Using BPEL Processes in WebSphere Business Integration Server Foundation
Business Process Integration and Supply Chain Solutions

Using WebSphere Business Integration Server Foundation

Developing data aggregation and transformation scenarios

Streamlining the supply chain

Bill Moore
Jason Collier
John Mount
Naveen Sachdeva
Craig Spiteri
Darryl Whyatt

ibm.com/redbooks
Using BPEL Processes in WebSphere Business Integration Server Foundation - Business Process Integration and Supply Chain Solutions

September 2004
Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (September 2004)


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

Notices ................................................................. ix
Trademarks ......................................................... x

Preface .................................................................. xi
The team that wrote this redbook. ......................... xi
Become a published author ................................. xiv
Comments welcome ........................................... xiv

Chapter 1. Introduction ................................... 1
1.1 Who should read this publication .................... 4
1.2 What we do in this publication ....................... 4
1.3 How to use this publication ......................... 5

Chapter 2. Supply chain and business integration .......... 11
2.1 Supply chain concepts .................................. 12
2.2 Electronic commerce in the supply chain ........... 13
2.3 The supply chain exchange ............................ 14
2.4 Private Exchange scenario business problem ....... 16
2.5 Private Exchange scenario requirements overview . 18
  2.5.1 Functional requirements ............................ 18
  2.5.2 Non-functional requirements ...................... 19

Chapter 3. Solution design and technical overview ....... 29
3.1 Scenario introduction .................................... 30
3.2 Functional component breakdown ................... 30
  3.2.1 Enterprise information systems (EIS) ............ 30
  3.2.2 Exchange server ...................................... 32
  3.2.3 Data approval, update, and aggregation .......... 33
3.3 Process descriptions ..................................... 34
  3.3.1 Aggregating information across enterprise systems 34
  3.3.2 Approving item information ....................... 35
  3.3.3 Updating catalog information ..................... 36
  3.3.4 Processes after migration to BPEL ............... 37

Chapter 4. Key technologies .............................. 39
4.1 Standards ...................................................... 40
  4.1.1 Java and XML standards ......................... 41
  4.1.2 Web services ......................................... 42
  4.1.3 Business process management ................... 46
4.1.4 Business document formats .................................................. 51
4.2 Products .................................................................................. 52
  4.2.1 WebSphere Business Integration Server Foundation V5.1 .......... 53
  4.2.2 WebSphere Studio Application Developer Integration Edition V5.1 54
4.3 Applying BPEL to a supply chain scenario .................................. 55
  4.3.1 Scenario overview ............................................................... 56
  4.3.2 Mapping to UML sequence diagram ....................................... 57
  4.3.3 Building the public abstract process ....................................... 60
  4.3.4 Building the private executable process ................................... 70

Chapter 5. Using WebSphere Studio ............................................. 83
5.1 WebSphere Studio .................................................................... 84
  5.1.1 Highlights and benefits ........................................................ 84
  5.1.2 Configurations .................................................................... 86
  5.1.3 Packaging .......................................................................... 92
  5.1.4 Platform support ................................................................. 93
5.2 What is new in WebSphere Studio V5.1 ..................................... 94
  5.2.1 New functions and features ................................................ 95
  5.2.2 Changes in function and features .......................................... 95
5.3 Installing V5.1 .......................................................................... 96
5.4 Business integration perspective ............................................. 101
  5.4.1 Views ................................................................................. 102
  5.4.2 Tools on the toolbar ........................................................... 105
  5.4.3 Editors .............................................................................. 106
5.5 Using the WSDL editor ............................................................ 106
  5.5.1 WSDL elements ................................................................ 107
  5.5.2 WSDL editor views ............................................................ 108
  5.5.3 Editor panes ....................................................................... 108
  5.5.4 Tasks in the WSDL editor ................................................... 110
  5.5.5 Creating a sample WSDL .................................................... 115
  5.5.6 Best practices with WSDL files .......................................... 119
5.6 Using the BPEL editor ............................................................. 119
  5.6.1 Basic terminology used in BPEL .......................................... 119
  5.6.2 Anatomy of the BPEL editor .............................................. 120
  5.6.3 Activities and their icons .................................................... 121
  5.6.4 Tasks ............................................................................... 124
  5.6.5 Sample BPEL process ....................................................... 128
5.7 Common tasks .......................................................................... 165
  5.7.1 Create service project ......................................................... 166
  5.7.2 Create test server ............................................................... 167
  5.7.3 Enabling security on a test server ........................................ 169
  5.7.4 Adding and removing applications to the test server ............. 171
  5.7.5 Setting JMS properties in test server .................................... 172
5.7.6 Setting a project build path .................................................... 176
5.7.7 Exporting code for deployment ................................................ 184

Chapter 6. Deploying the existing sample solution ................................. 191
6.1 Sample code ............................................................................. 192
  6.1.1 Obtaining the sample code ..................................................... 192
  6.1.2 Overview of code ................................................................. 192
  6.1.3 Deployment architecture ....................................................... 196
6.2 Sample code deployment in WebSphere Studio Application Developer
    Integration Edition V5.0 ............................................................... 196
  6.2.1 Environment ...................................................................... 197
  6.2.2 Importing the sample code ................................................... 197
  6.2.3 Create a test server .............................................................. 198
  6.2.4 Deploy the application EAR files .......................................... 200
  6.2.5 Configure the test server ....................................................... 202
  6.2.6 Publishing and starting the server ......................................... 208
  6.2.7 Testing the deployed application .......................................... 210
6.3 Sample code deployment on WebSphere Application Server Enterprise
    V5.02 ......................................................................................... 220
  6.3.1 Prerequisite software and settings ......................................... 220
  6.3.2 Preparing the application server to run processes ................. 221
  6.3.3 Configuring the application server ....................................... 226
  6.3.4 Deploy the applications ....................................................... 231
  6.3.5 Starting and stopping the server and applications ............... 235
  6.3.6 Test sample code ............................................................... 235
6.4 Troubleshooting ........................................................................ 236
  6.4.1 Changing deployment descriptors for the Web service ........... 237

Chapter 7. Migrating to WebSphere Business Integration Server Foundation
                                                                 239
  7.1 Importing an existing workspace .............................................. 241
    7.1.1 Opening the Version 5.0 workspace ................................... 241
    7.1.2 Resolving broken library links and external JAR file locations .... 242
    7.1.3 Unnecessary BPE projects ................................................. 244
  7.2 Migrating the aggregation and approval business processes ............. 245
    7.2.1 Automatic migration of the aggservice project ................. 245
  7.3 Resolving possible migration errors .......................................... 247
    7.3.1 Staff activities ................................................................. 247
    7.3.2 Missing activity output variables ........................................ 248
    7.3.3 Mass update process names .............................................. 251
    7.3.4 Resolving approval process invocation errors .................... 252
    7.3.5 The exchange partner link ................................................. 257
    7.3.6 Integrate AggServiceMigrationV5 with the other projects ....... 259
Online resources ................................................................. 323
How to get IBM Redbooks .................................................. 325
Help from IBM ..................................................................... 325

Index .................................................................................... 327
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:

AIX®
ClearCase®
Cloudscape™
CICS®
DB2®
e-business on demand™
Informix®
IBM®
ibm.com®
IMS™
iSeries™
LOVEM®
MQSeries®
MVS™
OS/400®
PartnerLink®
Redbooks™
RRedbooks (logo) ™
Tivoli®
VisualAge®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

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

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.

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

This IBM Redbook examines business process integration using Business Process Choreographer to build solutions that help to streamline the supply chain process. We provide a sample scenario that integrates business processes to streamline a supply chain showing how suppliers and customers can use global repositories to synchronize data.

The sample scenario discussed in this redbook was initially implemented using WebSphere Application Server Enterprise Process Choreographer to aggregate and transform supplier product data. Data is then published to a global repository using the WebSphere Web Services Gateway, and also loaded to a catalog that uses a WebSphere Commerce portal.

We take this existing scenario and detail how to migrate the business processes to run on WebSphere Business Integration Server Foundation V5.1. As part of this migration effort we describe how to use WebSphere Studio Application Developer Integration Edition V5.1 as the development environment for process migration. The migrated sample uses Business Process Execution Language (BPEL) processes.

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 lab 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.

Jason Collier recently graduated from Pennsylvania State University, where he studied Computer Engineering and Economics, before joining the IBM Software Group System House Solution Test team in North Carolina, USA. In his current position Jason develops applications for and tests the integration of Tivoli® management networks and WebSphere Business Integration Server Foundation
technologies as part of the System House Solution Test team. Prior his current position he was a member of the IBM AlphaWorks Web Services Toolkit (WSTK) team.

John Mount started in software application design and development some 30 years ago by writing scouting programs in FORTRAN for his college football team. These days he pursues interests in wholesale banking, insurance, and automobile distribution applications using loosely coupled architectures and the Internet. John writes and speaks extensively on topics related to process integration and enterprise solutions using WebSphere Application Server and Business Process Choreographer in particular. He is Practice Leader for Business Integration Services with Commerce Solutions, Petaluma, California. John also owns a development company, Venue Software Corporation, in Sausalito, California, a firm he founded to develop JMS-based applications in WebSphere.

Naveen Sachdeva is a Software Advisory Engineer with IBM’s Application Integration Middleware group. He has over 10 years of experience in large-scale system integration, and the design and development of distributed computing systems. He currently helps IBM business partners with technical enablement and proof-of-concept using WebSphere products. He holds a Masters degree in Chemical Engineering from the Indian Institute of Technology, and an MBA from Carnegie Mellon University.

Craig Spiteri is an Advisory IT Specialist working in IBM Global Services Australia. He has ten years of experience in software development and support, with a major focus in the area of enterprise system integration. He has done extensive work in interfacing smaller Intel-based systems with large enterprise systems, including his latest project of enabling wireless access for field engineers to large backend systems. He holds a degree with honors in Computer Science from the University of Sydney, Australia.

Darryl Whyatt is a Consulting IT Architect in IBM Global Services Australia, with 18 years of professional experience in enterprise architecture and IT strategy consulting, complex systems solution architecture, performance and availability engineering, network and real-time systems engineering, and IT project management. He has specific expertise in enterprise application integration, object-oriented software development, middleware, operating systems, networking, and Internet technologies.
Thanks to the following people for their contributions to this project:

Julie Czubik  
International Technical Support Organization, Poughkeepsie Center

Peter Kovari  
International Technical Support Organization, Raleigh Center
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
Chapter 1. Introduction

This publication gives a focused tour through some of the features of WebSphere Business Integration Server Foundation V5.1. While the WebSphere Business Integration Server Foundation V5.1 product includes many features, our redbook concentrates primarily in a particular area: We are mainly interested in the new business process integration tools, including deployment and run-time details of the Business Process Choreographer. The V5.1 release is the one that introduces business process mapping based on Business Process Execution Language (BPEL). Since moving to this industry standard is an important step towards greater process interoperability both within and between enterprises, our goal is to examine the new products in this light. We especially examine the impacts on legacy choreographer applications, which are minimal, in order to offer some insight into working on new models with the WebSphere Business Integration Server Foundation tools in WebSphere Studio Application Developer Integration Edition. BPEL supersedes Flow Definition Markup Language (FDML) and is conceptually different. A natural consequence of this is the new models we can build have new tools we use to build them. For example, it is that not possible to create abstract process models without run-time capabilities; these can be shared between trading partners and used to govern message protocols without having to share internal interfaces or even be aware of them.

Our vehicle for this tour is a sample application provided by IBM’s System House, and our route is to re-implement it first on WebSphere Studio Application Developer Integration Edition V5.0.2, then move it onto WebSphere Studio Application Developer Integration Edition V5.1, and beyond that to get it running
on WebSphere Business Integration Server Foundation V5.1 using the new Business Process Container. Of course there are numerous places to stop and collect details along this path, and our technical focus falls mainly on the configuration of the BPE Containers and the allied installation issues for both run times we use.

Although our sample application has a simple design, the designers were also ambitious because they sought to involve many of the products in the WebSphere family. So the design is extensive but functionally limited. However it serves our needs in another way: The solution domain is a private exchange in a supply chain, and this is an excellent terrain for a wide variety of requirements that can be addressed by business integration tools and in particular by Business Process Choreographer.

Business Process Choreographer V5.1 is the first release of the product to take advantage of the new business integration standard based on BPEL, and the server platforms themselves have been repackaged and renamed. The move to BPEL means we have not only to consider how existing applications will migrate from earlier versions of Business Process Choreographer, but also how to take advantage of BPEL itself. BPEL differs substantially from its predecessor, FDML, not only in the process models it supports and the constructs in which to design them, but also with BPEL we have to learn a new process editor. So there is significant change in the development tools to consider. Moreover, the move to WebSphere Business Integration Server Foundation V5.1 means we need to consider deployment plans for the new platforms.

**Role of System House**
The WebSphere System House Business Scenario Development team is an integral part of an IBM Software Group initiative to ensure cross-product integration across IBM’s family of software products. The team identifies real business scenarios based on our customer experiences, business objectives, and use cases and the technology needed to fulfill the scenario requirements. They then implement a prototype of the scenario to discover problem areas, help IBM product teams recognize and resolve cross-product deficiencies, and ultimately deliver working solutions that satisfy the business requirements. This orientation is represented in Figure 1-1 on page 3.
System House provided our redbook sample application based on their Private Exchange scenario. This focuses on message flows between partners in a private exchange with data flowing into and out of independent operations within and beyond the enterprise. The business scenario for supply chain exchanges provides a one-stop shopping focal point for a set of suppliers and buyers. Suppliers make their products available to the exchange. Buyers shop at the exchange, using it as a source for product information that in turn they provide to their customers.

The sample application provided by System House thus supplies us with a realistic framework to examine the use of BPEL in a supply chain environment. The code was developed originally for WebSphere Application Server Enterprise V5.0 and implemented in WebSphere Studio Application Developer Integration Edition V5.0. We document deployment of the sample code in these environments because we want to analyze and discuss the requirements for this type of application. These are fairly generic. Then we bring the application into WebSphere Studio Application Developer Integration Edition V5.1 so we can learn what that entails. The next step is to deploy this code to WebSphere Business Integration Server Foundation V5.1. In the course of these tasks we took the relevant technical features of WebSphere Business Integration Server Foundation and of also WebSphere Studio Integration Edition V5.1.

BPEL presents us with a different way to model business processes. While not a complete departure from its predecessor, FDML, a lot has changed and a lot is new. Therefore we consider what aspects of our sample application we would do differently in BPEL. We are able to make some recommendations that will help you evaluate whether to migrate existing Business Process Choreographer applications or redesign them to take advantage of BPEL and the new deployment options of WebSphere Business Integration Server Foundation.
1.1 Who should read this publication

This publication is intended for you if you are:

- An enterprise application architect interested in V5.1 of WebSphere Business Integration Server Foundation, particularly in the release of Business Process Choreographer based on Business Process Execution Language for Web Services Version 1.1 (BPEL)
- A business analyst looking at supply chain integration opportunities or ways to model and implement connections between your business units and your business partners
- A developer who works on business process integration applications, especially if you are considering using Business Process Choreographer (or will migrate work flow applications from earlier releases to WebSphere Business Integration Server Foundation V5.1)
- An application designer responsible for business process integration or work flow applications
- A project manager who will plan or lead a project based on Web services, process integration, or work flow
- Someone who designs either abstract or executable protocols for integrating processes within your enterprise or between your company and your trading partners
- An operations planner or engineer interested in how to install and work with Business Process Choreographer and BPEL
- Evaluating tools for modelling business processes and process integration projects
- Someone who wants to look into the capabilities and possibilities of process integration based on BPEL
- An IT Planner responsible for services and services-oriented architectures to integrate enterprise applications
- Involved in evaluating technologies and tools involving process integration, supply chain partner connections, or workflow across and between enterprises

1.2 What we do in this publication

The approach we take is to consider a hypothetical business problem and conduct a simple analysis to determine its basic requirements. Next we discuss a simple solution that meets these requirements. This gives us the basis to look at
the key IBM technologies and products in the new release level of WebSphere Business Integration Server Foundation, Version 5.1. Then we use these tools to implement our example application. The simplest migration path possible is just to migrate our application from V5.0.2 to the new version, changing as little as possible. This is our first step. Perhaps more interesting, and certainly more challenging, is the question of how to migrate or transform an application to take advantage of BPEL. We do this in a limited fashion so we can illustrate what is involved.

