager starts, but, at this time, it will be stopped. To start the Listener, right-click the Listener in the window to the right and select **Start.** Using the MQSeries services snap-in, you should be able to see the configuration illustrated in Figure A-20.

![Figure A-20](image)

*Figure A-20  WebSphere MQ services console with Listener installed and started.*
To create the WebSphere MQ queue objects, open a command prompt, click Start → Programs → Accessories → Command Prompt, and type these commands at the command prompt:

```
Example: A-3  WebSphere MQ queue objects commands

cd \sab404r\mq
runmqsc < AdapterMQObjects.txt
exit
```

This will create all objects that do not already exist, including the server connection channel and the eight objects potentially used by each instance of the adaptor.

**Tip:** While you have the MQSeries services window open, you can make each of the services start automatically after you reboot your machine. To do this, click the queue manager, then right-click Queue Manager, and click Properties. In the properties dialog, click the Startup list and change the startup type to automatic. Follow the same steps for Channel Initiator and Listener.

### Testing our WebSphere Business Integration sample

The focus of this book is providing developers with a working solution to a sample business problem using WebSphere Business Integration Server Express Plus. This working solution will be imported into System Manager using Import Solution. From System Manager, you will be able to view the solution while it is working and inspect the code and configurations of the Business Objects, collaborations, connectors, and other components.

There are a number of steps involved in starting and running the solution. First, the solution must be imported, then various components must be compiled, and finally, the components must be deployed to the server.

### Importing the solution

To import the solution, start System Manager by clicking Start → Programs → IBM WebSphere Business Integration Server Express → Toolset Express → Administrative → System Manager.

**Tip:** Details about starting System Manager and WebSphere Business Integration Server can be found in Chapter 11, “Administration tools” on page 379.
Appendix A. Deploying the redbook sample code

In the WebSphere Business Integration System Manager window, expand the User Projects folder and right-click **Interchange Server Projects** (as illustrated in Figure A-21), and click the **Import Solution** menu item.

![Figure A-21 Import System Solution Into System Manager](image)

From the Import Solution dialog box, click **Browse**, navigate to `c:\SAB404R\Solution`, and click **OK**. The Import Solution dialog with the suggested import part is shown in Figure A-22.

![Figure A-22 Import solution](image)
After the import process has completed, you should have a user project called SAB404RRealtyICS and an Interchange Component Library called SAB404RRealtyIL.

**Compiling components of the solution**

There are two groups of artifacts to compile:

1. Maps
2. Collaboration templates

Using System Manager, expand the user projects and the Interchange Server projects in the System Manager window and navigate to Maps. Expand Maps to expand to display the maps. Right-click each map that does not have the prefix sub_ and select **Compile with submap(s)** as shown in Figure A-23 on page 633.

The maps you should right-click are:

- EMailNotification_Property
- Employee_SAL301R_USR
- Property_EMailNotification
- Property_PublishingService
- SAB404R_EMPLOYEE_Employee
- SAB404R_Inventory_Property
- SAL301R_Property_Inventory
- SAL301R_PROP_Property
- SAL301R_USR_Employee
Appendix A. Deploying the redbook sample code

Figure A-23  Compiling the maps

Navigate the user project and expand **Collaboration Templates**. Right-click each collaboration in turn and click **compile** as shown in Figure A-24 on page 634. The templates that must be compiled are:

- EmployeeSync::SAB404RRealtyICL
- InventorySync::SAB404RRealtyICL
Deploying the solution to the server

Once the compilation of objects is complete, the solution is ready to be deployed to the Interchange Server. The server instance needs to be running for the deployment to succeed.

**Important:** The solution contains only objects that do not exist in the default installation of WebSphere Business Integration Server Express Plus. If you have developed any of the samples or are performing this action again, you must stop all executing components prior to the full deployment described in this section.

Review all collaboration objects, connectors, maps, and relationships to ensure that there are no running objects other than the default verification components.

We are going to deploy the entire project because it is new. During development you will probably only deploy changes.

Navigate to the user project, expand Interchange Server Projects, and click **SAB404RRealtyICS**. Right-click **SAB404RRealtyICS** and select Deploy user.
project. The Deploy wizard dialog will display. Select WebSphereICS from the list that appears. Expand the items available for SAB404RRealtyICS. Click the check box for SAB404RRealtyICS, which will select all the check boxes for the different component types. The Deploy dialog should now appear as illustrated in Figure A-25.

Whether the Next or Finish button is active will depend on whether any of the objects being deployed exist on the server.

- If the Next button is active, click it and perform the same selection to override everything just as you did in the Server Duplicate Objects dialog. Then click Finish.
- If the Finish button is active, click Finish.

![Figure A-25 Full deployment options](image)

The hour glass will be displayed for a period of time before an information dialog appears with the text:

2810 Reboot the server to activate the deployed components

This is illustrated in Figure A-26 on page 636.
To reboot the server as requested in the window, right-click the active server instance **WebSphereICS (Production)** in the InterChange Server Component Management window and click **Shut down → Gracefully**.

Start the Interchange Server again by clicking **Start → Programs → IBM WebSphere Business Integration Server Express → Interchange Server Express → Interchange Server Express**. Reconnect to the Interchange Server by right-clicking **WebSphereICS (Production)** in the InterChange Server Component Management window in System Manager. Then, click **Connect**. When the logon dialog appears, enter any necessary values and click **OK**.

The components deployed now must be started. Expand the component types listed below, select all instances and right-click **start all** if there is more than one option.

- Collaboration objects
- Connectors
- Maps
- Relationships → Static
- Relationships → Dynamic

If the component is started successfully, the icon should be updated to include a green triangle.

**Connector agents**

The solution in this book uses two connectors: the JDBC connector and a Web services connector. These configurations are available in the material supplied in the zipped file.