1.3 How to use this publication

Not every section in this publication will be equally useful to you. What you want to look at depends on your role and interests as well as on how familiar you are with earlier versions of WebSphere Application Server and Business Process Choreographer. If these products are both new to you, most of this publication will be of some interest. Others will want to focus on only the material that is new to them.

We organized our chapters around the following topics:

- **Supply chain integration**
  Supply chain applications are commonly characterized by multiple sources and forms of content (data), which are nevertheless capable of being organized into internal business processes. This perspective motivates and defines our business problem in order to give us a clear and simple exercise in the analysis and design of a solution based on WebSphere Business Integration Server Foundation V5.1 and the new tools that come with WebSphere Studio Application Developer Integration Edition V5.1.
  - Chapter 1, “Introduction” on page 1, briefly outlines the sample application based on the System House Private Exchange scenario.
  - Chapter 3, “Solution design and technical overview” on page 29, translates the requirements from Chapter 2, “Supply chain and business integration” on page 11, into elements of the Private Exchange sample application originally developed for V5.0.

- **Business process modeling and BPEL**
  - Chapter 2, “Supply chain and business integration” on page 11, introduces BPEL.
– Chapter 4, “Key technologies” on page 39, presents our use of abstract processes to define BPEL-based services.

➤ Key technologies and products
– Chapter 4, “Key technologies” on page 39, describes the key IBM process integration technologies offered with WebSphere Application Server V5.1, namely, WebSphere Business Integration Server Foundation, the tooling in WebSphere Studio Application Developer Integration Edition and the Web Services Gateway and UDDI.

➤ Designing a business process using WebSphere Studio Application Developer Integration Edition V5.1
– Chapter 3, “Solution design and technical overview” on page 29, translates the technical requirements into technical components and processes.
– Chapter 5, “Using WebSphere Studio” on page 83, discusses the new tooling and BPEL editor used with Business Process Choreographer.

➤ Tooling
– Chapter 5, “Using WebSphere Studio” on page 83, reviews the tooling.
– Chapter 6, “Deploying the existing sample solution” on page 191, illustrates the use of the new tooling, and BPEL editor and can be used as a tutorial.
– Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239, is where we move the code to the new version of WebSphere Business Integration Server Foundation using WebSphere Studio Application Developer Integration Edition V5.1.

➤ Migration
– Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239, shows how to use WebSphere Studio Application Developer Integration Edition.
– Chapter 8, “Deploying sample on WebSphere Business Integration Server Foundation” on page 271: We discuss in detail how we migrated our sample application from V5.0 to V5.1. Key points of where the versions are the same as well as where they depart are discussed as we encounter them. Several migration options exist. Your choices will depend on what you want to do with your application after you move to the new version. Our goal was to migrate and then convert our sample application from V5.0.2 to V5.1 to be able to take advantage of BPEL in the future. Therefore we discuss this process in detail.
Deployment

- Chapter 6, “Deploying the existing sample solution” on page 191, covers WebSphere Studio Application Developer Integration Edition and WebSphere Application Server Enterprise.

- Chapter 8, “Deploying sample on WebSphere Business Integration Server Foundation” on page 271: First we deployed the sample application on WebSphere Studio Application Developer Integration Edition using the embedded Extended Edition server. Then we did this again on the new WebSphere Business Integration Server Foundation platform. We found this fairly straightforward and similar to earlier release procedures; however, there are some changes to the WebSphere Business Integration Server Foundation Administrative Console. If you have WebSphere Business Integration Server Foundation installed you can follow the steps we took to get our solution running in V5.1, but if you do not have the product, this chapter will still give you a look at what the new deployment environment and process entails.

Reading plans
Clearly this publication touches on a wide range of business and technical topics. Perhaps you will be able to read it all, but if you are like most of us your time is limited. Here are several reading plans you can follow according to your interests:

- Supply chain data management, applications, and solutions:
  - Chapter 2, “Supply chain and business integration” on page 11
  - Chapter 3, “Solution design and technical overview” on page 29

- Web services and service-oriented application development, focus on:
  - Chapter 2, “Supply chain and business integration” on page 11
  - Chapter 4, “Key technologies” on page 39

- What is new in WebSphere Business Integration Server Foundation Server Foundation V5.1 is mainly in:
  - Chapter 4, “Key technologies” on page 39
  - Chapter 6, “Deploying the existing sample solution” on page 191
  - Chapter 8, “Deploying sample on WebSphere Business Integration Server Foundation” on page 271 (especially V5.1 BPE Container configuration)

- If you are going to be involved in any phase of application development with Business Process Choreographer you will want to read:
  - Chapter 3, “Solution design and technical overview” on page 29
– Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239, using WebSphere Studio Application Developer Integration Edition

► WebSphere Business Integration Server Foundation V5.1 and BPEL:
  – Chapter 4, “Key technologies” on page 39
  – Chapter 8, “Deploying sample on WebSphere Business Integration Server Foundation” on page 271

► If you are just getting started on WebSphere Application Server and process integration you may find most of the publication useful, although you may want to skip over Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239, if you do not have any legacy Choreographer applications.

► How to migrate business processes and Business Process Choreographer applications from FDML to BPEL:
  – Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239, using WebSphere Studio Application Developer Integration Edition
  – Chapter 8, “Deploying sample on WebSphere Business Integration Server Foundation” on page 271

► If you are already comfortable with business integration and Web services concepts, as well as how to implement these using Business Process Choreographer, the most interesting material will be the parts about how to move to the new BPEL-based V5.1. If this is the case, review the application chapters just to get a sense for our supply chain application. Then concentrate on Chapter 4, “Key technologies” on page 39, and Chapter 5, “Using WebSphere Studio” on page 83, where we discuss the V5.1 generation of the products.

► Business process integration tooling in WebSphere Studio Application Developer Integration Edition V5.1:
  – Chapter 5, “Using WebSphere Studio” on page 83
  – Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239, using WebSphere Studio Application Developer Integration Edition

► If application development with Business Process Choreographer is your main concern, read Chapter 3, “Solution design and technical overview” on page 29.
Deployment in WebSphere Business Integration Server Foundation V5.1 is discussed in Chapter 8, “Deploying sample on WebSphere Business Integration Server Foundation” on page 271.

For the V5.1 release the WebSphere Application Server Enterprise product has been rebranded as WebSphere Business Integration Server Foundation. The best place to look for what this entails is Chapter 4, “Key technologies” on page 39.

Also refer to the redbook WebSphere Business Integration Server Foundation V5.1, SG24-6318.
Supply chain and business integration

This chapter provides an introduction to supply chain concepts and electronic commerce in the supply chain domain. It describes the business problem and requirements for the Private Exchange scenario presented in this redbook.

We also describe an approach to defining requirements in the functional and non-functional domains to provide guidance on how to successfully apply the technologies and techniques described in this redbook to real projects.
2.1 Supply chain concepts

A supply chain may be defined as a network of storage and production facilities interconnected by transportation and logistics. The purpose of a supply chain is to source materials, transform them by one or more steps of processing or manufacturing, then deliver finished goods to a customer. In practice, a supply chain for complex manufactured goods will involve many transformation steps performed by many different companies.

A subpart of a supply chain will also look like a supply chain itself. In this case, materials on the input side would equate to the finished goods from another supply chain, and the finished goods on the output side would equate to the materials for yet another supply chain. For those familiar with fractals or geometric forms in nature, supply chains may be seen to exhibit similar recursive characteristics.

Supply chain management is concerned with optimizing the performance of a supply chain. Although end-to-end supply chain optimization may be possible in theory, in practice it is performed from the perspective of a company and its immediate trading partners and customers.

Supply chain management may be divided into the following areas:

- Production
  Production is concerned with forecasting, planning, and scheduling processing or manufacturing facilities to meet expected demand. Aspects such as which products, what type and configuration, how many, where, and when need to be considered. Production efficiency usually varies with run size, with larger runs typically more efficient than smaller runs. Facilities may have different production capacities. Excess or shortfall in production can have a direct impact on inventory levels.

- Inventory
  Inventory is concerned with managing stocks of production inputs (materials), outputs (finished goods), or intermediate components. Inventory creates a buffer between various stages of production, or between links in the supply chain. Part items can be classified in terms such as part number, type, configuration, variant number, or color. Other types of material may require continuous quantification such as weight, length, volume, percentage, concentration, or other grading scale. Inventory may be interchangeable as in the example of concentration or grade versus weight or volume. Linear quantities may be on rolls or other discrete lengths. Inventory may also have a time component, as in the case of perishable goods. Low inventory risks production stalling on the inputs side, and delayed supply to customers and trading partners on the outputs side. High inventory increases costs in
storage and working capital, and may cause wastage in the case of perishable goods.

► Location

Location is concerned with the relative placement of material sources, production facilities, storage facilities, distribution, and customer market. Some decisions made in this area will have a long-term effect in cases where they represent major investments in plants, buildings, infrastructures, human resources, or other relatively fixed location resources. Location will have a direct impact on transportation costs. It can also have a major effect on costs in areas such as taxes, duties and tariffs, property values, wage costs, and currency exchange rates. Both production and inventory are directly linked to location.

► Transportation

Transportation is concerned with moving inventory between locations to meet the needs of material sources, production, storage, or supply to customers and trading partners. There are trade-offs in mode of transport (road, rail, air, or sea) to do with cost, time to deliver, and location of transport infrastructure such as railways, ports, or airline terminals. The mode of transport is also related to the shipping quantities and inventory levels.

Successful modeling and optimization of supply chains and inventory can return significant business benefits. For example, in the late 1980’s IBM developed a decision support system called Optimizer that resulted in a reduction in inventory in excess of $250 m, savings in annual operating costs of $20 m, and improved customer service.

2.2 Electronic commerce in the supply chain

Electronic commerce in the supply chain has existed for nearly 30 years, largely in the form of Electronic Data Interchange (EDI). EDI stills drives more than 95 percent of the electronic commerce in the world today.

The most common EDI standards are:

► ANSI ASC X.12 - Developed by the American National Standards Institute (ANSI) and popular in North America

► UN/EDIFACT - Developed by the United Nations Centre for Electronic Commerce for Administration, Commerce, and Transport (UN/CEFACT) and popular in Europe and Asia

EDI document exchange was typically conducted using X.400 messaging over Value Added Networks (VANs) provided by major telecommunication providers.
However, EDI standards are technology independent, allowing EDI document exchange to be conducted over the Internet using common protocols such as File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP).

Although EDI is mature and stable, it has the following disadvantages.

- Although EDI is technology independent, many EDI implementations still use VANs. These services are often charged on a subscription plus per-message or data volume based charges. As transaction volumes increase so do costs. Many EDI implementations are now perceived as expensive to operate.

- As XML increasingly becomes the dominant and universal standard for data formatting, EDI formats are by comparison inflexible and more difficult to work with.

- EDI software has been typically quite expensive, effectively discouraging wider adoption by larger numbers of smaller companies. Software for dealing with XML and related standards such as SOAP and WSDL is now widely available and either inexpensive or free.

- The wide availability and accessibility of XML tools has a corresponding positive effect on the availability and cost of skilled development resources.

The movement towards XML and Web services as the ubiquitous platform for electronic commerce has clearly commenced, and achievement of this goal at some time in the future appears inevitable. However, technologies in the form of standards and products are still undergoing rapid evolution, and early adoption can be challenging.

Electronic commerce in the supply chain takes on many forms such as direct interaction between business partners, catalogue aggregation, trading hubs, public or open exchanges, and private exchanges. Since this redbook deals with an exchange scenario, further elaboration is provided in the following section.

### 2.3 The supply chain exchange

The purpose of an exchange in this context is analogous to the role that an exchange plays in any market (for example, stocks, futures, commodities, or foreign currency); namely to bring buyers and sellers together in one logical place and time to buy and sell, and to set prices by some method based on supply and demand.
Exchanges may be described by one or more of the following characteristics.

- **Large buyer**
  A large buyer may establish an exchange to generate competition among sellers, thereby reducing prices. The buyer needs to be dominant enough to be able to set standards for the exchange and motivate sellers to participate.

- **Buyer aggregation**
  A consortia of often small buyers may establish an exchange to consolidate purchasing into larger volumes, thereby taking advantage of volume discounts and increased buying power. The cost of implementing and operating the exchange can also be shared among the consortia members.

- **Large seller**
  A large seller may establish an exchange to maintain prices at the highest level that still drives the required sales volume. The seller needs to be dominant enough to be able to set standards for the exchange and motivate buyers to participate.

- **Seller aggregation**
  A consortia of often small sellers may establish an exchange to compete with larger competitors, or gain entry into markets from which individual sellers may be excluded because of their size. The cost of implementing and operating the exchange can be also be shared among the consortia members.

- **Buyer-seller neutral**
  A neutral exchange is established to bring together buyers and sellers without any particular bias. In this case, the motivation is usually simply to generate revenue from operation of the exchange itself by charging either buyers, sellers, or both.

- **Private**
  A private exchange operates within a closed domain of business partner relationships, usually for the direct benefit of one or more of the parties.

- **Public**
  A public exchange is open to all buyers and sellers, and is usually operated by an independent third party for the purpose of generating revenue from the exchange itself.

- **Vertical**
  A vertical exchange is aligned to the trading needs of a particular industry segment.
Horizontal
A horizontal exchange is aligned to product, service, or commodity types, and is industry agnostic.

Set price
The seller sets a price for all buyers (excluding discounts or premiums that may apply).

Negotiated price
The buyer and seller negotiate directly to arrive at a mutually acceptable price.

Auction
Initiated by the seller, the price is set by the highest offer coming from a group of competing buyers.

Reverse auction
Initiated by the buyer, the price is set by the lowest offer coming from a group of competing sellers.

Closed tender
This is similar to a reverse auction, except that sellers do not see each others’ offers, and each offer is first and final.

2.4 Private Exchange scenario business problem

The scenario we use in this redbook leverages assets developed by the IBM System House group. IBM System House is an integral part of an IBM initiative to ensure cross-product integration across IBM's family of software products. The group identifies real business scenarios based on customer experiences, business objectives, use cases, and the technology needed to fulfill the scenario requirements. They then implement a prototype of the scenario to discover problem areas, help IBM product teams recognize and resolve cross-product deficiencies, and ultimately deliver working solutions that satisfy the business requirements.

In the Private Exchange scenario, a supplier operates a private exchange for selling their product lines. The exchanges provide access via Web services for buyers to integrate their supply chain systems.

The exchange needs to be kept up to date with item descriptions, pricing, and availability from the company's line of business systems that manage this data. The supplier also publishes this information to a commerce Web site to allow buyers to purchase using a Web browser.
Given our definitions introduced in 2.3, “The supply chain exchange” on page 14, the exchange in this scenario can be described by the following characteristics:

- Large seller - Established by the seller with the intent that buyers will be motivated to participate
- Private - Owned and operated by the seller
- Horizontal - No specific industry alignment
- Set price - Prices set by seller, not dynamically by the exchange

The key components of the required solution are shown in Figure 2-1.

![Figure 2-1](Private%20Exchange%20scenario%20solution%20overview)

The components in this diagram are as follows.

- EIS1 holds item information such as item number, description, and supplier. New items are created in this system.
- EIS2 holds item pricing information.
- EIS3 holds item availability information.
- Business Process Manager is the new system that we will build in this scenario.
- Approver Staff represents the staff who approve new items to be added to the exchange and the commerce Web site.
- Exchange is the private exchange to support electronic commerce via Web services.
- Commerce is the Web site to online purchasing using a browser.
Supply Chain System represents the buyer’s system that interacts with the exchange.

Buyer Staff represents the staff who buy items using the commerce Web site.

To update item entries in the exchange and commerce Web site, we need to bring together description, pricing, and availability information from EIS1, EIS2, and EIS3. To create new item entries, we also need to include an approval step. The aggregated information then needs to be published to the exchange and commerce Web site.

2.5 Private Exchange scenario requirements overview

For all IT development, including supply chain solutions, it is constructive to partition requirements into the following domains:

- Functional
- Non-functional

Functional requirements define the behavior of the system and the manifestation of its interfaces. Functional requirements are related to the primary motivation for the implementation of the system. Functional requirements will often be traceable to specific aspects or areas of the solution. For example, the following requirement may map to a single user interface page.

When the payment method has been confirmed, the system shall present a screen to the user to allow them to review and confirm the transaction.

Non-functional requirements define the technical and operational characteristics of the solution. They define the manner in which the functional requirements are to be met, and the conditions under which it is expected that the system will operate. Non-functional requirements may have a wider or global impact on the solution. For example, the following requirement may impact all user interface pages and could potentially influence the overall solution architecture.

The maximum system response time as perceived by the user shall not exceed 20 seconds.

Failure to define non-functional requirements in a manner that is both realistic and measurable is a common cause of project failure and client dissatisfaction.

2.5.1 Functional requirements

The functional requirements elicitation and analysis process would in most real project cases use and produce a range of artifacts such as:

- Classified business terms
For brevity, we will only define the functional requirements for the supply chain scenario in this redbook using use cases.

**Use case #1 - Update item entry in Exchange and Commerce**
On a scheduled basis, the Business Process Manager triggers itself to perform a mass update of the Exchange and Commerce Web site. For each item in EIS2 (the pricing system), aggregate description information from Exchange (this could also have been EIS1), pricing information from EIS2, and availability information from EIS3. The resultant data is sent to Exchange and Commerce.

**Use case #2 - Create new item entry in Exchange**
A user creates an entry for a new item in EIS1. Business Process Manager is notified of this event and sends the new item description to Approver Staff for approval. If approved, it is aggregated with pricing information from EIS2 and availability information from EIS3. The resultant data is sent to Exchange and Commerce.

### 2.5.2 Non-functional requirements
Since the Private Example scenario presented in this redbook is a sample application only, no non-functional requirements were defined. However, we will present here a rigorous approach to defining a non-functional requirement that is applicable to most if not all IT solutions, including supply chain integration.

Irrespective of the advanced capabilities of the tools and infrastructure available to develop these solutions, due care must still be taken to define and realize a solution’s non-functional requirements.

Non-functional requirements may be classified into the following areas:

- Performance
- Scalability
- Availability
- Maintainability
- Security
- Manageability
- Environmental
- System usability
- Data integrity

These areas are described in the following sections.

**Performance**
Performance requirements define the static and dynamic parametric characteristics of the system, and should be defined in the following areas.

- **Response time**
  Response time is the elapsed time taken by a system to complete a specific process. Some examples are user wait time in interactive user interfaces and two-way Web services.

  A common trap is to be specific about the time parameter (number of seconds, for example), without being specific about the definition of the start and end of the process to which it applies. With a Web application, for example, is the response time measured from the user action to the start of the next page being displayed, the first complete visual screen (making an assumption about effective screen size), or completion of the page load in the browser? Each interpretation may be valid in a given context.

- **Throughput**
  Throughput is the workload capacity of a system, and is usually specified in terms of number of units of work per time period. Some examples are transactions per second, settlement transactions per day, business process instantiations per day, and Web service invocations per hour.

- **Utilization**
  Utilization is the percentage of a system resource being used for a given workload. For example, a system could be specified to use no more than 70 percent of available memory and 50 percent of available processor capacity while performing to the specified response time and throughput requirements.

- **Static volumetrics**
  Static volumetrics are counts or sizes of entities within a system that are likely to be significant in terms of system architecture or operational dimensioning. An example of static volumetrics affecting system architecture are the assumed practical size limits of RAM or disk capacity and their association with the addressing schemes used to access them. Some examples affecting operational dimensioning are number of customer records, number of accounts, number of products, number of registered users, and number of concurrent user sessions.
Scalability

Scalability requirements define the expected response required to accommodate change from the specified performance requirements. Scalability requirements are by nature speculative in part, given that they address future scenarios that may not necessary be tested or testable in the delivered system.

However, scalability requirements can be used to provide a technical and commercial boundary around client expectation. For example, the following requirement could be used to define the processing scalability for a system.

The system shall be scalable to 10 times the currently specified throughput within 2 years through the addition or replacement of hardware alone, and without change to application code. The hardware cost shall be less than or equal to 10 times the original hardware cost (assuming flat or reducing hardware price/performance ratio).

Continuous improvement in the price/performance ratio of hardware over time works in favor of future scalability within reasonable cost boundaries. Moore’s Law, which observes that performance or capacity doubles every 18 months (or so), continues to be reasonable guide. However, scalability requirements significantly more aggressive than this need to be treated with due consideration.

Even if not stated, scalability requirements typically assume little or no application code change. Obviously, most systems are scalable if they can be totally redeveloped.

Requirements can be for scaling upward, downward, or both. For example, a system that might need to be scalable to a high availability server farm might also need to be downwardly scalable to a single processor personal workstation.

Extreme caution must be taken with scalability requirements for a system that has operational dependencies on one or more resources outside of its scope. Typically, these will be existing systems owned or used by the client, but increasingly they may be external Web services over which both the system developer and client has little control. In our redbook supply chain scenario, this would equate to EIS2 and EIS3, and the exchange itself. Scalability of the new system is a moot point if a dependency on an external resource acts as a constraint that cannot be circumvented.

Availability

Availability requirements define what, when, who, and from where users or other systems can access the system being developed. Availability can be finer grained than simply ‘on’ or ‘off’. For example:

- Certain functions (such as update) may be unavailable at particular times of the day, week, month, or year.
A system failure may impact all or only part of the system’s functionality.

General user access may be unavailable while the system is used exclusively by business or system administrators.

Failures or planned outages may only effect users in certain locations or geographies.

The following aspects should be covered when specifying availability:

- Normal hours of operation (do not overlook the effect of multiple time zones)
- Planned outages (for maintenance, system upgrade, and so on)
- Unplanned outages (number of failures, time to recover from failure)
- Disaster recovery

Avoid or seek clarification for availability requirements expressed as a flat percentage, as in the following example.

The system shall be available for 99% of the time within normal business hours.

Assuming a 10-hour day and a 5-day week, this requirement could be met by either one 26-hour (2.6-day) failure per year, one 12-hour (1.2-day) failure every month, one 30-minute failure every week, or one 6-minute failure every day. It is almost certain that the reliability of the system would be perceived quite differently with each of these failure characteristics.

**Maintainability**

Maintainability requirements define the expectation as to the level of effort required to either change the functionality of the system, or maintain the functionality of the system in a changed technical environment.

Aspects used to define these characteristics would include:

- Configurability
- Flexibility
- Portability
- Interface independence
- Product independence
- Technology independence

Some maintainability characteristics such as product or technology independence may be expensive to implement, but have a very delayed or non-existent return on investment. It should be understood that these forms of maintainability may or may not be appropriate given the circumstances.
Security

Security is a critical aspect of most electronic commerce solutions, especially those dealing with areas such as proprietary or confidential business information, personal information, or commercial transactions.

An understanding of the following primary security domains will assist the process of identifying security requirements.

- **Confidentiality**
  Information is disclosed only to users authorized to access it.

- **Integrity**
  Information is modified only by users who have the right to do so, and only in authorized ways. It is transferred only between intended users and in intended ways.

- **Accountability**
  Users are accountable for their security-relevant actions. A particular case of this is non-repudiation, where responsibility for an action cannot be denied.

- **Availability**
  Use of the system cannot be maliciously denied to authorized users.

Requirements in these domains typically give rise to solution features in the following categories.

- **Identification and authentication**
  This is the process of establishing the identity of a principal and verifying that they are who they claim to be. Principals may be either human users or system objects that need to operate under their own rights.

- **Authorization and access control**
  This is the process of determining whether or not a principal can access a system object. The decision is made on the basis of the identity and authorization of the principal and the access control attributes of the target object. Authorization may be indirectly defined for a principal via role assignments or group membership. Access control attributes may be applied to objects of either a functional or data-oriented nature.

- **Security auditing**
  The purpose of auditing is to make users accountable for their security-related actions (non-repudiation). It is normally the human user who should be accountable. Auditing mechanisms should be able to identify the user correctly, even after a chain of calls through many objects or Web services.
Security of communication

Communication is frequently over insecure lower layer communications, and in most instances it is best to assume this to be the case. Security of messages in transit may involve mutual authentication and trust between client and target (perhaps involving a third-party trusted authority), measures to detect malicious data modification (integrity), measures to void malicious session replay, and measures to ensure that data is not subject to eavesdropping (confidentiality).

Administration

Administration is required for security information such as usernames and passwords, role associations and group memberships, access control lists for target objects, auditing policies, and security policies.

Availability management

Protection is required from malicious or inadvertent disruption to system availability through events such as request flooding, request malformation, and requests resulting in excessively resource intensive response.

Manageability

Manageability requirements are associated with production operations and systems management. They typically exclude business or user administration requirements, as these would be specified in the functional domain. The following subject areas should be considered.

Installation

Software installation requirements include areas such as level of automation versus manual processes, attended or unattended installation, packaging for electronic distribution, and type of user interface support.

Startup and shutdown

Server startup and shutdown requirements include areas such as level of automation versus manual processes, server boot time startup control, control or redirection of load prior to shutdown, graceful termination of in-progress processing or user sessions, and scope of control (for example, server, cluster, or domain).

Notifications

In this context, an event is defined as a status associated with a predetermined trigger condition being met, and an alarm is defined as a status requiring human operator observance or intervention. The associativity between events and alarms will be highly solution specific.

Notification requirements include areas such as event generation, event correlation, analysis, response, log file usage, alarm determination, alarm
notification via e-mail/pager/SMS, SNMP traps, and enterprise management console integration.

- **Configuration management**

  Configuration management requirements include areas such as software component upgrade procedures, upgrade rollback procedures, planned outage management, level of automation, level of user interface support, integration with systems management and software deployment environments, level of support for multiple concurrent versions of interfaces, and level of support for multiple concurrent versions of business process instances (alternatively referred to as macroflows, interruptible processes, or long running transactions).

- **Backup and recovery**

  Backup and recovery requirements include areas such as system restoration time, whole or partial data recovery, backup media handling and storage, backup media cycling and retention policies, disaster recovery, and business continuity.

**Environmental**

Environmental requirements arise as a result of the environment or context in which the system is to be developed or deployed. Note that only non-functional environmental requirements should be included here. Explicit system features needed to meet legal, regulatory, safety, or technical requirements should be specified as functional requirements. For example, a state law may exist that requires all transaction audit logs to be stored on at least two discrete media stored in physically separate locations for a minimum of seven years. This would normally be classified as a non-functional requirement. Conversely, calculation of a state tax in accordance with a formula defined in tax legislation would be classified as a functional requirement.

The following subject areas should be considered.

- **Legal**

  Legal requirements include areas such as applicable international laws, common law, national laws, and state legislation.

- **Regulatory**

  Regulatory requirements include areas such as industry regulation or self-regulation, codes of practice, and agency compliance.

- **Safety**

  Safety requirements apply to systems that have the potential to affect human health and safety. This would include systems such as medical equipment, automotive control systems, air traffic control, and railway control systems.
Safety requirements would not normally apply to supply chain solutions, but have been included here for completeness.

- Technical

Technical requirements (or constraints) would specify the required compliance in areas such as enterprise architecture, technology strategy, technical standards, standard operating environments, and corporate procurement policies. A well-defined Enterprise Architecture (EA) would normally include a standard process to review and approve requests for compliance exemption if technically and/or financially justified.

System usability

System usability requirements define the features and characteristics required to ensure that the system is able to be used effectively by its intended human users. Requirements should consider:

- Who are the users (normally categorized into user profiles)?
- What are their characteristics in terms such as knowledge, skills, experience, language or cultural background, and physical capabilities?
- What physical device or devices are to be used for the interaction between user and system (for example, personal computer, personal digital assistant, mobile phone, touch-screen kiosk, or telephone)?
- What is the environment for the interaction between user and system (for example, lighting levels, ambient noise levels, public/private space, or mobile/fixed location)?
- What user interface standards might apply (for example, platform standards, client branding or look and feel standards, or industry standards or conventions)?
- What help methods are to be available?

Usability requirements should be written in such a way as to be verifiable by an objective test. For example, there is little value in the requirement:

The system shall be easy to use.

It would be preferable to specify the level of proficiency that would be expected to be achieved by an average user within a user profile after a nominated period of training and/or familiarization.

Although supply chain integration solutions are primarily concerned with application-to-application integration, process ‘staff’ steps introduce human users into the solution. Requirements for system usability should not be overlooked in this context.
Data integrity
Data integrity requirements define the requirements for data handling both within the system and across system interfaces. Areas to be considered are:

- Transactional characteristics (ACID properties of atomicity, consistency, isolation, and durability)
- Scope of transactions
- Interruptible processes (also known as long-running processes or macroflows) versus non-interruptible processes (also known as short-running processes or microflows)
- Transaction rollback, exception handling, and compensation
- Data retention requirements (for example, policies for audit data or policies for process instance data)
Solution design and technical overview

In this chapter we take the requirements laid out in Chapter 2, “Supply chain and business integration” on page 11, and translate them directly into technical components and processes. We look critically at how a typical inventory control system in a supply chain might look. The details covered include the functional components of our solution, the technologies utilized in the solution, and an in-depth look at the business processes we defined.
3.1 Scenario introduction

The code chosen to work with for this redbook was provided to us by the IBM System House group. This group inside of IBM focuses on engineering sample customer IT solutions and then uses this sample solution to implement a real-world scenario of the technology. Implementing and analyzing scenarios helps improve IBM product quality and technology.

The scenario that we use is called the Private Exchange (which we often abbreviate as PE), and it models a hypothetical supply chain and the interface between the supply chain and a parts exchange specific to the supplier.

3.2 Functional component breakdown

While this redbook is mainly interested in the operations of the supplier's business process logic, we must also be aware of the overall environment that this process will function in. As in any organization, a typical IT system will contain components of various complexities, age, flexibility, and maintainability, which this environment emulates through the following logical systems.

3.2.1 Enterprise information systems (EIS)

These systems, representing the data inputs to our business processes, can largely be viewed as existing legacy systems. The need, for our purposes, to model such systems is so that the business processes we are implementing have logical, reasonable data to manipulate. Remember that in all instances these enterprise information systems are self-contained, independently operational, fully functional systems. They are not dependent on any externally defined business processes to function. The processes we will define simply tie together their functionality.

In our example, we will utilize three such services:

- EIS1 - The system responsible for the insertion of new items into the supply chain system.
- EIS2 - The system responsible for item pricing information.
- EIS3 - The system responsible for available item quantity information.

EIS1

EIS1 is the interface for employees inside the supplier's company to introduce new items into the supplier's parts exchange.
All of the information related to an item is input from a single screen except for the pricing and the available quantity of an item. These remaining two pieces of information are the responsibility of entirely separate enterprise systems, EIS2 and EIS3.

EIS1 was implemented by the System House group as an IBM WebSphere MQ based application with a JavaServer Pages (JSP) Web page frontend. WebSphere MQ was utilized for this application to exercise the ability of WebSphere Application Server to interface with Extended Messaging.

Since EIS1 is the gateway for new items being added to the supplier’s inventory, it is also a natural invocation point for one of the business processes we define later: The approval of new items being introduced.

**EIS2**

The second system we look at is EIS2, responsible for maintaining accurate pricing information on all of the items in the supplier’s inventory. EIS2, as originally designed by System House, is a JCA-based Java™ application interfacing to a backend Oracle database system. For the purposes of application testing and the design of the architecture there is no formal GUI to the EIS2 system. Updates and changes to the pricing information of EIS2 are to be made directly at the database level.

**EIS2 details**

EIS2 is normally invoked by passing an XML message to the service. Table 3-1 shows details of the message interfaces used by EIS2 and Example 3-1 shows a sample XML return message from EIS2.

**Table 3-1  EIS2 message interfaces**

<table>
<thead>
<tr>
<th>Application</th>
<th>Operation</th>
<th>Input message</th>
<th>Output message</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIS2</td>
<td>getPartPrice</td>
<td>getPartPriceRequest</td>
<td>getPartPriceResponse</td>
</tr>
</tbody>
</table>

**Example 3-1  Sample XML return message for EIS2**

```xml
<?xml version="1.0"?>
<ExchangeDBroot>
  <Part>
    <ItemNumber>IN1111111</ItemNumber>
    <Price>19.95</Price>
  </Part>
</ExchangeDBroot>
```
EIS3
Last there is EIS3, responsible for the maintaining and providing the available quantity information of each item. EIS3’s data is also maintained in a similar Oracle database; however, access to it is through a message driven bean with a JMS transport layer. Like EIS2, EIS3 also lacks a GUI for data manipulation. Changes to quantity information is similarly performed at the database level. Table 3-2 lists the available functions provided by EIS3, and Example 3-2 shows a sample XML return message from EIS3.