Copy the directories in the `c:\SAB44R\Connectors` directory to the `C:\IBM\WebSphereServer\connectors` directory. The target directory is where the other standard supplied connectors are installed.
The connectors included in the solution are:

- SAL301R - A JDBC adaptor
- SAB404R - A JDBC adaptor
- PropertyPublishing - A Web service adaptor
- E-mail - An e-mail adaptor

**Important:** In addition to these directories, the file EMailConnector.cfg needs to be copied from the e-mail directory to the base directory for the Interchange Server; by default, this is C:\IBM\WebSphereServer.

We are not sure why this is required. The other connectors are fine with the config file in the connector specific directory.

Each connector is supplied in its own folder and independent of any others. The connector can be started using Windows Explorer. Navigate to the directory where the connector has been copied, locate the short cut, and double-click the file. The icon for the short cut shows a plug.

In addition to the connector-specific directories, there is a directory that contains the message files for these four connectors. Ensure that the `c:\SAB44R\Connectors\messages` directory is also copied to the `C:\IBM\WebSphereServer\connectors` directory.

**Note:** The instructions above and the batch files that start the connector agents assume the default installed directories. You must modify the batch files that start the agents with other directories if different paths are used.

**The moment of truth - verification**

The final task is to verify end-to-end functionality. If the solution meets the business goals, then a user should be able to verify this functionality with the existing systems.

To perform a verification test, we will create a user in the SAB404R system. The user will be synchronized from the SALB404R system, and the WebSphere Business Integration employeeSync collaboration will perform the synchronization. We will then be able to log on to the SAL301R system using the newly created user.

Before running the test, use the check list supplied in Table A-3 on page 638 to ensure that all components are running.
### Table A-3  Pre-test Checklist

<table>
<thead>
<tr>
<th>Component</th>
<th>Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Application Server - server 1 instance to service the SAL301R and SAB404R Web applications</td>
<td></td>
</tr>
<tr>
<td>WebSphere Business Integration Server Express - InterChange Server</td>
<td></td>
</tr>
<tr>
<td>All components deployed to the InterChange server</td>
<td></td>
</tr>
<tr>
<td>MQSeries - Queue manager, Listener and Channel Initiator</td>
<td></td>
</tr>
<tr>
<td>Connector Agents:</td>
<td></td>
</tr>
<tr>
<td>SAL301R</td>
<td></td>
</tr>
<tr>
<td>SAB404R</td>
<td></td>
</tr>
<tr>
<td>PropertyPublishing</td>
<td></td>
</tr>
<tr>
<td>E-mail</td>
<td></td>
</tr>
</tbody>
</table>

Assuming that the applications are installed on the local system with WebSphere Application Server configured to use port 7080, start the SAB404R application using this URL:

http://localhost:7080/AgencyWeb/faces/employee/add_employee.jsp

Enter some values for the new values such as those shown in Figure A-27 on page 639. Click the **Insert Employee** button.
Start the SAL301R application using this URL:

http://localhost:7080/SAL301RRealtyWeb/index.jsp

When the SAL301R Realty home page appears, click the Log in option. Enter the values for the dialog as follows:

- User ID = [first letter of first name][surname] all in lowercase - in our example the value to use would be fnerk for Frederick Nerk
- password=password (default)

See Figure A-28 on page 640 for the sample login.
If everything is running, then you will see a Web page like that illustrated in Figure A-29 on page 641. The new employee in SAB404R has been replicated to SAL301R automatically. Verify that the attributes of the employee such as address have also been correctly propagated in the database.
Appendix A. Deploying the redbook sample code

There is no user interface in the SAB404R application for viewing properties, but if you perform a property create in SAL301R, you should find the property in the INVENTORY table in the SAB404R database.

Tip: You can see the statistics for the connectors and collaborations using the techniques discussed in 11.6, “Using statistics for system interrogation” on page 413 in either System Manager or the ICSMonitor.

For the employee synchronization performed in the test above, you should see a successful send for connector SAB404R and a receive for connector SAL301R, with the collaboration statistics showing 1 for total events and 1 for successful.
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/SG246353

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, SG246353.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG246353.zip</td>
<td>Zipped Code Samples</td>
</tr>
</tbody>
</table>
System requirements for downloading the Web material

The following system configuration is recommended:

*Hard disk space:* 25 MB minimum  
*Operating System:* Windows 2000 Server  
*Processor:* Pentium® III 1GHz or higher  
*Memory:* 512 MB or higher