Table 3-2 Available functions for EIS3

<table>
<thead>
<tr>
<th>Application</th>
<th>Operation</th>
<th>Input message</th>
<th>Output message</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIS3</td>
<td>sendWithResponse</td>
<td>sendWithResponse Request</td>
<td>sendWithResponse Response</td>
</tr>
</tbody>
</table>

Example 3-2 Sample return message for EIS3

```xml
<?xml version="1.0"?>
<ExchangeDBroot>
  <Part>
    <ItemNumber>IN1111111</ItemNumber>
    <Quantity>100</Quantity>
  </Part>
</ExchangeDBroot>
```

3.2.2 Exchange server

The exchange server is one of two externally accessible systems that the supplier will interact with. We implement the exchange service as an EJB accessible to users via the Web. For a view of what the exchange server looks like from the GUI see Figure 3-1 on page 33. Again, for our needs the actual implementation of a robust exchange service is not critical; all that is necessary is a realistic interface to the exchange for our business processes.

To interface with the exchange we use a Web service and SOAP over HTTP. As another step towards real-world accuracy, the IBM System House group implemented interfacing with the exchange through a Web Services Gateway.

Web Services Gateway

A Web Services Gateway, such as the one provided in WebSphere Application Server Network Deployment, is used to provide a level of abstraction between the calling client Web services code and the Web service provider. This additional layer, along with the extra features provided by the gateway, allow for more flexibility in architectures. Through a gateway it is possible to encrypt HTTP transmissions, providing for more secure data communications, and to perform
manipulations on the SOAP messages passing through the gateway for added versatility in your Web services client.

The exchange GUI provides basic functionality to perform queries for all parts in the exchange, returning a table listing of all relevant item information.

![Exchange Repository Query](image)

**Figure 3-1**  The exchange GUI

### 3.2.3 Data approval, update, and aggregation

There are two main activities that we wish to solve through the use of business processes for our supplier:

- Approving new items that the supplier is offering for inclusion into the item exchange
- Providing updates to an online Web commerce system for the items that the supplier offers

These two processes demand very different requirements. From a user’s perspective there are two ways to interact with these business processes.

To approve new items, users must have a way to interact with the system so that it recognizes new items. Once this is accomplished, there must be a way for
users to verify and confirm that the new item is in fact approved for entry into the exchange.

To update the supplier’s data in the Web commerce system, there must be a form of scheduled or real-time updates to change any necessary information. In contrast to the first process, this process must be highly automated with little or no user intervention.

Common to both processes is the need to interface with the supplier’s legacy information systems. The disparate data kept in individual repositories must be aggregated together so that the processes can manipulate the data as needed.

Taken together, these three processes are implemented as two uninterruptible processes and one long-running, or staff interruptible, process.

### 3.3 Process descriptions

The logic of the supplier’s solution will originally be defined in three FDML business processes. These processes are used by the supplier to approve new parts for sale and to update the company’s product listings in an online commerce system. In later sections of this publication we migrate these processes to the most current BPEL business process technology.

#### 3.3.1 Aggregating information across enterprise systems

Before discussing either the approval or publishing processes, we must first understand the aggregation process. The aggregation process is called from inside of both the other processes and is used to gather possibly disparate data from the EIS2 and EIS3 pricing and inventory systems.

In Figure 3-2 on page 35 you can see the exact paths of execution for the aggregation process. The service is invoked at the Input node with the part number passed to it from the calling application. After invocation, the prepare activity is run to read the part number passed into the process and to set the proper fields for the getAvailability and getPartPrice service calls.

From the prepare step, the getAvailability and getPartPrice activities are called simultaneously to acquire the necessary part data from the EIS backend systems. After the execution of the EIS activities, the collect activity reports the data queried from the EIS’s and readies an XML message for the publish step.

In the version of the supplier scenario we are using for this redbook, the publish step is not implemented. This step was provided in the process originally designed by IBM System House so that it could interface with IBM’s WebSphere
Commerce offering. With this process the newly approved or updated products could be passed on to an online shopping portal. Since our intention is to illustrate the migration and implementation of business processes, we leave the addition of WebSphere Commerce, or any other online shopping application, out of our examples. However, the option is always there for future use should you so desire. The publish step would normally place an XML message onto a specific JMS queue for consumption by an external application such as WebSphere Commerce. For our uses, manually browsing the queue and reading the message will be enough to see that this step of the process is functioning properly.

![Aggregation Process Diagram](image)

**Figure 3-2  Aggregation process diagram**

### 3.3.2 Approving item information

The first externally accessible process we discuss is the approval process shown in Figure 3-3 on page 36. This process is needed to approve new parts that the supplier is offering into the part exchange and for inclusion into an online commerce system.

This process starts out similarly to the aggregation process with a prepare activity. This activity echoes the item number and the description being processed for approval, and performs a quick test based upon the supplier’s name field whether approval can be bypassed or not. This is most useful as a debugging feature; however, it is possible that bypassing approval of certain parts based upon one of their attributes may be valid business logic. If the supplier’s name is equal to “jibe” the item will bypass approval.

After this step there is a branch in logic depending upon whether approval is required. If it is required a staff activity will be executed; if it should be bypassed, the aggregation process we defined previously will be invoked.
Assuming approval is required, a staff activity is executed so that there can be human intervention in the process. The staff activity will wait on input from a human operator who accesses the process through the BPE Web client (located at http://<hostname>:9080/bpe/webclient). From the Web client the user can see a list of available items awaiting approval, claim an unclaimed item, and finally decide whether to permit or deny continuation of processing for the item.

Rejecting the item will trigger the reject event, printing the rejection event to the WebSphere system log, and ending the process’ execution at the output node. Approving the item will lead to the triggering of the aggregate process, the same point in the process that would be reached had approval been bypassed in the prepare activity. For a further explanation of the aggregation process see 3.3.1, “Aggregating information across enterprise systems” on page 34.

After the aggregate activity, an XML message is prepared for the call to createEntry via the prepareForExchange activity. After createEntry updates the Exchange server with the new item information, the process completes.

**Figure 3-3  Approval process diagram**

**3.3.3 Updating catalog information**

The mass update process exists for the purpose of fully refreshing the contents of WebSphere Commerce, or any other online Web commerce application that can interface through a WebSphere MQ queue. The process begins by retrieving a listing of all of the parts found in EIS2, and then iterates through a loop retrieving the product description, price, and available quantity that is sent one part at a time onto a queue. Visually, the process is defined as in Figure 3-4, with the process loop defined as in Figure 3-5 on page 37.

**Figure 3-4  Mass update process**
To begin, in prepare_to_get_all_parts we prepare a call to EIS2 with the item number equal to a percent character (%) wildcard. This triggers unique logic inside EIS2 to return a price listing for every part in its system, and thus every part that the supplier would have to offer. After prepare_to_get_all_parts, getPartPrice is called to invoke the EIS2 application and perform the wildcard query.

The loop defined in Figure 3-5 then iterates through all of the item numbers returned from the EIS2 wildcard query. For each item we call getDescription to access the exchange and retrieve item description information. A call to prepareToAggregate creates an XML message specific to the item we are processing. This message is then sent to the agg_service aggregation process, as defined in 3.3.1, “Aggregating information across enterprise systems” on page 34. The aggregation call, as explained previously, acquires all the relevant item information and copies it onto a WebSphere MQ queue for consumption by an outside application, presumably WebSphere Commerce.

![Figure 3-5](image)

*Figure 3-5  Inside the mass update loop*

After the loop terminates, the mass update process has completed.

### 3.3.4 Processes after migration to BPEL

The process logic should remain unchanged after we migrate them to BPEL processes. There will be slight changes in the definitions of the processes, but they will functionally remain the same. As a brief preview, converting the processes to BPEL will force us to convert our JMS-invoked processes, namely the approval process, into one-way processes, as opposed to the request/response input/output design discussed previously.

All of these changes and the steps involved in the FDML to BPEL migration are covered in detail in Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239.
Key technologies

In this chapter, we introduce the key technologies used in the supply chain scenario in this redbook, and also discuss technologies generally used in electronic commerce. We focus on standards and products that were used to develop the sample solution, or that would be likely to be required in real projects based on the WebSphere business process management architecture.

Business Process Execution Language for Web Services (BPEL4WS, or simply BPEL) is the central element to business process development in WebSphere Studio Application Developer Integration Edition V5.1. We provide a detailed example to illustrate the key concepts and provide guidance on how BPEL may be applied in real projects.
4.1 Standards

Standards play an essential role in creating an environment that encourages, supports, and makes cost effective the development of electronic commerce solutions. More particularly, it is convergence to common standards that accelerates technology adoption in an enterprise context.

During phases of rapid innovation, competing standards often emerge, and this is not necessarily either avoidable or undesirable. However, at some point in time, consensus on a common standard is needed to allow the next phase of innovation to flourish. For example, for some time TCP/IP and ISO Open Systems Interconnection (OSI) competed in the data networking standards domain. The emergence of TCP/IP as the single dominant standard for open global data networking has effectively caused a paradigm shift in how we look at information technology today.

In an obvious way, standards are very effective as the basis for achieving inter-operability between systems developed by different vendors or implemented by different companies. However, convergence on common standards and ubiquitous deployment also plays a vital role in reducing the cost of solution development and implementation. For example, HTTP (or HTTPS) does not necessarily have all of the characteristics desired for transport of commercial documents and transactions; however, its widespread deployment and pre-existing firewall traversal support in most corporate networks makes it a practical and cost-effective choice.

We will describe standards in the following areas:

- Java and XML related
- Web services
- Business process management
- Business document formats

No distinction is made between de jure, de facto, or proposed standards in this context. A de jure standard is one that has been approved by a standards organization. A de facto standard is one that is widely used within an industry and is generally accepted as a standard. A proposed standard is one that is put forward as a potential standard, but has not yet achieved this status.

A supply chain integration scenario will be presented to illustrate how BPEL and WSDL may be applied in a practical example.
4.1.1 Java and XML standards

Java and XML play a fundamental role in the WebSphere business process management architecture. The Java API for XML-based RPC (JAX-RPC) and XML Path Language (XPath) in particular are described here as they are used in WebSphere Studio Application Developer Integration Edition V5.1.

Java API for XML-based RPC (JAX-RPC)

JAX-RPC provides a Java programming model for XML-based remote procedure calls (RPC). It defines mappings between XML and Java data types and provides development tools for generating XML-to-Java and Java-to-XML bindings. For interoperability, at least SOAP 1.1 over HTTP 1.1 must be supported by compliant implementations.

JAX-RPC can be used to create Web services clients and endpoints. Web service endpoints may be implemented as either Enterprise JavaBeans or Java Servlets. The language independence of Web services allows JAX-RPC clients and endpoints to interoperate with other Web service implementations across heterogeneous platforms. It has been adopted by J2EE 1.4 as the core Web services API for the Java platform.

JAX-RPC is currently limited in scope to invocation of SOAP services.

Prior to JAX-RPC, IBM developed the Web Services Invocation Framework (WSIF). WSIF is an API for invoking any Web service described in Web Services Description Language (WSDL), regardless of implementation. It defines WSDL bindings so that Enterprise JavaBeans, enterprise software accessible through Java Message Service (JMS), or J2EE Connector Architecture (JCA), and local Java classes can be described as first class Web services and then invoked using the same protocol-independent WSIF API.

WebSphere Application Server added JAX-RPC support from V5.0.2. WebSphere Studio Application Developer V5.1 comes with WebSphere Application Server V5.0.2 run time. WebSphere Studio Application Developer Integration Edition V5.1 includes full JAX-RPC support and tools to generate JAX-RPC proxies from WSDL.

The JAX-RPC standard was developed under the Java Community process as JSR-101. Further information on JAX-RPC may be found at:

http://java.sun.com/xml/jaxrpc

XML Path Language (XPath)

XPath is a language to access and operate on data within an XML document. It was created to provide a standard syntax and base function library to support Extensible Stylesheet Language Transformations (XSLT) and XML Pointer
Language (XPointer). XSLT is a language for transforming XML documents, and XPointer is a language to be used as a fragment identifier for accessing data within an XML document located by a URI.

XPath is used in WebSphere Studio Application Developer Integration Edition V5.1 to define transformations within a transformer service. A transformer service allows one or more messages to be transformed into an output message. The transformation service and the editor support within WebSphere Studio Application Developer Integration Edition provides the primary vehicle for mapping messages from one format to another, a very common requirement in electronic commerce and business process integration solutions.

Simple copies, extractions, and joins may be defined directly with the assistance of the wizard; however, for any more complex transformation, XPath can be used directly.

XPath is a World Wide Web Consortium (W3C) standard. Further information on XPath may be found at:

http://www.w3.org/TR/xpath

4.1.2 Web services

Web services are system interfaces based on standards that allow systems to interoperate without human intervention. Web services specifically exploit the technologies and infrastructure of the Internet originally deployed to support Web browsing by users using a browser.

For example, suppose a new plug-in is required by an application. Prior to Web services, the following scenario would have been quite typical. First the user would use a Web browser to access a search engine and submit an appropriate search term. From the result set (often large, and containing many irrelevant items), the user would select the correct Web site, and navigate within the Web site to find the link to download the plug-in. Once downloaded, the user would then locate the file on her local system and run the plug-in installation program to install it. With Web services, this entire scenario could be implemented by the application itself without intervention or assistance from the user.

Although Web services were primarily intended for the Internet, it is increasingly common to see Web services being considered for application integration projects within enterprise confines. The same dynamics that resulted in many companies adopting Internet and Web technologies on their private intranets are likely to apply in the case of Web services. These dynamics include:

- Wide availability
- Low cost
- Platform independence
Global standardization

The key standards in Web services are:

- Web Services Description Language (WSDL)
- Universal Description, Discovery, and Integration (UDDI)
- SOAP

The WSDL and UDDI standards were developed specifically for Web services, and are described further in the following sections. Many other standards are being proposed by many different vendors and standards bodies that layer upon Web services to support enhanced capabilities, many of these driven by the needs of electronic commerce. Some of these are:

- WS-Addressing
- WS-Policy
- WS-Security
- WS-SecureConversation
- WS-Trust
- WS-ReliableMessaging
- WS-Coordination
- WS-Transaction
- WS-BusinessActivity
- WS-Routing
- WS-Federation
- WS-Eventing
- Web Services Choreography Description Language (WS-CDL)
- Business Process Execution Language for Web Services (BPEL)

We will describe the standards relating to business process management in more detail in 4.1.3, “Business process management” on page 46.

For further information on Web services, the World Wide Web Consortium (W3C) Web Services Architecture (WSA) definition may be found at:

http://www.w3.org/TR/ws-arch

Web Service Description Language (WSDL)

WSDL is the XML-based language used to describe a Web service. The language contains the following primary constructs.

- Messages and parts

  A message defines the structure of the information to be passed to or from a Web service. A message is made up of parts that represent the data entities within the message. Parts may either be simple or complex data types. A message and its parts may be likened to a Java class with attributes only.
Port types and operations

An operation is an action that can be invoked on a Web service. An operation is associated with up to three messages, one each for input, output, and fault handling. Operations are logically grouped into port types. A port type and its operations may be likened to a Java class with methods only.

Bindings

A binding links a port type to a communication protocol. The nominal protocol for Web services is SOAP; however, other protocols such as HTTP, Java, EJB, and JMS may be used.

Services and ports

A service is the Web service being described. A service is made up of one or more ports. A port has a physical location (for example, a URL) and association with a binding. Through the binding, the port has its port type and hence operations defined. A port may be likened to an instantiation of a Java class (a Java object) with methods only.

A schematic representation of these concepts is shown in Figure 4-1. This illustration is a screen capture from the Graph window of the WSDL editor in WebSphere Studio Application Developer Integration Edition V5.1.