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

For more details of the sample code and instructions on how to install and work with our redbook samples refer to Appendix A, “Deploying the redbook sample code” on page 603.
### Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADK</td>
<td>Adapter Development Kit</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>ASBO</td>
<td>Application Specific Business Object</td>
</tr>
<tr>
<td>B2B</td>
<td>business-to-business</td>
</tr>
<tr>
<td>B2C</td>
<td>business-to-consumer</td>
</tr>
<tr>
<td>BOM</td>
<td>bill of materials</td>
</tr>
<tr>
<td>BPEL</td>
<td>business process execution language</td>
</tr>
<tr>
<td>CDK</td>
<td>Connector Development Kit</td>
</tr>
<tr>
<td>COM</td>
<td>Common Object Model</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer Relationship Management</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions System</td>
</tr>
<tr>
<td>DTD</td>
<td>document type definition</td>
</tr>
<tr>
<td>EAI</td>
<td>Enterprise Application Integration</td>
</tr>
<tr>
<td>EAR</td>
<td>Enterprise Archive</td>
</tr>
<tr>
<td>EDI</td>
<td>electronic data interchange</td>
</tr>
<tr>
<td>EIS</td>
<td>Enterprise Information System</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise Java Bean</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>GBO</td>
<td>Generic Business Object</td>
</tr>
<tr>
<td>HR</td>
<td>Human Resources</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>ICL</td>
<td>integration component library</td>
</tr>
<tr>
<td>IDL</td>
<td>Interface Definition Language</td>
</tr>
<tr>
<td>IIOP</td>
<td>Internet Inter-ORB Protocol</td>
</tr>
<tr>
<td>ITSO</td>
<td>International Technical Support Organization</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Platform, Enterprise Edition</td>
</tr>
<tr>
<td>J2SE</td>
<td>Java 2 Platform, Standard Edition</td>
</tr>
<tr>
<td>JCA</td>
<td>J2EE Connector Architecture</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>JSP</td>
<td>JavaServer Pages</td>
</tr>
<tr>
<td>MIME</td>
<td>Multipurpose Internet Mail Extensions</td>
</tr>
<tr>
<td>MOM</td>
<td>message-oriented middleware</td>
</tr>
<tr>
<td>MVC</td>
<td>model-view-controller</td>
</tr>
<tr>
<td>ODA</td>
<td>Object Discovery Agent</td>
</tr>
<tr>
<td>ODK</td>
<td>Object Discovery Agent Development Kit</td>
</tr>
<tr>
<td>SAI</td>
<td>Server Access Interface</td>
</tr>
<tr>
<td>SCM</td>
<td>Supply Chain Management</td>
</tr>
<tr>
<td>SDK</td>
<td>Java 2 Software Development Kit, Standard Edition</td>
</tr>
<tr>
<td>SMB</td>
<td>small and medium business</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple mail transfer protocol</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SWIFT</td>
<td>Society for the Worldwide Interbank Financial Telecommunication</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</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>
XSD  XML Schema Definition
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 on ordering these publications, see “How to get IBM Redbooks” on page 650. Note that some of the documents referenced here may be available in softcopy only.

- Patterns: Implementing an SOA using an Enterprise Service Bus, SG24-6346
- Patterns: Broker Interactions for Intra- and Inter-enterprise, SG24-6075
- Patterns: Serial and Parallel Processes for Process Choreography and Workflow, SG24-6306
- WebSphere Application Server - Express: A Development Example for New Developers, SG24-6301

Online resources

These Web sites and URLs are also relevant as further information sources:

- SOA and Web services
  http://www.ibm.com/developerworks/webservices
- Understand Enterprise Service Bus scenarios and solutions in Service-Oriented Architecture, Part 1
- Understand Enterprise Service Bus scenarios and solutions in Service-Oriented Architecture, Part 2
- Understand Enterprise Service Bus scenarios and solutions in Service-Oriented Architecture, Part 3
- IBM Patterns for e-business
  http://ibm.com/developerworks/patterns
WebSphere Express software

WebSphere Portal - Express product overview
http://ibm.com/software/genservers/portalexpress/

WebSphere Business Integration Connect - Express product overview

WebSphere Application Server - Express product overview
http://ibm.com/software/webservers/appserv/express/

WebSphere Commerce - Express product overview
http://ibm.com/software/genservers/commerce/express/

WebSphere MQ Express product overview
http://ibm.com/software/integration/wmq/express/

WebSphere Business Integration Server Express product overview
http://ibm.com/software/integration/wbiserverexpress/

WebSphere Business Integration Adapter for e-mail
http://ibm.com/software/integration/wbiadapters/email/

WebSphere Business Integration Adapter for JDBC

WebSphere Business Integration Adapter for JMS
http://ibm.com/software/integration/wbiadapters/jms/

WebSphere Business Integration Adapter for WebSphere MQ
http://ibm.com/software/integration/wbiadapters/mq/

WebSphere Business Integration Adapter for XML
http://ibm.com/software/integration/wbiadapters/xml/

WebSphere Business Integration Adapter for Web services
http://ibm.com/software/integration/wbiadapters/services/

COM adapter

SWIFT adapter
http://ibm.com/software/integration/wbiadapters/swift/

iSeries adapter
http://ibm.com/software/integration/wbiadapters/iseries/
- Microsoft Exchange adapter
- Portal Infranet adapter
- Lotus Domino adapter
- i2 adapter
- JD Edwards OneWorld adapter
- MetaSolv adapter
- mySAP.com adapter
- Oracle applications adapter
- PeopleSoft adapter
- Siebel e-business applications adapter
- Telcordia adapter
- WebSphere Commerce adapter
- WebSphere Business Integration Information Center
- CVS home
  [http://www.cvshome.org](http://www.cvshome.org)
- Eclipse home
  [http://www.eclipse.org](http://www.eclipse.org)
- Compiling DB2 stored procedures using GCC
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:

http://www.ibm.com/redbooks

Help from IBM

IBM Support and downloads

http://www.ibm.com/support

IBM Global Services

http://www.ibm.com/services
Index

A
abstraction 40, 160
Access Integration pattern 30
access request 315
access requests 316
accessibility 426
action nodes 336
Activity Decision Flow Diagram 436
activity diagram 137–138, 140, 331, 336
Activity Editor 12, 56, 249, 268, 280
Adapter Agent 52, 54, 161–162, 166, 490, 493, 496, 498
configuration 194
Adapter capacity pack 44, 159, 170
adapter controller 51–52
Adapter Development Kit See, ADK
Adapter Framework 161–162
adapters 9, 24–27, 45, 47–48, 51–52, 57, 75, 159, 161, 211, 448
  application 168, 170
  COM 169
  configuration 101, 121, 181, 477, 569, 586
  create 477
  deploy 198
  e-mail 168, 451
  event notification 175
  i2 170
  implementation 172
  installation 172
  iSeries 170
  JDBC 76, 168, 451, 488
  Lotus Domino 170
  MetaSolv 171
  Microsoft Exchange 170
  mySAP.com 171
  OneWorld 171
  Oracle application 171
  PeopleSoft 171
  portal infranet 170
  Siebel 172
  specific properties 186, 487, 491
  standard properties 183
  supported Business Objects 192
  supported maps 193
  SWIFT 169
technology 168, 502
  Telcordia 172
  trigger 178
  uninstall 175
  Web services 169, 451
  WebSphere Commerce 172
  WebSphere MQ 169
  XML 169
adapters See also, connectors
addresses 547
ADK 206
administration tools 379
agents 51, 101, 503, 506
agility 40
AIX 36
analyze 47, 435
API 208, 210
application adapters 168, 170
application integration 136
Application Integration pattern 31
Application Specific Business Object See, ASBO
applications 24
architecture 4–6, 19, 48, 160, 427, 449
  integration 22
archive 592
ASBO 10, 12, 48, 53, 55, 101, 103, 107, 120, 162, 165, 188, 210, 218, 310, 502
  create 188
  SAL301R 506
asynchronous coupling 29
asynchronous request 316
attributes 103–104, 106, 113, 134, 435
  join 261
  splitting 263
automation 449
availability 426