Figure 4-1  Graph view of WSDL file showing relationship between key elements

The corresponding XML for the schematic above is shown in Example 4-1 on page 45. This is available in the source window in the WSDL editor. Note that it should be rarely required to understand and edit WSDL files directly in XML when using WebSphere Studio Application Developer Integration Edition. However, this view can be useful when resolving errors generated by the WSDL or BPEL validation tools.
Example 4-1   Source view of WSDL file in Figure 4-1 on page 44

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="wsdl" targetNamespace="http://wsdl/wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <message name="Message">
    <part name="Part" type="xsd:string"></part>
  </message>
  <portType name="PortType">
    <operation name="Operation">
      <input message="tns:Message"></input>
    </operation>
  </portType>
  <binding name="Binding" type="tns:PortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"></soap:binding>
    <operation name="Operation">
      <soap:operation
soapAction="http://wsdl/wsdl/Operation"></soap:operation>
      <input>
        <soap:body use="literal"></soap:body>
      </input>
    </operation>
  </binding>
  <service name="Service">
    <port name="Port" binding="tns:Binding">
      <soap:address location="http://www.ibm.com" />
    </port>
  </service>
</definitions>
```

Web services and WSDL play a primary role in the definition of business processes and business protocols in BPEL, and hence editing WSDL files is a common task.

WSDL is a World Wide Web Consortium (W3C) standard. Further information on WSDL may be found at:

[http://www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)

**Universal Description, Discovery, and Integration (UDDI)**

UDDI provides specifications for publishing and discovering Web services through the use of registries. The Web services provider publishes information to the registry, and the Web services consumer searches the registry to find the required Web services. UDDI also includes specifications for the protocols used for inter-node communication within multi-node registry implementations.
The UDDI registry is business-centric. In this context, a *business* may be any entity such as a company, organization, or government agency that provides Web services. The registry entry for a business would include information in the following key areas:

- The business name, address, contact details, and any other relevant identification information
- The industry sector and/or category of the business based on standardized or common classification codes or taxonomies
- The Web services provided by the business
- The technical binding details required to access Web services provided by the business

A UDDI registry may be classified in terms of its scope as follows.

- **Public**
  - A public registry is deployed on the Internet and intended for public use.
- **Private**
  - A private registry is deployed on an intranet and intended for private use within a business.
- **Semi-Private**
  - A semi-private registry is deployed on an extranet and intended for use by a business and its business partners.

Much of the initial UDDI standards development focussed on establishing a centralized public UDDI Business Registry (UBR). The UBR was envisaged as being to Web services what the white pages and yellow pages are to telecommunications, with the same ubiquity as the Internet Domain Name Server (DNS) environment. However, this has largely failed to materialize, and the most common implementations today are either private or semi-private.

If a UDDI registry was used in this redbook supply chain integration scenario, it would have been semi-private.

UDDI is an Organization for the Advancement of Structured Information Standards (OASIS) standard. Further information on UDDI may be found at:

http://www.uddi.org

### 4.1.3 Business process management

Web services provide a basic one-way or request-response mechanism that can be used by two systems to communicate. Web services standards are open, cross platform, and fully aligned with Internet standards and technologies.
However, it is widely recognized that the interaction of several or many Web services is often required to create business value. This has led to several initiatives to create languages to express and define business processes that coordinate Web services.

Business processes may be classified as either executable or abstract. These and other related terms are defined as follows.

- **Executable process**
  
  An *executable* process defines a coordinated set of activities that are performed within a single scope of control in response to a business event. The single scope of control aspect is important because it allows an executable process to be controlled directly by a system. In an electronic commerce context, executable processes exist within business entity boundaries, and their details are not visible externally. Hence they are also referred to as *private*. The term *orchestration* has come into common use to refer to the coordination of Web services in a private executable context.

- **Abstract process**
  
  An *abstract* process defines the interaction between two or more independent systems via their external interfaces in response to a business event. It can also be referred to as a *business protocol*. An abstract process is by definition not directly controlled by any of its participating entities, and hence cannot be implemented as an executable system. In an electronic commerce context, abstract processes exist between business entity boundaries and are visible externally. Hence they are also referred to as *public*. The term *choreography* has come into common use to refer to the coordination of Web services in a public abstract context.

These concepts and terms are depicted in Figure 4-2.
The key standards in the business process management domain is described in the following sections.

**Business Process Modelling Language (BPML)**
BPML was released in March, 2001 by the Business Process Management Initiative (BPML) as a proposed standard language for describing executable processes. BPML refers to ebXML and RosettaNet as a means to define abstract processes.

Further information on BPML may be found at:

http://www.bpmi.org/bpml.esp

**Web Services Flow Language (WSFL)**
WSFL was published in May, 2001 by IBM as a language for defining both abstract and executable processes based on Web services. WSFL refers to abstract and executable processes by the terms *global model* and *flow model*, respectively. WSFL has been superseded by BPEL (described later in this chapter).

Further information on WSFL may be found at:


**Flow Definition Markup Language (FDML)**
FDML was developed by IBM as a proprietary language for defining executable processes. FDML was used in a number of IBM products, including the former MQSeries® Workflow and WebSphere Studio Application Developer Integration Edition V5.0. FDML has been superseded by BPEL.

**XLANG**
XLANG was published in 2001 by Microsoft® as a language for defining abstract and executable processes. XLANG was used in the BizTalk Server product family. XLANG has been superseded by BPEL.

Further information on XLANG may be found at:

http://www.gotdotnet.com/team/xml_wsspecs/xlang-c

**Web Services Choreography Description Language (WS-CDL)**
WS-CDL was released in April, 2004 by the World Wide Web Consortium (W3C) as a standard language for defining abstract processes. It does not support executable processes. WS-CDL is independent of the platform or programming model used to implement the Web services participating in a collaboration, and therefore is intended to inter-operate with BPEL executable processes.
Since BPEL is also able to describe abstract processes, WS-CDL and BPEL are, at the time of writing this publication, competing with each other in this domain. It remains to be seen whether or not the synergy between BPEL abstract and executable processes will outweigh benefits attributed to a language designed for abstract processes only. Like all standards, the market will decide.

Further information on WS-CDL may be found at:

http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427

**Business Process Execution Language for Web Services**

BPEL 1.0 was published in July, 2002 by IBM, Microsoft, and BEA Systems as a proposed standard for defining abstract and executable processes. BPEL supersedes IBM's WSFL and FDML, and Microsoft's XLANG.

The language was updated to Version 1.1 in May, 2003 with the additional co-authors SAP and Siebel Systems. This version was submitted at that time to the Organization for the Advancement of Structured Information Standards (OASIS) for further development and endorsement as an OASIS standard.

BPEL is dependent on the following standards:
- WSDL 1.1
- XML Schema 1.0
- XPath 1.0
- WS-Addressing

BPEL exploits the synergy between abstract processes and executable processes by defining a core set of common constructs with extensions added to support each of these distinct process types.

The language includes the following key elements:
- **Partner link type**
  A partner link type defines the relationship between two Web services by specifying the role played by each of the services the port types used in the interaction.
- **Variable**
  A variable is used to store a local copy of a message or data in a process instance.
- **Correlation set**
  A correlation set defines the message properties (aliases for message parts) to be used to match incoming messages to the correct process instance. A correlation set is needed if a process has more than one receive or pick
activity. A correlation set, once initialized, acts as a constant for the rest of the process lifetime.

- **Receive**
  A receive is used to wait for a request message sent to a Web service implemented by the process.

- **Reply**
  A reply is used to send a response message to the sender of the request message that initiated the process instance.

- **Pick**
  A pick is used to receive a message via one or more port types and select a branch based on message type received.

- **Sequence**
  A sequence is used to perform a set of activities in a series.

- **Flow**
  A flow is used to perform a set of activities in parallel.

- **Link**
  A link is used to create a synchronization point between parallel activities.

- **Switch**
  A switch is used to select a branch based on the evaluation of an expression.

- **While**
  A while loop is used to iterate under the control of a boolean expression.

- **Invoke**
  An invoke is used to call a Web service.

- **Wait**
  A wait is used to delay for a period of time.

- **Empty**
  An empty activity does nothing. It may be used as a placeholder for an activity yet to be specified, or when a fault needs to be caught and suppressed.

BPEL also has constructs for expression handling, defining scope, event handling, fault handling within units of work, and compensation to reverse the effect of committed units of work. Further information on BPEL may be found at:

4.1.4 Business document formats

Web services provide the necessary infrastructure to move business information between business partners in an electronic commerce context. However, a shared understanding of the content, format, and semantics of these documents is required for meaningful communication between business partners.

It is obviously possible for a business and its business partners to work together to create and agree on a set of private document formats. However, this can be a very costly and time-consuming approach. Dominant vendors can create document formats; however, these are often skewed towards the needs of their own products, and do not solve the inter-operability problem when multiple vendors’ products are involved.

Broad adoption of electronic commerce is accelerated when entire industry sectors work together to develop common standards. This is amply demonstrated by EDI with its success and longevity. However, with Web services and XML, we are faced with this challenge again.

XML is often described as self describing. This can be taken to mean that an XML document can be understood by the receiver as its sender intended without prior knowledge or context. This misnomer is supported by the clear-text appearance of XML tags, as in simple cases meaning can be inferred (guessed, that is) from the name of the tag. For example, an XML tag of *price* could be easily inferred to mean the cost to the buyer. But does it include locally applicable sales taxes or duties? The tag name alone cannot define this. Outside of the XML document and its schema, sender and receiver need to have a common definition.

Along with XML business document format standards developed within large business partner networks and industry sectors, there have been several initiatives to define broadly applicable standards for use across multiple industry sectors. A few examples are as follows.

- **XML Common Business Library (xCBL)**
  
  xCBL originated in 1997 at Veo Systems, a company that was later acquired by Commerce One, Inc. xCBL was evolved to be inter-operable with EDI. Further information on xCBL may be found at:
  
  http://www.xcbl.org

- **RosettaNet**
  
  RosettaNet is an industry-based standards organization formed in 1998 by a consortium of major IT and Telecommunications companies. RosettaNet
produces standards for business documents and business processes. Further information on RosettaNet may be found at:

http://rosettanet.org

➤ Electronic Business XML (ebXML)

ebXML commenced as a initiative of the Organization for the Advancement of Structured Information Standards (OASIS) in 1999. It leverages the strengths of EDI, and covers a broad range of electronic commerce requirements beyond that of business document formats. Further information on ebXML may be found at:

http://www.ebxml.org

➤ Commerce XML (cXML)

cXML was developed by a consortium of companies led by Ariba, Inc., first made available in February, 1999. cXML focusses on providing simple business document schema that intentionally diverge from some of the complexity aspects of EDI. Further information on cXML may be found at:

http://www.cxml.org

➤ Universal Business Language (UBL)

UBL was announced in October, 2001 by the Organization for the Advancement of Structured Information Standards (OASIS) out of recognition of the benefits of reducing the number of competing XML business document standards. Their stated aim is for UBL to become an international standard for XML-based electronic commerce. UBL is based on xCBL 3.0, but the intention is to develop it in harmony with ebXML and EDI. Further information on UBL may be found at:

http://www.oasis-open.org/committees/ubl

An essential early step in any Web services electronic commerce project is to make an informed, considered, and mutually agreed upon decision regarding the business document formats to be used. Until a single and clearly dominant standard emerges for your industry or across all industries, this will probably be one of the most challenging aspects of your integration project.

4.2 Products

When IBM WebSphere Business Integration Server Foundation V5.1 is used in conjunction with WebSphere Studio Application Developer Integration Edition V5.1 for development, it can deliver a next generation integration platform optimized for building and deploying composite applications that extend and integrate your existing IT assets.
This section discusses the features of WebSphere Business Integration Server Foundation V5.1 and WebSphere Studio Application Developer Integration Edition V5.1.

4.2.1 WebSphere Business Integration Server Foundation V5.1

WebSphere Business Integration Server Foundation V5.1 builds on the WebSphere Application Server to provide a premier Java 2 Platform, Enterprise Edition (J2EE), and Web services technology-based application platform for deploying enterprise Web services solutions for dynamic e-business on demand™.

It represents IBM’s approach to building and deploying SOA-based applications that can adapt quickly and easily to change. It is designed to support the creation of reusable services (either new ones or those based on existing services, back-end systems, Java assets, and packaged applications). Services can then be combined to form both composite applications and business processes, which can further leverage business rules to make these applications and business processes adaptable.

WebSphere Business Integration Server Foundation V5.1 includes all of the features available in WebSphere Application Server Network Deployment V5.1, including:

- J2EE 1.3 support (support for some features planned for J2EE 1.4)
- Full XML support
- Full Web services support
- Support for private UDDI registries
- Web Services Gateway
- Database connectivity
- Embedded HTTP server
- Web server plug-ins
- Authentication and authorization for secure access to Web resources
- Single sign-on and support for LDAP
- Java Message Service (JMS) support
- Dynamic caching
- IBM Tivoli Performance Viewer
- Integration with third-party performance management tools
- Browser-based administration and workload management
- Intelligent workload distribution across a cluster
- Failure bypass
- Clustering support
- Migration support
Platforms
The platforms are:

- Solaris
- Linux (various distributions)
- HP-UX
- AIX®

Further information on WebSphere Business Integration Server Foundation V5.1 may be found at:

http://www.ibm.com/software/integration/wbisf/requirements

4.2.2 WebSphere Studio Application Developer Integration Edition V5.1

WebSphere Studio Application Developer Integration Edition V5.1 provides the tools you need to create, develop, test, and manage all of the resources involved with building Web and enterprise-scale Java 2 Enterprise Edition (J2EE) and Web services applications. WebSphere Studio Application Developer Integration Edition V5.1 offers creation tools, editors, wizards, templates, and code generators that help you rapidly develop J2EE resources such as HTML files, JSP pages, Java classes and servlets, EJB beans, and XML deployment descriptors. You organize these resources into projects that correspond to modules defined in the J2EE specification. Once the resources have been created, you can easily test and debug them within the development environment, or export and test them on a remote server.

A major focus of this product is on improving developer efficiency, as reflected in its existing functionality as well as new features introduced as part of V5.1. The product allows users to visually develop business processes, and V5.1 updates this capability with a new business process designer and debugger that support the creation of process flows that conform to the Business Process Execution Language for Web Services (BPEL) 1.1 standard. V5.1 also includes a new editor for the Web Services Description Language (WSDL) that simplifies user interaction with the product and adds visual clarity to how the various components interact.

In summary, the major new features in WebSphere Studio Application Developer Integration Edition V5.1 are as follows:

- Business process designer for creating BPEL process flows
- Integrated visual BPEL debugger
- Enhanced performance for installing and debugging, including support for J9 Hot Swap
- New visual condition builder to direct the execution of BPEL processes
- Automated migration of process flows from Flow Definition Markup Language (FDML) to BPEL

WebSphere Studio Application Developer Integration Edition V5.1 also includes the following Programming Model Extensions (PMEs) that build on J2EE standards to accelerate large-scale application development:

- Asynchronous beans
- Startup beans
- Last participant support
- Internationalization service
- Work areas
- Scheduler service
- Activity session services
- Dynamic query service
- Web Services Gateway Filters
- Object pools
- Container Managed Messaging
- Distributed Map
- Container Managed Persistence over anything
- Application profiling
- Backup Cluster Support

Platforms
The platforms are:

- Windows XP
- Windows 2000
- Windows NT®
- Linux (various distributions)

Further information on WebSphere Studio Application Developer Integration Edition V5.1 may be found at:

http://www.ibm.com/software/awdtools/studiointegration/

### 4.3 Applying BPEL to a supply chain scenario

Given that BPEL is the primary language used to develop business process and integration solutions in WebSphere Studio Application Developer Integration
Edition V5.1, we will explore this area further, looking at how it can be used in a practical example.

The starting point is two business partners who wish to integrate their supply chain systems. They work through from a high-level narrative of the collaboration scenario to precise definitions of the public abstract process that describes their agreed upon business protocol, and the private executable processes that implement the solution within their respective organizations.

We show how an abstract process definition may be directly used in the definition of an executable process. This ensures that the executable process conforms to the business protocol agreed upon with the business partner. Abstract process definitions may be further exploited in the future to provide run-time business protocol conformance validation and executable process code generation.

The processes created in this chapter are supplied as part of our redbook additional material. For instructions on obtaining the additional material refer to Appendix B, "Additional material" on page 319. For details of how to work with the sample code relevant to this chapter see Appendix A, "Working with the redbook sample code" on page 315.

We used WebSphere Studio Application Developer Integration Edition V5.1 and its BPEL editor to develop the samples in this chapter. For an overview of how to work with WebSphere Studio Application Developer Integration Edition see Chapter 5, “Using WebSphere Studio” on page 83. For more details see the redbook *WebSphere Business Integration Server Foundation V5.1*, SG24-6318.

### 4.3.1 Scenario overview

The example is based on a purchasing scenario between buyer and seller. An overview of the overall process flow between buyer and seller and within each organization is described in the following narration.

1. The buyer wishes to purchase goods from the seller, so they send them a Purchase Order with a buyer-assigned Customer Order Number.
2. The seller's Order Fulfilment department confirms that they have sufficient inventory to fulfil the order, and sends back a Purchase Order Confirmation with both the original Customer Order Number and a seller-assigned Purchase Order Number.
3. The buyer matches the Purchase Order Confirmation with their original Purchase Order using the Customer Order Number. From now on, the seller-assigned Purchase Order Number will be used for all matching.
4. The seller’s Order Fulfilment department then picks, packs, and dispatches the goods, and sends a Delivery Note to the buyer. The same department also sends a Dispatch Note to the Accounts Receivable department within the seller organization. Accounts Receivable then sends an Invoice to the buyer.

5. The buyer’s Goods Receiving department, on receiving both the goods and the Delivery Note, confirms that they match. If so, they send a Delivery Receipt to the Accounts Payable department within the buyer organization. Accounts Payable then confirm that the Delivery Receipt matches the Invoice and sends the Payment to the seller.

We have introduced notional roles (as might be systems or departments) within the respective buyer and seller organizations. We have also taken a transparent view through both organizations to validate the basic process mechanics. In a real project situation, this type of discussion might well take place in initial all-party workshops. However, once the public business protocol is agreed to, the true details of the roles and interactions within each organization can remain private.

Given that we want to define this scenario as one public abstract process representing the business protocol, and at least one private executable business process within each organization, we will need to add two additional roles that may appear redundant, namely Buyer and Seller. These roles are introduced so that the real system and department roles can be obscured behind them and remain private. In reality, they are the actual executable processes that implement the public Web services published by each organization.

4.3.2 Mapping to UML sequence diagram

The overall process is documented in the UML sequence diagram shown in Figure 4-3 on page 58. The vertical lines represent actors equating to the roles discussed above, and the arrows represent the messages passed between them. Time flows from top to bottom. Note that the order numbers described as used for matching messages are not depicted in this diagram, however, we will see how they are used later.
This type of diagram is very effective to use in a workshop and white board situation. For those more familiar with Line Of Visibility Enterprise Modelling (LOVEM®) or swim lane charts, these can be used to equal effect. (In a superficial way, a LOVEM or swim lane chart can be likened to a UML sequence diagram rotated 90 degrees counter-clockwise.)
The process of mapping this to Web services (WSDL definitions) and abstract and executable processes (BPEL definitions) is as follows:

- Each of the actors (vertical lines in the diagram) represents a Web services provider with at least one port type in a WSDL definition.
- If an actor is implemented by a business process, it will also have a BPEL definition. Note that Web services are not constrained to be only implemented by business processes; any realization that can be bound in a WSDL definition is permissible.
- Each of the arrows represents a Web service operation invocation. The actor on the arrow head end is the provider of the Web service.
- The Buyer and Seller actors implement the boundary between public and private processes, and participate in both. To separate these domains, we separate the Web services they provide into two distinct port types, public and private. In this scenario, Buyer has port types BuyerPublic and BuyerPrivate, and Seller has port types SellerPublic and SellerPrivate. The public and private port types for a single actor will be defined in separate WSDL definitions, as we will see later.
- The other actors participate exclusively in private processes and have been defined with one port type each, implicitly private.

The diagram in Figure 4-4 on page 60 shows these entities and their relationships.
We now describe the steps to create this scenario using WebSphere Studio Application Developer Integration Edition V5.1. We will create the public abstract process representing the business protocol between buyer and seller first, then use this definition to create the Buyer private executable process. The Seller process may be created in the same way, but is not described in this publication.

It is assumed that you are familiar with WebSphere Studio Application Developer Integration Edition V5.1 and its WSDL and BPEL editors.

### 4.3.3 Building the public abstract process

First we create the initial business process for the abstract process called Protocol that defines the business protocol between Buyer and Seller.

1. In WebSphere Studio Application Developer Integration Edition, open the business integration perspective. Create a service project and name it Purchasing, accepting all defaults.

   Create a business process with package your_domain.protocol and file name Protocol. Click **Next** rather than **Finish**. The next dialog presents three
options, as shown in Figure 4-5: Flow-based BPEL Process, Sequence-based BPEL Process, and WSAD-IE v5.0 Business Process. Since we are exploring BPEL in this scenario, we will not consider the third option. The choice of flow-based or sequence-based will set the default BPEL construct used in the editor; however, it does not prevent the other from being used. Select **Sequence-based BPEL Process** and click **Finish**.

![Choose Process Type](image)

*Figure 4-5  Process type selection in business process creation wizard*

2. Next we will edit the WSDL file for protocol.

   Open *ProtocolInterface.wsdl* in the WSDL editor. Delete the default *ProcessPortType* and *InputMessage* definitions. Create the port types, operations, messages, and parts, as shown in Figure 4-6 on page 62. When complete, save it.
We elected to use only one-way operations with no outputs or faults defined to preserve asynchronous or loose coupling between systems participating in the abstract process.

Note that in this one WSDL file we have defined the public port types and associated operations, messages, and parts for all of the entities participating in the abstract process. We will see why this is important later.

The source view of the WSDL file for Protocol is shown in Figure 4-2.

Example 4-2  Source view of WSDL file for Protocol in purchasing scenario