B
B2B 30–31, 35
best practices 19
bindings 311
<table>
<thead>
<tr>
<th>BO See, Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOM 310</td>
</tr>
<tr>
<td>BPEL 53, 327</td>
</tr>
<tr>
<td>breakpoints 119</td>
</tr>
<tr>
<td>broker 19, 25, 160</td>
</tr>
<tr>
<td>browser 56, 69</td>
</tr>
<tr>
<td>bus 26</td>
</tr>
<tr>
<td>bus topology 26</td>
</tr>
<tr>
<td>business modeling 434</td>
</tr>
<tr>
<td>Business Object 7–11, 13, 47–48, 51–53, 56, 76, 100–101, 103, 120, 164, 166, 209, 499, 545, 548, 585</td>
</tr>
<tr>
<td>application-specific information 244</td>
</tr>
<tr>
<td>attributes 10, 212, 214</td>
</tr>
<tr>
<td>create 84, 103, 188</td>
</tr>
<tr>
<td>definition 104, 210–211</td>
</tr>
<tr>
<td>deploy 240</td>
</tr>
<tr>
<td>design 221, 450</td>
</tr>
<tr>
<td>header 10, 212–213</td>
</tr>
<tr>
<td>hierarchical 10</td>
</tr>
<tr>
<td>instances 218</td>
</tr>
<tr>
<td>samples 225, 450, 454</td>
</tr>
<tr>
<td>source 261</td>
</tr>
<tr>
<td>standards 222</td>
</tr>
<tr>
<td>testing 510</td>
</tr>
<tr>
<td>verbs 10, 105, 212, 217</td>
</tr>
<tr>
<td>Business Object Designer 8, 47, 56, 84, 103</td>
</tr>
<tr>
<td>Business Object Designer Express 11, 101, 103, 210, 223</td>
</tr>
<tr>
<td>Business Object probes 338</td>
</tr>
<tr>
<td>business partners 4</td>
</tr>
<tr>
<td>business patterns 30</td>
</tr>
<tr>
<td>business problem 427</td>
</tr>
<tr>
<td>Business-to-Business See B2B</td>
</tr>
<tr>
<td>buyers 470</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>calling context 117</td>
</tr>
<tr>
<td>cardinality 104</td>
</tr>
<tr>
<td>cases 437</td>
</tr>
<tr>
<td>CICS 20</td>
</tr>
<tr>
<td>Client Simulator 154</td>
</tr>
<tr>
<td>CN 245</td>
</tr>
<tr>
<td>Collaboration capacity pack 13, 45, 309, 311, 318, 321</td>
</tr>
<tr>
<td>Customer Relationship Management 321</td>
</tr>
<tr>
<td>Financial and Human Resources 321</td>
</tr>
<tr>
<td>installation 321</td>
</tr>
<tr>
<td>Order Management 321</td>
</tr>
<tr>
<td>Procurement 321</td>
</tr>
<tr>
<td>Collaboration Debugger 56</td>
</tr>
<tr>
<td>collaboration groups 13</td>
</tr>
<tr>
<td>collaboration object 13, 193, 321, 334, 586</td>
</tr>
<tr>
<td>collaboration template 137</td>
</tr>
<tr>
<td>CollaborationFoundation 312, 318</td>
</tr>
<tr>
<td>collaborations 9, 13, 45, 51, 57, 95, 102, 136, 164, 309–310, 320, 499, 543, 573, 582, 584, 641</td>
</tr>
<tr>
<td>access requests 316</td>
</tr>
<tr>
<td>CollaborationFoundation 318</td>
</tr>
<tr>
<td>compile 328, 333</td>
</tr>
<tr>
<td>components 311</td>
</tr>
<tr>
<td>configuration 311</td>
</tr>
<tr>
<td>debugging 380, 399</td>
</tr>
<tr>
<td>deployment 328, 335, 544</td>
</tr>
<tr>
<td>design 323, 450</td>
</tr>
<tr>
<td>development 326, 328</td>
</tr>
<tr>
<td>EmployeeSync 542</td>
</tr>
<tr>
<td>event 315</td>
</tr>
<tr>
<td>exceptions 346</td>
</tr>
<tr>
<td>export 327</td>
</tr>
<tr>
<td>groups 311–312</td>
</tr>
<tr>
<td>import 327</td>
</tr>
<tr>
<td>logging 346</td>
</tr>
<tr>
<td>message file 332</td>
</tr>
<tr>
<td>objects 311, 321, 334</td>
</tr>
<tr>
<td>ports 311, 314</td>
</tr>
<tr>
<td>process type 318</td>
</tr>
<tr>
<td>processing 315</td>
</tr>
<tr>
<td>samples 348, 450, 464, 542</td>
</tr>
<tr>
<td>scenarios 311–312, 331</td>
</tr>
<tr>
<td>Server Access Interface 582</td>
</tr>
<tr>
<td>service call 316</td>
</tr>
<tr>
<td>standards 324</td>
</tr>
<tr>
<td>statistics 417</td>
</tr>
<tr>
<td>subdiagrams 312</td>
</tr>
<tr>
<td>subscription 315</td>
</tr>
<tr>
<td>synchronization 317</td>
</tr>
<tr>
<td>templates 13, 311, 314, 318</td>
</tr>
<tr>
<td>testing 373, 545</td>
</tr>
<tr>
<td>tracing 388</td>
</tr>
<tr>
<td>trigger 315</td>
</tr>
<tr>
<td>triggers 13</td>
</tr>
<tr>
<td>types 317</td>
</tr>
<tr>
<td>wrapper 317</td>
</tr>
<tr>
<td>WrapperFoundation 319</td>
</tr>
<tr>
<td>COM 502</td>
</tr>
<tr>
<td>COM adapter 169</td>
</tr>
</tbody>
</table>
compensation 26, 316
compile 143, 328, 632
  maps 632
  templates 632
compiler 60, 297, 606
Component Management 420
components 311
  start 420
  stop 420
conditions 267
configuration 311
  adapters 477, 569, 586
  connectors 586
connect 427
connection pool 273
connectivity 448
Connector Configurator Express 8, 101, 119, 192
connector controller 161–162, 165–166, 193
Connector Development Kit See, CDK
connector framework 9, 119
connectors 7–9, 76, 310–311, 316, 367, 413, 509, 543, 573, 636, 641
  configuration 586
  create 81
JDBC 76, 81, 592, 636
JMS 595
  statistics 413
  test 8, 90
  tracing 390
Web services 636
connectors See also, adapters
ContactWrapper 320
CORBA 51–52, 168, 582
coupling 29
  asynchronous 29
  synchronous 29
create
  Business Object 103
  ICL 472
  project 472
CRM 21, 53, 310, 321
Cross-reference 114, 249, 279
Custom 114, 249
Customer Relationship Management See, CRM
CustomerPartnerWrapper 320
CustomerSynchronization 320
CustomerWrapper 320
CVS 76, 95, 598
data enrichment 26
data handlers 10, 203, 205
  configuration 206
data synchronization 101
data transformation 23
database 5, 7, 20, 50, 52, 71, 84, 120, 273, 426, 592, 625
  create 608
  SAB404R 608
DataHandler class 205
DataSource 614
DB2 95, 502
  installation 71
DB2 UDB Express 49
DDL 7, 604, 608, 626
debugging 118
  collaborations 380, 399
decision nodes 339
definitions 138
delimiters 261
deployment 10, 19, 512
  collaborations 328, 335, 544
  relationships 297, 572
  SAB404R 609
  samples 371, 603, 634
design 5, 10, 19, 26, 451
  Business Object 221, 450
  collaborations 323, 450
  samples 448, 451
design by contract 41
design tools 56
developers 4
development 4, 16, 102
  collaborations 326, 328
e-commerce 22
e-commerce 22
e-commerce 22
e-commerce 22
e-commerce 22
e-commerce 22
e-commerce 22
e-commerce 22
e-commerce 22