```xml
<definitions
    targetNamespace="http://www.example.com/process33748449/interface"
    xmlns="http://schemas.xmlsoap.org/wsdl/
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/
    xmlns:tns="http://www.example.com/process33748449/interface"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <message name="PurchaseOrder">
        <part name="CustomerOrderNumber" type="xsd:string"/>
    </message>
</definitions>
```
3. To complete our abstract process definition, we will edit the BPEL file for Protocol.

**Tip:** A WSDL definition may be created quickly by pasting the corresponding XML directly into the source window. The result can be confirmed by changing back to the graph window.
Open Protocol.bpel in the BPEL editor. Delete the default InputVariable, PartnerLink, and Reply invoke activity definitions. We will be creating the process definition shown in Figure 4-7.

![Graph view of BPEL file for Protocol in purchasing scenario](image)

Select the icon containing the process name Protocol and select the **Process is long-running** option in its definition, as shown in Figure 4-8.

![Process is long-running definition in purchasing scenario](image)

4. Create the partner links by dragging the ProtocolInterface.wsdl file from the tree in the Services view into the BPEL editor. As it is dropped, a dialog will appear allowing the port type to be selected. Repeat this operation for both BuyerPublic and SellerPublic port types.

**Tip:** If menus in the BPEL editor do not appear to be populated correctly, and all modified WSDL files have been saved, try exiting and re-entering the BPEL editor.
Since this process will be defined from the perspective of the seller, the role of its SellerPublic partner link will need to be reversed. Click \(<\rightarrow>\) as shown in Figure 4-9.

![SellerPublic partner link role reversal in purchasing scenario](image)

**Figure 4-9** SellerPublic partner link role reversal in purchasing scenario

5. Create the variables PurchaseOrder, PurchaseOrderConfirmation, DeliveryNote, Invoice, and Payment. Define each variable's type as the message definition of the same name in ProtocolInterface.wsdl, as shown in the example for PurchaseOrder shown in Figure 4-10.

![PurchaseOrder variable definition in purchasing scenario](image)

**Figure 4-10** PurchaseOrder variable definition in purchasing scenario

6. We will now create the required correlation sets. In our scenario, we have created the need for two correlation sets. The first is based on the Customer Order Number, and allows the buyer to match received Purchase Order Confirmations with their associated Purchase Orders. The second is based on the Purchase Order Number, and this is used throughout the rest of the process.

Create the CustomerOrder correlation set with property as shown in Figure 4-11 on page 66. The property CustomerOrderReference needs to be created by clicking **New** and completing the dialog as shown in Figure 4-12 on page 66.
Repeat this operation for the PurchaseOrder correlation set, aliasing it with the PurchaseOrderNumber part of messages that contain it. The completed message property is shown in the edit dialog shown in Figure 4-13 on page 67.
7. Having created the partner links, variables, and correlation sets required by the process, the activities can now be added to complete the definition that was shown in Figure 4-7 on page 64 above. Unless noted otherwise, use of a correlation set is without initiation.

**Tip:** When an icon in the BPEL editor is selected, clicking the icons in the call-out displayed allows certain parameters to be defined directly.

- Create the receive activity Receive Purchase Order with partner link SellerPublic, operation transferPurchaserOrder, request variable PurchaseOrder, and correlation set CustomerOrder (with initiation).
- Create the invoke activity Send Purchase Order Confirmation with partner link BuyerPublic, operation transferPurchaseOrderConfirmation, request variable PurchaseOrderConfirmation, and correlation sets CustomerOrder and PurchaseOrder (with initiation).
- Create the invoke activity Send Delivery Note with partner link BuyerPublic, operation transferDeliveryNote, request variable DeliveryNote, and correlation set PurchaseOrder.
- Create the invoke activity Send Invoice with partner link BuyerPublic, operation transferInvoice, request variable Invoice, and correlation set PurchaseOrder.
Create the receive activity Wait for Payment with partner link SellerPublic,
operation transferPayment, request variable Payment, and correlation set
PurchaseOrder.

When complete, save it.

**Tip:** If you get BPEL validation errors of the form BPED0103E: Validation
error: 'Multiple property alias definitions found... check for and
remove any duplicate extensibility elements in the associated WSDL files.

Note that an abstract process must not include a terminate activity.

The source view of the BPEL file for Protocol is shown in Example 4-3.

*Example 4-3  Source view of BPEL file for Protocol in purchasing scenario*

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process expressionLanguage="Java" name="Protocol"
  suppressJoinFailure="yes"
  targetNamespace="http://www.example.com/process73339203"
  wpc:autoDelete="no" wpc:autonomy="peer" wpc:businessRelevant="yes"
  wpc:compensationSphere="notSupported" wpc:displayName="Protocol"
  wpc:executionMode="longRunning" wpc:validFrom="2003-01-01T00:00:00"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/
  xmlns:wsdl="http://www.example.com/process73339203/interface"
  xmlns:wsdl0="http://www.example.com/process73339203"
  xmlns:wsdl1="http://www.example.com/process33748449/interface"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  http://schemas.xmlsoap.org/ws/2003/03/business-process/">
  <partnerLinks>
    <partnerLink name="BuyerPublic"
      partnerLinkType="wsdl0:BuyerPublicLT"
      partnerRole="BuyerPublicRole"/>
    <partnerLink name="SellerPublic"
      partnerLinkType="wsdl0:SellerPublicLT"/>
  </partnerLinks>
  <variables>
    <variable messageType="wsdl1:PurchaseOrder" name="PurchaseOrder"/>
    <variable messageType="wsdl1:PurchaseOrderConfirmation"
      name="PurchaseOrderConfirmation"/>
    <variable messageType="wsdl1:DeliveryNote" name="DeliveryNote"/>
    <variable messageType="wsdl1:Invoice" name="Invoice"/>
    <variable messageType="wsdl1:Payment" name="Payment"/>
  </variables>
  <correlationSets>
    <correlationSet name="CustomerOrder"
      properties="wsdl1:CustomerOrderReference"/>
```
<correlationSet name="PurchaseOrder"
properties="wsdl1:PurchaseOrderReference"/>
</correlationSets>
<sequence name="Sequence" wpc:displayName="Sequence" wpc:id="1073741825">
  <receive createInstance="yes" name="ReceivePurchaseOrder"
operation="transferPurchaseOrder" partnerLink="SellerPublic"
portType="wsdl1:SellerPublic" variable="PurchaseOrder"
wpc:displayName="Receive Purchase Order" wpc:id="2">
    <correlations>
      <correlation initiate="yes" set="CustomerOrder"/>
    </correlations>
  </receive>
  <invoke inputVariable="PurchaseOrderConfirmation"
name="SendPurchaseOrderConfirmation"
operation="transferPurchaseOrderConfirmation" partnerLink="BuyerPublic"
portType="wsdl1:BuyerPublic" wpc:displayName="Send Purchase Order Confirmation" wpc:id="3">
    <correlations>
      <correlation initiate="no" pattern="out" set="CustomerOrder"/>
      <correlation initiate="yes" pattern="out" set="PurchaseOrder"/>
    </correlations>
  </invoke>
  <invoke inputVariable="DeliveryNote" name="SendDeliveryNote"
operation="transferDeliveryNote" partnerLink="BuyerPublic"
portType="wsdl1:BuyerPublic" wpc:displayName="Send Delivery Note" wpc:id="4">
    <correlations>
      <correlation initiate="no" pattern="out" set="PurchaseOrder"/>
    </correlations>
  </invoke>
  <invoke inputVariable="Invoice" name="SendInvoice"
operation="transferInvoice" partnerLink="BuyerPublic"
portType="wsdl1:BuyerPublic" wpc:displayName="Send Invoice" wpc:id="5">
    <correlations>
      <correlation initiate="no" pattern="out" set="PurchaseOrder"/>
    </correlations>
  </invoke>
  <receive name="WaitforPayment1" operation="transferPayment"
partnerLink="SellerPublic" portType="wsdl1:SellerPublic"
variable="Payment" wpc:displayName="Wait for Payment" wpc:id="7">
    <correlations>
      <correlation initiate="no" set="PurchaseOrder"/>
    </correlations>
  </receive>
</sequence>
</process>
This completes the abstract process and Web service definitions representing
the public business protocol between buyer and seller.

In a real project situation, the Protocol.bpel and ProtocolInterface.wsdl files
produced at this point could be distributed to all parties as an agreement
baseline and to act as a starting point for the development of private
executable processes within each organization.

4.3.4 Building the private executable process

We now create the initial Buyer private executable process and edit its WSDL
file. In doing so, we directly use the WSDL file for the Protocol process produced
in the previous steps. This ensures that our private executable process conforms
to the port types, operations, messages, parts, and correlation sets defined
within the public abstract process.

1. Create a new BPEL flow-based business process in the package
   your_domain.buyer with file name Buyer. Open BuyerInterface.wsdl in the
   WSDL editor. Delete the default ProcessPortType and InputMessage
definitions as before. Create the imports, port types, operations, messages,
   and parts as shown in Figure 4-14.

   When complete, save it.

Figure 4-14  Graph view of WSDL file for Buyer in purchasing scenario
Note that the imported port types, operations, messages, and parts are displayed in a grey color. Since they belong to another WSDL file, they are read-only in this view.

The source view of the WSDL file for Buyer is shown in Figure 4-4.

Example 4-4  Source view of WSDL file for Buyer in purchasing scenario

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    targetNamespace="http://www.example.com/process86910947/interface"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
    xmlns:tns="http://www.example.com/process86910947/interface"
    xmlns:wsdl0="http://www.example.com/process33748449/interface"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <import location="../protocol/ProtocolInterface.wsdl"
        namespace="http://www.example.com/process33748449/interface"/>
    <message name="DeliveryReceipt">
        <part name="PurchaseOrderNumber" type="xsd:string"/>
    </message>
    <portType name="BuyerPrivate">
        <operation name="acceptDeliveryReceipt">
            <input message="tns:DeliveryReceipt"/>
        </operation>
        <operation name="processPurchaseOrder">
            <input message="wsdl0:PurchaseOrder"/>
        </operation>
        <operation name="makePayment">
            <input message="wsdl0:Payment"/>
        </operation>
    </portType>
    <plnk:partnerLinkType name="PartnerLinkType">
        <plnk:role name="ProcessRole">
            <plnk:portType name="tns:ProcessPortType"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

2. Before we edit the BPEL file for the Buyer process, we need to create the Web services definitions for the GoodsReceiving and AccountsPayable roles identified in our scenario. We will create them as empty services because we will not be concerned as to their implementation in this exercise.

Create a new empty service in the `your_domain.buyer` package and name it GoodsReceiving. Open GoodsReceiving.wsdl in the WSDL editor and create the imports, port type, and operation as shown in Figure 4-15 on page 72. When complete, save it.
The source view of the WSDL file for Goods Receiving is shown in Example 4-5.