diversity 42
DTD 234
dynamic relationship 279

E
EAI 4, 6, 19
Eclipse 7, 9, 100
e-commerce 22
EDI 20
EIS 162
EJB 583
E-mail 470
   Adapter Agent 498
e-mail 9, 558
E-mail adapter 168
e-mail adapter 451
EMailConnector 477, 497
   properties 497
Employee GBOs 500
employee synchronization 499
EmployeeSync 542
encapsulation 39
end nodes 345
Enterprise Application Integration See, EAI
Enterprise JavaBeans See, EJB
Enterprise Resource Planning See, ERP
ERP 21, 53, 310, 321
errors 582
escrow 443
event 50, 80, 136, 139, 163, 166, 312, 314–315, 320, 331, 406, 449, 592, 625
   subscription 202
event archives 166
event notification 164, 175, 491, 493
event table 164
exceptions 346
executable 311
execution 315
export 327
Extended Enterprise business pattern 30

F
Failed Event Manager 69, 379, 406
   actions 412
   start 385
fail-over 583
FK 245
flexibility 448
Flow manager 56, 100, 379, 406
   actions 412
   start 382, 406
foreign key 104, 245
foundation collaborations 57
foundation templates 318
functionality 42

G
GBO 10, 40, 48, 53–55, 87, 103, 107–108, 162, 210, 218, 310, 500, 549
   create 192
   Employee 500
GCC 297
Generic Business Object See, GBO
groups 311–312

H
hardware requirements 60
high availability 26
HR 53
hub 25–26, 53, 160
hub and spoke 25
topology 23

I
i2 adapter 170
ICL 7, 75–76, 81, 143, 181, 225, 240, 371, 501
   create 472
ICS 119, 121, 136
ICSMonitor 384, 415, 419, 421, 641
   start 384
identity relationships 279, 293
IDL 582
IIOP 582
implementation 19
import 327
IMS 20
Information Integration pattern 31
installation 5, 59
   adapters 172
   Collaboration capacity pack 321
   DB2 71
   Internet Explorer 69
   JDK 66
   Netscape Navigator 70
   SAB404RAgencyEAR 611
   WebSphere Application Server - Express 69
   WebSphere Business Integration Server Express 73
   WebSphere Business Integration Server Express Plus 6
   WebSphere MQ 67
integrate 427
Integrated Test Environment 148, 546
Integrated Test Environment Console
Integrated Test Environment Navigator 153
integration 4–5, 19, 22, 426
architecture 22
topologies 23
integration component library See, ICL
integration patterns 30
Interchange Server 241, 316, 450, 624
register instance 474
Interchange Server Express 8, 46, 50, 148, 449, 582
start 380
Interface Definition Language 582
interfaces 25
Internet Explorer
installation 69
Internet Inter-ORB Protocol See, IIOP
interoperability 426
iSeries 20, 36
iSeries adapter 170
isolation 39
ItemWrapper 320
iterators 343

J
J2EE 50, 583
J2SE 50
JAAS 614
Java 108, 136, 249, 333, 582, 590
Java 2 Platform, Enterprise Edition See, J2EE
Java development kit See, JDK
Java Servlets 36, 69
JavaServer Pages See, JSP
JD Edwards OneWorld adapter 171
JDBC 10, 75, 103, 119, 228, 502, 592, 614, 636
connectors 592
ODA 188
JDBC adapter 76, 168, 175, 451, 488
JDBC connector 76, 81
JDBC ODA 503, 506
JDBC Providers 616
JDBCODA 11, 228, 244, 503, 506, 551
JDK 50
installation 66
JMS 7, 9, 48, 468, 470, 593, 595, 628
connectors 595
JMS adapter 169
Join 113, 249
JRE 50
JSP 36, 591

K
KEEP_RELATIONSHIP 246
key 104

L
Launchpad 6, 63
legacy 5
legacy system 53
Linux 36, 62
listener 68
load-balancing 583
Log Viewer 100, 379, 394, 396
logging 124, 346, 380, 386
configuration
configuration
logging 386
logic 23–25, 27, 136, 267, 310–311, 313, 331
lookup 12
lookup relationships 280
Lotus Domino adapter 170

M
maintainability 40
manage 48, 427
management 24
map 193
Map Designer 47, 56, 102
Map Designer Express 8, 12, 87, 101, 107–108, 248, 253, 267
mapping 11, 51, 53, 55, 102, 167, 247, 310, 450
development 248
relationships 279
maps 8, 47, 75–76, 107–108, 110, 126, 162, 167, 193, 499, 545, 573, 585
Activity Editor 268
compile 632
create 87, 253
create join 261
Cross-reference 249, 279
Custom 249
debugging 118
development 248
document 282
Join 249

Index 655
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>249</td>
</tr>
<tr>
<td>property synchronization</td>
<td>559</td>
</tr>
<tr>
<td>relationships</td>
<td>279</td>
</tr>
<tr>
<td>reverse</td>
<td>261</td>
</tr>
<tr>
<td>samples</td>
<td>523</td>
</tr>
<tr>
<td>Set Value</td>
<td>249</td>
</tr>
<tr>
<td>Split</td>
<td>249</td>
</tr>
<tr>
<td>Submap</td>
<td>249</td>
</tr>
<tr>
<td>testing</td>
<td>257, 538</td>
</tr>
<tr>
<td>validate</td>
<td>118</td>
</tr>
<tr>
<td>Web services</td>
<td>275</td>
</tr>
<tr>
<td>message file</td>
<td>332</td>
</tr>
<tr>
<td>messages</td>
<td>138, 597</td>
</tr>
<tr>
<td>meta-data</td>
<td>119, 205</td>
</tr>
<tr>
<td>meta-objects</td>
<td>206</td>
</tr>
<tr>
<td>MetaSolv adapter</td>
<td>171</td>
</tr>
<tr>
<td>methodology</td>
<td>13, 221, 323</td>
</tr>
<tr>
<td>development</td>
<td>450</td>
</tr>
<tr>
<td>Microsoft Exchange adapter</td>
<td>170</td>
</tr>
<tr>
<td>middleware</td>
<td>48, 426, 448</td>
</tr>
<tr>
<td>MIME</td>
<td>205–206</td>
</tr>
<tr>
<td>mixed topology</td>
<td>27</td>
</tr>
<tr>
<td>MO_DataHandler_Default</td>
<td>206</td>
</tr>
<tr>
<td>model</td>
<td>427, 434–435, 438</td>
</tr>
<tr>
<td>MOM</td>
<td>168</td>
</tr>
<tr>
<td>monitor</td>
<td>48, 427, 435</td>
</tr>
<tr>
<td>Move</td>
<td>114, 249</td>
</tr>
<tr>
<td>MQSeries Workflow</td>
<td>435</td>
</tr>
<tr>
<td>mySAP.com adapter</td>
<td>171</td>
</tr>
</tbody>
</table>

N
naming 11
Netscape Navigator
  installation 70
nodes 336
  action 336
  decision 339
  end 345
  start 336

O
Object Discovery Agent Development Kit See, ODK
Object Discovery Agent See, ODA
Object Request Broker 50
objects 311
ODA 8, 11, 54, 103, 188, 206–207, 210, 228, 502, 551, 558
  JDBC 188, 228, 503, 506
XML 234, 558
ODK 210
  API 208
operational considerations 23
Oracle applications adapter 171
organization 435
organization data 435
OS/390 53
OS/400 61

P
participant type 132
participants 9, 132, 286, 288
paths 437
patterns 5, 29
  Application Integration 31
  business 30
  Information integration 31
  integration 30
Patterns for e-business 30
PeopleSoft adapter 171
perspective 76
point-to-point 19, 24
  topology 23
Portal Infranet 502
Portal Infranet adapter 170
ports 13, 138–139, 311, 314, 316, 320, 331, 543
prerequisites 603–604
probes 338
process collaborations 318
process data 435
Process Designer 56, 102
Process Designer Express 9, 44, 320, 326
Process Integration 30
process metrics 435
process modeling 430
processes 13, 26, 47, 50–51, 310–311, 318, 427, 430
  long lived 317
  model 438
product mapping 32
project
  create 472
projects 7, 131
properties 487, 491, 494
property synchronization 548
PropertyPublishing 554
  Adapter Agent 496

656 WebSphere Business Integration Server Express The Express Route to Business Integration
PropertyPublishingConnector 477
   properties 494
PropertyPublishingConnector 494
protocol 24
publish 315
publish-and-subscribe 315