Example 4-5  Source view of WSDL file for Goods Receiving in purchasing scenario

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="GoodsReceiving"
    targetNamespace="http://Buyer/GoodsReceiving"
    xmlns="http://schemas.xmlsoap.org/wsdl/
    xmlns:tns="http://Buyer/GoodsReceiving"
    xmlns:wsdl="http://www.example.com/process33748449/interface"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <import location="../protocol/ProtocolInterface.wsdl"
        namespace="http://www.example.com/process33748449/interface"/>
    <portType name="GoodsReceiving">
        <operation name="validateDeliveryNote">
            <input message="wsdl:DeliveryNote"/>
        </operation>
    </portType>
</definitions>
```

Similarly, create another empty service in the same package and name it AccountsPayable. Open AccountsPayable.wsdl in the WSDL editor and create the imports, port type, and operation as shown in Figure 4-16 on page 73.

When complete, save it.
Chapter 4. Key technologies

Figure 4-16  Graph view of WSDL file for AccountsPayable in purchasing scenario

The source view of the WSDL file for AccountsPayable is shown in Example 4-6.

Example 4-6  Source view of WSDL file for AccountsPayable in purchasing scenario

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="AccountsPayable"
    targetNamespace="http://Buyer/AccountsPayable"
    xmlns="http://schemas.xmlsoap.org/wsdl/
    xmlns:tns="http://Buyer/AccountsPayable"
    xmlns:wsdl="http://www.example.com/process33748449/interface"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <import location="../protocol/ProtocolInterface.wsdl"
        namespace="http://www.example.com/process33748449/interface"/>
    <portType name="AccountsPayable">
        <operation name="settleInvoice">
            <input message="wsdl:Invoice"/>
        </operation>
    </portType>
</definitions>
```

3. To complete our executable process definition, we will edit the BPEL file for Buyer.

Open Buyer.bpel in the BPEL editor. Delete the default InputVariable, PartnerLink, and Reply invoke activity definitions. We will be creating the process definition as shown in Figure 4-17 on page 74.
Select the icon containing the process name Buyer and select the **Process is long-running** option in its definition.

4. Creating the BuyerPublic and SellerPublic partner links by dragging the ProtocolInterface.wsdl file from the tree in the Services view into the BPEL editor was done for the Protocol process. In the same way, create the BuyerPrivate partner link from BuyerInterface.wsdl, GoodsReceiving partner link from GoodsReceiving.wsdl, and AccountsPayable partner link from AccountsPayable.wsdl.

Since this BPEL is defined from the Buyer process perspective, reverse the roles of the BuyerPublic and BuyerPrivate partner links.

5. Create the variables PurchaseOrder, PurchaseOrderConfirmation, DeliveryNote, Invoice, and Payment and define each variable's type as the message definition of the same name in ProtocolInterface.wsdl.
Create the variable DeliveryReceipt and define it as the message definition of the same name in BuyerInterface.wsdl. We introduced the Delivery Receipt document as an example of a message defined and used only within the scope of the Buyer private executable process. In a real project situation, many message of this scope would be expected.

6. We will now create the correlation sets. The message properties required for the CustomerOrder and PurchaseOrder correlation sets are already defined in the Protocol public abstract process, so we can simply reference them. The message properties required for the DeliveryReceipt correlation set will need to be created in this step.

Create the CustomerOrder correlation set and assign it the CustomerOrderReference message property from ProtocolInterface.wsdl. Clicking Browse is used to do this rather than New. Repeat this operation to create the PurchaseOrder correlation set with the PurchaseOrderReference message property.

Create the DeliveryReceipt correlation set, this time clicking New to create the message property DeliveryReceiptReference in BuyerInterface.wsdl, as shown in Figure 4-18.

Figure 4-18 DeliveryReceiptReference message property creation in purchasing scenario

7. Now that we have created the partner links, variables, and correlation sets required by the process, the activities can be added to complete the definition
that was shown in Figure 4-17 on page 74 above. Unless noted otherwise, use of a correlation set is without initiation.

- Create the receive activity Receive Purchase Order processing request with partner link BuyerPrivate, operation processPurchaseOrder, request variable PurchaseOrder, and correlation set CustomerOrder (with initiation).

- Create the invoke activity Send Purchase Order to Seller with partner link SellerPublic, operation transferPurchaserOrder, request variable PurchaseOrder, and correlation set CustomerOrder.

- Create the receive activity Wait for Purchase Order Confirmation from Seller with partner link BuyerPublic, operation transferPurchaseOrderConfirmation, request variable PurchaseOrderConfirmation, and correlation sets CustomerOrder and PurchaseOrder (with initiation).

- Create the receive activity Wait for Delivery Note from Seller with partner link BuyerPublic, operation transferDeliveryNote, request variable DeliveryNote, and correlation set PurchaseOrder.

- Create the flow.

- Create the sequence Match Delivery Note to delivered goods.

- Create the invoke activity Send Delivery Note to Goods Receiving with partner link GoodsReceiving, operation validateDeliveryNote, request variable DeliveryNote, and correlation set DeliveryReceipt (with initiation).

- Create the receive activity Wait for Delivery Receipt from Goods Receiving with partner link BuyerPrivate, operation acceptDeliveryReceipt, request variable DeliveryReceipt, and correlation set DeliveryReceipt.

- Create the receive activity Wait for Invoice from Seller with partner link BuyerPublic, operation transferInvoice, request variable Invoice, and correlation set PurchaseOrder.

- Create the invoke activity Send Invoice to Accounts Payable with partner link AccountsPayable, operation settleInvoice, request variable Invoice, and correlation set PurchaseOrder.

- Create the receive activity Wait for Payment release from Accounts Payable with partner link BuyerPrivate, operation makePayment, request variable Payment, and correlation set PurchaseOrder.

- Create the invoke activity Send Payment to Seller with partner link SellerPublic, operation transferPayment, request variable Payment, and correlation set PurchaseOrder.

When complete, save it. You will be notified of any validation errors.

The source view of the BPEL file for Buyer is shown in Example 4-7.
Example 4-7  Source view of BPEL file for Buyer in purchasing scenario