Q
queue 593, 597

R
Redbooks Web site 650
   Contact us xvi
redundancy 26
Relationship Designer 56
Relationship Designer Express 9, 101, 130
Relationship Manager 56, 100, 305
relationships 12, 130, 247, 279, 286, 477, 499, 517, 570, 627
   attributes 134
   create 131
   deployment 297, 572
   development 288
dynamic 279, 517
identity 249, 279, 293
lookup 280, 290
management 303
participants 132
static 279, 519
testing 298
reliability 426
repository 52, 435
requirements
   samples 425
reuse 40
reverse
   maps 261
RFID 22
routing 24–25, 27
rules 111, 136, 275
runmqsc 593
runtime mapping 32

S
SAB404R
   DataSource 614
   WebSphere MQ settings 628
SAB404R Adapter Agent 493
SAB404RAgencyEAR 604
SAB404RConnector 477, 509, 542
   properties 491
SAB404RRealtyICL 501, 517, 519
SAB404RRealtyICS 634
SAI 582
SAL301 76
SAL301R Adapter Agent 490
SAL301R ASBOs 506
SAL301RConnector 477, 542
   properties 486
samples 4, 16, 75, 225, 348, 451
   Business Object 450, 454
   business problem 427
collaborations 450, 464, 542
deployment 371, 603, 634
design 448
development 450, 469
maps 523
objects 429
prerequisites 604
requirements 425
SAB404R 609
Server Access Interface 583
solution 430
testing 373, 630
Web services 624
Scalability 25
scalability 23–24, 26–27, 45
scenarios 13, 138, 140, 225, 311–312, 331
   Main 312
   Retrieve 312
SCM 21, 310
SCN 244
SDK 50
security 27, 614
sellers 470
Server Access Interface 310, 316, 582
   client 588
collaborations 582
   overview 582
   samples 583
service call 315–316, 340
service call request 316
service call response 316
Service-Oriented Architecture See, SOA
Set Value 114, 249
Siebel e-business applications adapter 172
simulate 48, 435
simulator 154
SiteWrapper 320
SMB 4, 37, 427, 448
SMTP 60, 607
SOA 5
SOAP 169
software requirements 60
specification 15
Split 114, 249
splitting attributes 263
spoke 160
SQL 304
SQL Server 71
standards 13, 222
  collaborations 324
start nodes 336
static relationship 279
statistics 380, 413
  collaborations 417
  connectors 413
stored procedures 60, 288, 477, 606
subdiagrams 312, 342
Submap 114, 249
  conditions 267
subscribe 315
subscription 10, 50, 164, 202, 315
subscription list 164
supply chain 22
Supply Change Management See, SCM
SWIFT adapter 169
synchronization 17, 317, 641
synchronous coupling 29
synchronous request 316
System Manager 7, 9, 15, 56, 76, 100, 108, 131,
143, 338, 379, 388, 413, 417, 420, 471, 598, 630, 641
  start 381
System Monitor 14, 56, 69, 148, 379
SystemTest 314, 320

CollaborationFoundation 318
compile 328, 333, 632
ContactWrapper 320
create 137, 328
CustomerpartnerWrapper 320
CustomerSynchronization 320
CustomerWrapper 320
foundation 318
ItemWrapper 320
SiteWrapper 320
SystemTest 320
testing 320
VendorPartnerWrapper 320
WrapperFoundation 319
Test Connector 8–9, 101, 126, 373, 392
Test Environment 56, 148
Test Unit Editor 156
testing
  collaborations 373, 545
  connectors 90
  maps 257, 538
  property synchronization 579
  relationships 298
  SAB404R 624
  SAL301R 510
  samples 373, 630
  templates 320
TN 244
Tomcat 60, 69
tools 8, 406
  administration 379
Toolset Express 8, 44, 47–48, 55, 99, 379
topology 5
  bus 26
  hub and spoke 23, 160
  mixed 27
  point-to-point 23
tracing 124, 380, 386, 393
  collaborations 388
  configuration 386
  connectors 390
transaction control 26
transactions 583
transformation 24–25, 102, 107, 110, 264
  Cross-reference 249
  Custom 249
  Join 249
  join 261
  Move 249

T
  tables 592
Task Manager 154
technology adapters 168, 502
Telcordia adapter 172
templates 47, 51, 57, 211, 311, 314, 318, 320, 448,
  511, 543, 573
<table>
<thead>
<tr>
<th>Page</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>240</td>
<td>Set Value 249</td>
</tr>
<tr>
<td>241</td>
<td>Split 249</td>
</tr>
<tr>
<td>242</td>
<td>Submap 249</td>
</tr>
<tr>
<td>250</td>
<td>transformations 12, 25–26, 249, 282</td>
</tr>
<tr>
<td></td>
<td>Custom 268</td>
</tr>
<tr>
<td></td>
<td>Submap 265</td>
</tr>
<tr>
<td>338</td>
<td>transition links 338</td>
</tr>
<tr>
<td>280</td>
<td>translation 280</td>
</tr>
<tr>
<td>166–167</td>
<td>transport layer 166–167</td>
</tr>
<tr>
<td>54, 136, 139, 163, 178, 313, 315, 320, 331, 573, 592, 625</td>
<td>trigger 54, 136, 139, 163, 178, 313, 315, 320, 331, 573, 592, 625</td>
</tr>
<tr>
<td></td>
<td>collaborations 13</td>
</tr>
<tr>
<td>139</td>
<td>triggering events 139</td>
</tr>
<tr>
<td>245</td>
<td>UID 245</td>
</tr>
<tr>
<td>136, 140, 327, 435, 438</td>
<td>UML 136, 140, 327, 435, 438</td>
</tr>
<tr>
<td>240, 472</td>
<td>user project 240, 472</td>
</tr>
<tr>
<td>31</td>
<td>User-to-Business 31</td>
</tr>
<tr>
<td>48</td>
<td>validate 48</td>
</tr>
<tr>
<td>118</td>
<td>validate map 118</td>
</tr>
<tr>
<td>320</td>
<td>VendorPartnerWrapper 320</td>
</tr>
<tr>
<td></td>
<td>Create 105</td>
</tr>
<tr>
<td></td>
<td>Delete 105</td>
</tr>
<tr>
<td></td>
<td>Retrieve 105</td>
</tr>
<tr>
<td></td>
<td>Update 105</td>
</tr>
<tr>
<td>596</td>
<td>verification 596</td>
</tr>
<tr>
<td>435, 599</td>
<td>version control 435, 599</td>
</tr>
<tr>
<td>382</td>
<td>Web server 382</td>
</tr>
<tr>
<td>9, 12, 28, 100, 275, 470, 502, 636</td>
<td>Web services 9, 12, 28, 100, 275, 470, 502, 636</td>
</tr>
<tr>
<td>624</td>
<td>SAB404R 624</td>
</tr>
<tr>
<td>169, 451</td>
<td>Web services adapter 169, 451</td>
</tr>
<tr>
<td></td>
<td>Web Services Description Language, See WSDL</td>
</tr>
<tr>
<td>6</td>
<td>WebSphere 6</td>
</tr>
<tr>
<td>53, 610</td>
<td>WebSphere Application Server 53, 610</td>
</tr>
<tr>
<td>7, 33, 36, 49, 60, 382, 470</td>
<td>WebSphere Application Server - Express 7, 33, 36, 49, 60, 382, 470</td>
</tr>
<tr>
<td>69</td>
<td>installation 69</td>
</tr>
<tr>
<td>35</td>
<td>WebSphere Business Integration Connect - Express 35</td>
</tr>
<tr>
<td>33</td>
<td>WebSphere Business Integration Server 4</td>
</tr>
<tr>
<td>33, 44, 47, 100, 247, 448, 500</td>
<td>WebSphere Business Integration Server Express 33, 44, 47, 100, 247, 448, 500</td>
</tr>
<tr>
<td>59, 73</td>
<td>installation 59, 73</td>
</tr>
<tr>
<td>4, 15, 44, 48, 159</td>
<td>WebSphere Business Integration Server Express Plus 4, 15, 44, 48, 159</td>
</tr>
<tr>
<td>6</td>
<td>installation 6</td>
</tr>
<tr>
<td>436</td>
<td>Activity Group 436</td>
</tr>
<tr>
<td>437</td>
<td>cases 437</td>
</tr>
<tr>
<td>436</td>
<td>Connector 436</td>
</tr>
<tr>
<td>436</td>
<td>Decision 436</td>
</tr>
<tr>
<td>436</td>
<td>External Entity 436</td>
</tr>
<tr>
<td>436</td>
<td>Go To 436</td>
</tr>
<tr>
<td>436</td>
<td>Input/Output 436</td>
</tr>
<tr>
<td>436</td>
<td>Partner Interaction 436</td>
</tr>
<tr>
<td>436</td>
<td>Process 436</td>
</tr>
<tr>
<td>436</td>
<td>Stop 436</td>
</tr>
<tr>
<td>436</td>
<td>Task 436</td>
</tr>
<tr>
<td>33, 37</td>
<td>WebSphere Commerce - Express 33, 37</td>
</tr>
<tr>
<td>172</td>
<td>WebSphere Commerce adapter 172</td>
</tr>
<tr>
<td>7, 49–50, 95, 169, 593, 628</td>
<td>WebSphere MQ 7, 49–50, 95, 169, 593, 628</td>
</tr>
<tr>
<td>67</td>
<td>installation 67</td>
</tr>
<tr>
<td>169</td>
<td>WebSphere MQ adapter 169</td>
</tr>
<tr>
<td>33, 38</td>
<td>WebSphere MQ Express 33, 38</td>
</tr>
<tr>
<td>33</td>
<td>WebSphere Portal Server Express 33</td>
</tr>
<tr>
<td>37</td>
<td>WebSphere Studio 37</td>
</tr>
<tr>
<td>60</td>
<td>Windows 2000 60</td>
</tr>
<tr>
<td>51</td>
<td>work in progress 51</td>
</tr>
<tr>
<td>317</td>
<td>wrapper 317</td>
</tr>
<tr>
<td>319</td>
<td>WrapperFoundation 319</td>
</tr>
<tr>
<td>28</td>
<td>WSDL 28</td>
</tr>
<tr>
<td>28, 435, 502</td>
<td>XML 28, 435, 502</td>
</tr>
<tr>
<td>234, 558</td>
<td>ODA 234, 558</td>
</tr>
<tr>
<td>169</td>
<td>XML adapter 169</td>
</tr>
<tr>
<td>11, 234, 558</td>
<td>XML schema, See XSD</td>
</tr>
<tr>
<td>28</td>
<td>XMLODA 11, 234, 558</td>
</tr>
<tr>
<td>28</td>
<td>XSD 28</td>
</tr>
<tr>
<td>53</td>
<td>z/OS 53</td>
</tr>
</tbody>
</table>
The Express Route to Business Integration
WebSphere Business Integration Server Express
WebSphere Business Integration Server Express
The Express Route to Business Integration

Integration basics
This IBM Redbook provides guidance for IT specialists and architects who are implementing enterprise application integration (EAI) projects for small and medium businesses. It provides best practices and advice for all levels of an EAI solution, including design, implementation and deployment using the WebSphere Business Integration Server Express.

Overview of WebSphere Business Integration Server Express
This redbook discusses types of application integration, such as data level (information) integration, application logic (API) integration and process integration, and considers different EAI architectures such as brokers or point-to-point integration.

Working example for new developers
It provides guidance for analyzing and understanding requirements such as real time, nearly real time and delayed integration and describes how to architect a manageable solution.

A sample application will be implemented which focuses on the development of a practical EAI example for developers using WebSphere Business Integration Server Express.

For more information:
ibm.com/redbooks