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process expressionLanguage="Java" name="Buyer"
    suppressJoinFailure="yes"
    targetNamespace="http://www.example.com/process60219238"
    wpc:autoDelete="no" wpc:autonomy="peer" wpc:businessRelevant="yes"
    wpc:compensationSphere="notSupported" wpc:displayName="Buyer"
    wpc:executionMode="longRunning" wpc:validFrom="2003-01-01T00:00:00"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/
    xmlns:wsdl="http://www.example.com/process60219238/interface"
    xmlns:wsdl0="http://www.example.com/process60219238"
    xmlns:wsdl1="http://www.example.com/process33748449/interface"
    xmlns:wsdl2="http://Buyer/GoodsReceiving"
    xmlns:wsdl3="http://Buyer/AccountsPayable"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    http://schemas.xmlsoap.org/ws/2003/03/business-process/"
    <partnerLinks>
        <partnerLink name="SellerPublic"
            partnerLinkType="wsdl0:SellerPublicLT"
            partnerRole="SellerPublicRole"/>
        <partnerLink myRole="BuyerPublicRole" name="BuyerPublic"
            partnerLinkType="wsdl0:BuyerPublicLT"/>
        <partnerLink myRole="BuyerPrivateRole" name="BuyerPrivate"
            partnerLinkType="wsdl0:BuyerPrivateLT"/>
    </partnerLinks>
    <variables>
        <variable messageType="wsdl1:PurchaseOrder" name="PurchaseOrder"/>
        <variable messageType="wsdl1:PurchaseOrderConfirmation"
            name="PurchaseOrderConfirmation"/>
        <variable messageType="wsdl1:DeliveryNote" name="DeliveryNote"/>
        <variable messageType="wsdl1:Invoice" name="Invoice"/>
        <variable messageType="wsdl1:Payment" name="Payment"/>
        <variable messageType="wsdl1:DeliveryReceipt" name="DeliveryReceipt"/>
    </variables>
    <correlationSets>
        <correlationSet name="CustomerOrder"
            properties="wsdl1:CustomerOrderReference"/>
        <correlationSet name="PurchaseOrder"
            properties="wsdl1:PurchaseOrderReference"/>
    </correlationSets>
```
<correlationSet name="DeliveryReceipt" properties="wsdl:DeliveryReceiptReference"/>
</correlationSets>

<scope name="Scope" wpc:businessRelevant="no" wpc:id="21">
  <sequence joinCondition="DefinedByJavaCode" name="Sequence" wpc:displayName="Sequence" wpc:id="1073741825">
    <wpc:joinCondition>
      <wpc:false/>
    </wpc:joinCondition>
    <receive createInstance="yes" name="ReceivePurchaseOrderprocessingrequest" operation="processPurchaseOrder" partnerLink="BuyerPrivate" portType="wsdl:BuyerPrivate" variable="PurchaseOrder" wpc:displayName="Receive Purchase Order processing request" wpc:id="2">
      <correlations>
        <correlation initiate="yes" set="CustomerOrder"/>
      </correlations>
    </receive>
    <invoke inputVariable="PurchaseOrder" name="SendPurchaseOrdertoSeller" operation="transferPurchaseOrder" partnerLink="SellerPublic" portType="wsdl1:SellerPublic" wpc:displayName="Send Purchase Order to Seller" wpc:id="4">
      <correlations>
        <correlation initiate="no" pattern="out" set="CustomerOrder"/>
      </correlations>
    </invoke>
    <receive name="WaitforPurchaseOrderConfirmationfromSeller" operation="transferPurchaseOrderConfirmation" partnerLink="BuyerPublic" portType="wsdl1:BuyerPublic" variable="PurchaseOrderConfirmation" wpc:displayName="Wait for Purchase Order Confirmation from Seller" wpc:id="5">
      <correlations>
        <correlation initiate="no" set="CustomerOrder"/>
        <correlation initiate="yes" set="PurchaseOrder"/>
      </correlations>
    </receive>
    <receive name="WaitforDeliveryNotefromSeller" operation="transferDeliveryNote" partnerLink="BuyerPublic" portType="wsdl1:BuyerPublic" variable="DeliveryNote" wpc:displayName="Wait for Delivery Note from Seller" wpc:id="6">
      <correlations>
        <correlation initiate="no" set="PurchaseOrder"/>
      </correlations>
    </receive>
  </sequence>
</scope>
<correlations>
</correlations>
</receive>
<flow joinCondition="DefinedByJavaCode" name="Flow"
    wpc:displayName="Flow" wpc:id="16">
    <wpc:joinCondition>
        <wpc:any/>
    </wpc:joinCondition>
    <sequence name="MatchDeliveryNotetodeliveredgoods"
        wpc:displayName="Match Delivery Note to delivered goods"
        wpc:id="17">
        <invoke inputVariable="DeliveryNote"
            name="SendDeliveryNotetoGoodsReceiving"
            operation="validateDeliveryNote"
            partnerLink="GoodsReceiving"
            portType="wsdl2:GoodsReceiving"
            wpc:displayName="Send Delivery Note to Goods Receiving"
            wpc:id="7">
            <correlations>
                <correlation initiate="yes" pattern="out"
                    set="DeliveryReceipt"/>
            </correlations>
        </invoke>
        <receive name="WaitforDeliveryReceiptfromGoodsReceiving"
            operation="acceptDeliveryReceipt"
            partnerLink="BuyerPrivate"
            portType="wsdl:BuyerPrivate"
            variable="DeliveryReceipt"
            wpc:displayName="Wait for Delivery Receipt from Goods Receiving"
            wpc:id="8">
            <correlations>
                <correlation initiate="no" set="DeliveryReceipt"/>
            </correlations>
        </receive>
    </sequence>
</flow>
<invoke inputVariable="Invoice"
    name="SendInvoicetoAccountsPayable1"
    operation="settleInvoice" partnerLink="AccountsPayable"
    portType="wsdl3:AccountsPayable"
This completes the definition of the purchasing scenario. The key points to take from this exercise are as follows:

- The process of developing an electronic commerce integration scenario starts with a high-level view of the process and the respective needs of the participants in the process.

- A narrative definition of the process can be translated to a UML sequence diagram (or a LOVEM or swim lane chart) to provide a more analytical view of the process. Teams may prefer to initially capture process definitions using graphical approaches.

- UML sequence diagrams can be directly used to identify the public abstract processes, private executable processes, and Web services required to implement a collaboration.

- WSDL files produced from the definition of public abstract processes can be directly imported into private executable process definitions to define the
public port types, operations, messages, parts, and message properties. This significantly reduces the risk of implementation misalignment between business partners.

- Abstract process definitions may be used in the future for other valuable purposes such as providing run-time business protocol conformance validation, creating skeletons for executable processes, and perhaps even executable process code generation.

- BPEL validation errors to do with correlation sets can be difficult to resolve. Check for and remove any duplicate extensibility elements in the associated WSDL files.

- Although only two business partners have been included in this scenario, the same approach can be applied to any number of participants. With a greater number, it may be necessary to restrict even the initial process definition to public abstract processes only.

- There are many BPEL and WSDL aspects that have not been presented in this exercise. What has been presented is intended as a starting point.
Using WebSphere Studio

This chapter provides an introduction to using WebSphere Studio Application Developer Integration Edition, with particular emphasis on the tools used when working with the BPEL process. For a more detailed look at WebSphere Studio Application Developer Integration Edition see the redbook *WebSphere Business Integration Server Foundation V5.1*, SG24-6318.
5.1 WebSphere Studio

WebSphere Studio is a comprehensive development environment designed to meet all of your development needs—from state-of-the-art Web interfaces to server-side applications, from individual development to advanced team environments, from Java development to application integration. Available in a number of configurations, with extensions from IBM and partners, WebSphere Studio enables developers to use a single development environment designed to meet their specific development needs.

WebSphere Studio configurations are all built on the WebSphere Studio Workbench, which extends the open-source Eclipse platform and provides an open, extensible plug-in architecture. Numerous plug-ins are available from partners and the open source community. Or, using the included plug-in development environment, you can create your own plug-ins for specific needs.

The number of experienced users of WebSphere Studio continues to increase, and WebSphere Studio has won many awards as an integrated development environment for Java, Web services, and XML.

5.1.1 Highlights and benefits

WebSphere Studio is designed to be used by a wide variety of roles, such as Web developers, Java developers, enterprise programmers, business analysts, and system architects. From one development environment you can develop, test, deploy, and manage applications. A rich set of utilities and wizards helps simplify common tasks so that developers can concentrate on providing true business value and on rapidly getting robust applications in production.

Rapid Java and J2EE development

WebSphere Studio is specifically designed to make developing Java and J2EE applications as simple and streamlined as possible. Whether the programmer is a novice, experienced in Java and J2EE, or more familiar with legacy programming, WebSphere Studio provides the tools, editors, visual framework, and underlying services that allow rapid and efficient programming.

Open standards

All products in the WebSphere Studio suite are built on open standards, and the code that they generate complies with open standards. You can build and deploy state-of-the-art, server-side applications that meet the Java Servlets 2.3, JavaServer Pages (JSP) 1.2, and Enterprise JavaBeans™ (EJB) 2.0 specifications. Tools support current XML, Web services, and Unified Modeling Language (UML) standards. The open source Concurrent Versions System
(CVS) is included as a source configuration management system in all products built on WebSphere Studio Workbench.

The Eclipse platform is an open source project that allows you to easily create and integrate your own tools into the environment.

**Vertical and horizontal tool integration**
Traditionally, software vendors have provided vertical tools, forcing customers to do their own integration. Eclipse provides a platform that IBM, customers, and Independent Software Vendors can easily extend. Vendors have embraced this technology and are continuing to actively build tools on this foundation.

The WebSphere Studio product that is built on Eclipse offers tools that are already seamlessly integrated, freeing you to focus on building applications rather than interfacing or bridging between tools used by different roles (business analyst, graphic artist, Java programmer, tester, COBOL programmer, and others).

**Role-based development with consistent look and feel**
Each configuration of WebSphere Studio is designed to support particular e-business development roles. Roles are supported by task-oriented perspectives that combine views and editors of project resources. Perspectives filter out complexity, and present the developer with only those functions that are relevant to the task at hand.

A Web page designer, for example, has easy access to the tools required for that job and is not forced to search for them in an interface cluttered with tools irrelevant to the task. A Java developer, in contrast, works most often in the Java perspective. Developers can easily switch perspectives with a single click. Perspectives can be customized to let developers work according to their preferences, and perspectives can be shared to adapt the environment to enterprise-specific roles.

**Programming performance and productivity**
The common look and feel provided by WebSphere Studio across project roles and perspectives reduces the learning curve for new users and increases user productivity.

Team productivity and application quality improve through greater automation of the development and deployment process via an integrated development environment. Many productivity enhancing features are included in all configurations, such as:

- A tool to visually manage a Web site.
- Wizards that generate specification-compliant artifacts.
- Automatic updates to Web links as you move resources around.
- Rich media tools that enhance images and create JavaScript to add visual interest to Web sites.
- Built-in refactoring tools that make it easy to reorganize Java classes.
- An integrated unit test environment with hot deployment capabilities.
- Profiling tools that help optimize applications by detecting and analyzing performance problems.

**Multiple software configuration management repositories**

WebSphere Studio supports multiple software configuration management repositories (SCM). In addition to providing plug-ins that support Concurrent Versions Systems (CVS) and Rational ClearCase® LT, the open Eclipse framework allows you to add adapters available from the open source community, IBM, and other commercial vendors.

**Support for multiple target run-time environments**

Developers can develop applications for multiple target run-time environments on multiple platforms. Java Development Kits (JDK) from IBM and other vendors can be used in development.

**Support for Struts framework Version 1.1**

WebSphere Studio provides support for the latest release of the open source Struts framework for Web applications, which exploits the model-view-controller (MVC) design pattern. An integrated visual assembly tool and wizards accelerate Web development projects.

### 5.1.2 Configurations

There are four configurations of WebSphere Studio:

- Site Developer
- Application Developer
- Application Developer Integration Edition
- Enterprise Developer

Figure 5-1 on page 87 shows the user roles, main functions, and target run times of the four configurations. The base user interface and environment functions are provided by the Eclipse platform, along with frameworks to allow tool builders to develop plug-ins.
Each configuration includes the WebSphere Studio Workbench plus a set of additional plug-ins that provide increasing levels of development capability, starting with Site Developer and moving up to Enterprise Developer. For example, Application Developer includes all of Site Developer functions plus additional plug-ins that provide functions such as Enterprise JavaBean support.

**Figure 5-1 WebSphere Studio family**

**WebSphere Studio Site Developer**
Site Developer is designed for the entire Web development team—content authors, graphic artists, Web programmers, and Web masters—to make it easy to collaboratively design, create, assemble, publish, deploy, and maintain dynamic, interactive Web applications that exploit today’s open technology standards.

You can quickly build and test business logic, and enhance presentation artifacts with the built-in Web creation tools, before deploying on a production server such as WebSphere Application Server or Apache Tomcat.
This configuration includes an intuitive WYSIWYG editor that lets novice Web designers create and publish Web sites while incorporating the latest Web technology, including JavaScript, dynamic HTML, and Cascading Style Sheets.

Visual layout tools are provided to create dynamic Web sites with Java servlet or JavaServer Pages (JSP) components. Included is a built-in XML development environment, along with Web services tools for the creation, deployment, and publishing of Web services.

Struts tools assist in the creation of well-architected Web applications, while support for JavaServer Faces simplifies the construction of Web interfaces and Web applications. Procedural programmers will find Java application development easy with the EGL tools.

Site Developer also provides an interface to CVS for version control.

Note that a slightly modified version of Site Developer is packaged with WebSphere Application Server - Express. The modifications are designed to tailor the product for the Express environment.

The home page for WebSphere Studio Site Developer is at:

http://www.ibm.com/software/awdtools/studiositedev/

**WebSphere Studio Application Developer**

Application Developer is designed for professional developers of Java applications conforming to Java 2 Platform, Enterprise Edition (J2EE) specifications, which require integrated Java, Web, XML, and Web services support. It includes all of the features of Site Developer, and adds tools for developing Enterprise JavaBean (EJB) applications, as well as performance profiling and logging tools for both local and remote execution.

The integrated UML Visual Editor for Java and EJB allows application development and object modeling based on the Unified Modeling Language (UML), an industry standard for object modeling.

In addition to the CVS interface tools, Rational ClearCase LT is provided for version control.

WebSphere Studio Application Developer includes the IBM Portal Toolkit, which enables developers to create, edit, and test portlets that comply with the JSR 168 architecture and with the IBM Portlet API (an extension of JSR 168). Portlets can be created using the Struts framework and can include JavaServer Faces UI components.
The target run-time environment is a J2EE server, such as the WebSphere Application Server base and Network Deployment configurations, to take full advantage of EJB capabilities. Using the performance profiling and tracing tools, it is possible to detect application performance bottlenecks early in the development cycle. The built-in test environment for WebSphere Application Server and advanced tools for code generation help shorten the test cycle.

The home page for WebSphere Studio Application Developer is at:
http://www.ibm.com/software/awdtools/studioappdev/

**WebSphere Studio Application Developer Integration Edition**
Application Developer Integration Edition includes all of the features of Application Developer, plus the following tools for integration with back-end systems:

- **Enterprise services**
  
  A set of tools and wizards that facilitate service-oriented development. Included are tools to provide support for consuming services from and providing services to WebSphere Business Integration Server Foundation, which acts as the point of integration for a wide variety of services.

  Various types of resources are consumable, such as Simple Access Object Protocol (SOAP) Web services, Java beans, stateless session (EJB) beans, and J2EE Connector Architecture (JCA) services. Services deployed into the WebSphere Application Server can be provided as SOAP services via the EJB programming model.

  Enterprise services are based on open standards such as J2EE, JCA, Web Services Description Language (WSDL), Web Services Invocation Framework (WSIF), and Extensible Stylesheet Language Transforms (XSLT).

- **Graphical business process composition tools**

  A set of tools that enable you to visually compose a service out of one or more existing services, define and transform the flow of information between services, and create a business process that contains other nested processes.

- **Development connectors**

  For integration with back-end systems, Application Developer Integration Edition provides development connectors for Customer Information Control System (CICS®), Information Management System (IMS™), and Host-on-Demand (HOD). IBM provides a range of adapters for popular packaged applications including Customer Relationship Management, Supply Chain Management, Enterprise Resource Planning, and mainframe applications.
Enterprise services support

Support for the full set of Enterprise services provided by WebSphere Business Integration Server Foundation.

The target application server for Application Developer Integration Edition is WebSphere Business Integration Server Foundation.

The home page for WebSphere Studio Application Developer Integration Editions is at:

http://www.ibm.com/software/awdtools/studiointegration/

WebSphere Studio Enterprise Developer

Enterprise Developer provides enterprise-level development tools for interacting with back-end development and operating systems. In addition to Application Developer Integration Edition features, Enterprise Developer provides the following:

- **z/OS® application development tools** in an interactive, workstation-based environment where you can do the following tasks:
  - Perform MVS™-based system tasks such as managing datasets with a workstation-like directory structure and submitting JCL or TSO commands.
  - Create HLASM, COBOL, or PL/I code, including CICS, IMS, and SQL statements.
  - Edit, test, and compile the source code locally or remotely on a z/OS system.
  - Transfer CLISTs and REXX EXECs scripts to z/OS, run them, and view the resulting output.

- **Enterprise Generation Language (EGL) tools** that provide a common language for implementing business processes that target a variety of run-time platforms. EGL lets traditional enterprise developers rapidly deliver enterprise-wide data to browsers, regardless of the depth of their Web or object-oriented development expertise. In addition to the EGL code generation support provided by Site Developer and Application Developer that produces Java code, Enterprise Developer provides EGL code generation support (via a value add plugin) that produces COBOL code.

- **Integration with existing transactional environments** such as CICS and IMS.

- **Tools for creating new Enterprise Information System (EIS) applications** as part of a J2EE application.

- **The ability to integrate Struts-based model-view-controller applications** using J2EE Connector Architecture connectors and EGL.
Integration with WebSphere Studio Asset Analyzer (WSAA) to identify application processes and connecting points and providing the ability to generate components from existing code.

The target run-time environment is WebSphere Application Server Enterprise or WebSphere Application Server for z/OS.

The home page for WebSphere Studio Enterprise Developer is at:
http://www.ibm.com/software/awdtools/studioenterprisedev/

Application development for iSeries™
WebSphere Studio can build and deploy applications to the WebSphere Application Server on iSeries and all other supported platforms. For iSeries customers, IBM builds upon its strategic development tools and adds iSeries-specific features that help integrate Java applications with existing iSeries resources.

- WebSphere Development Studio for iSeries
  Prior to 2001, customers were required to purchase iSeries host-based development tools (ILE RPG, ILE COBOL, ILE C, and ILE C++) and client-side development tools separately. In May 2001, IBM bundled both the host-based and certain client-side tools into a single offering called WebSphere Development Studio for iSeries. iSeries customers with an active software subscription for any of the host-based development tools can order a no-charge upgrade to WebSphere Development Studio. Better yet, they also are provided with unlimited licenses of the client-based tools. This offering is only available to iSeries customers. Therefore, all entitled customers should have WebSphere Development Studio deployed in their environment.

- WebSphere Development Tools for iSeries
  The client-side development tools bundled within the initial offering of WebSphere Development Studio were called WebSphere Development Tools for iSeries. Available for customers deployed on OS/400® V4R5M0 through V5R1M0, WebSphere Development Tools included VisualAge® for Java, Professional Edition, WebSphere Studio Professional Edition, VisualAge RPG, the IBM Distributed Debugger, CODE, and the IBM WebFacing Tool. These products were enhanced with iSeries-only features and functions.

- WebSphere Development Studio Client for iSeries
  In June 2002, IBM refreshed the client-side tools within WebSphere Development Studio. Building upon WebSphere Studio Site Developer, IBM offers iSeries customers a replacement for WebSphere Development Tools called WebSphere Development Studio Client for iSeries. Like WebSphere Development Tools for iSeries, WebSphere Development Studio Client for
iSeries extends IBM strategic development tools and adds iSeries-specific value. And, for customers with active software subscriptions to WebSphere Development Studio, it comes with an unlimited license entitlement. Also, a single license of WebSphere Development Studio Client for iSeries is provided with the purchase of WebSphere Application Server - Express for iSeries.

WebSphere Development Studio Client for iSeries includes:
- IBM WebFacing Tool for Web enabling 5250-based applications
- Remote System Explorer (RSE), which provides integrated, workstation-based iSeries development functions
- Web tools
- Java tools
- Web Services tools
- Database tools
- XML tools
- CODE (classic version)
- VisualAge RPG (classic version)
- Distributed Debugger (classic version)

The current version of WebSphere Development Studio Client for iSeries included within WebSphere Development Studio is Version 4. WebSphere Development Studio Client for iSeries Version 4 can deploy applications to all editions of both WebSphere Application Server Version 4.0 for iSeries and WebSphere Application Server Version 5 for iSeries.

### 5.1.3 Packaging

Table 5-1 shows the components included in the packaging of each WebSphere Studio product.

| Table 5-1   WebSphere Studio configuration packaging |
|-------------|--------------------------------------------------|
| **Adapter for Rational® ClearCase LT** | WSSD V5.1.2 | WSAD V5.1.2 | WSAD-IE V5.1 | WSED V5.1.1 | WS Client for iSeries |
| **IBM Agent Controller** | X | X | X | X | X |
| **WebSphere Application Server for Developers V5.1** | X | X | X | X | X |
| **Application Clients 5.1** | X | X | | | |

*The adapter for CVS is included in each configuration automatically.*
## 5.1.4 Platform support

All WebSphere Studio configurations can run on multiple development platforms and use multiple Java Development Kits (JDK).

<table>
<thead>
<tr>
<th>Development platforms</th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site Developer 5.1.2</td>
<td>Windows 2000, XP</td>
<td>Red Hat 7.2, 8.0</td>
</tr>
<tr>
<td></td>
<td>SuSe 7.2, 8.1</td>
<td></td>
</tr>
</tbody>
</table>

*The adapter for CVS is included in each configuration automatically.*
<table>
<thead>
<tr>
<th>Application Developer 5.1.2</th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2000, XP</td>
<td></td>
<td>Red Hat 7.2, 8.0 SuSe 7.2, 8.1</td>
</tr>
</tbody>
</table>

For the exact platform levels and requirements, refer to the following:

- Requirements for WebSphere Studio Site Developer:

- Requirements for WebSphere Studio Application Developer:

- Requirements for WebSphere Studio Application Developer Integration Edition:

- Requirements for WebSphere Studio Enterprise Developer:

WebSphere Studio is shipped with JDK 1.3 and 1.2. Java development projects can use either JDK level. Other JDKs can be installed in the workbench.

Applications can be developed for multiple run-time platforms. Depending on the configuration of WebSphere Studio you are using, you can develop applications for Enterprise Information Systems such as CICS and IMS, Windows operating systems with various Java Runtime Environments (JRE), Linux operating systems with various JREs, and AIX up to Version 5.2.

Several Java Runtime Environments can be used in the WebSphere Test Environment. You can install and use additional JREs as well.

### 5.2 What is new in WebSphere Studio V5.1

This release of WebSphere Studio Application Developer Integration Edition V5.1 brings a major change to the process editor and the architecture it is based on. It also contains some enhancements to the product that adds some new functions to the product and brings greater ease of use to you. The following sections list specifically what is new and what has changed.
5.2.1 New functions and features

The new functions and features are:

- A new visual process editor based on the Business Process Execution Language (BPEL) open source specification.
- A new process debugger for BPEL processes.
- A migration wizard to migrate your Flow Description Markup Language (FDML) processes to BPEL-based ones.
- The new process editor also lets you work with FDML processes, thereby offering coexistence with the previous FDML processes.
- New inbound bindings increase the range of deployment options for services created by Application Developer Integration Edition. The new bindings are IBM Web Services and IBM Web Services over JMS.
- A new service proxy has been added to align with the new bindings. It is the Java API for XML-based RPC (JAX-RPC).
- A new WSDL editor has been added, simplifying user interaction and adding clarity to the relationship of the components.
- Viewlets have been added to the help. Viewlets are indicated by the text Show Me, and that is what they do: Show you a wizard in action.
- New cheat sheets have been added that help you develop services by guiding you through the set of wizards you need to complete your tasks.
- New format handler help has been added to clarify the details of format handling.

5.2.2 Changes in function and features

Most areas of the product have had small changes to add greater ease of use. If you are familiar with the wizards of 5.0, you will see a more natural interaction with them in 5.1. Here are some of the areas that have had changes to them:

- The test environment user interface has changed slightly.
- The navigation has improved. You will find that it aligns now with your goals.
- The deployment information has been expanded to cover this complex area.
- An improved deploy code generation user interface has been created.
5.3 Installing V5.1

To install Version 5.1:

1. To start the install WebSphere Studio Application Developer Integration Edition, run launchpad.exe from disk 1 of the CD package. After a little while you should see a screen like Figure 5-2.

![Launchpad](image)

Figure 5-2   Launchpad

2. Click Install IBMWebSphere Studio Application Developer Integration Edition, which will take you to the install shield welcome screen shown in Figure 5-3 on page 97.
3. Click **Next** to be taken to the licence agreement screen shown in Figure 5-4.

4. Read the licence agreement and if you agree click **Next** to continue on to the installation directory screen shown in Figure 5-5 on page 98.
5. You can install into anywhere you like. The default directory is a good choice. Click **Next** when you are ready to be taken to the feature install screen, as in Figure 5-6.
6. For the sample application we only need the following selected:
   – Integrated Development Environment
   – Integrated Test Environments
   – WebSphere Business Integration Server Foundation 5.1

   You can select other options but they won't be covered here.

   Click **Next** to be taken to the Summary page shown in Figure 5-7.

![Figure 5-7 Summary](image)

7. If you need to make changes you can click the **Back** button, but if you are ready to start the install click **Next**.
8. You will now see a progress screen of the installation similar to Figure 5-8. Depending on how many feature were selected for install, this could take some time.

9. When the installation is complete you will see a screen like Figure 5-9 on page 101. Click Finish to complete the installation.
5.4 Business integration perspective

The default perspective in WebSphere Studio Application Developer Integration Edition is the business integration perspective. This should really come as no surprise; as the name of the product suggests, the main purpose of this version of Studio is to build enterprise integration applications.

This perspective is only available in this edition of the Studio family, and as such we will discuss it in this redbook. We mention and briefly describe aspects of other perspectives, but for a more detailed discussion you should see the WebSphere Studio product documentation.

Figure 5-10 on page 102 shows the business integration perspective as it appears when started for the first time. You can see that it initially contains the welcome screen to WebSphere Studio Application Developer Integration Edition, and this screen is a good place to start for more detailed information than will be contained in this chapter.
5.4.1 Views

A perspective includes a number of views. Some of the important views provided on the business integration perspective include the following.

Services view

As the name suggests, this view displays the service resources in your project. There are two parent folders with this view:

- Service projects
  
  This folder holds the service definitions as well as supporting files. As you can see in Figure 5-11 on page 103, this folder holds such things as the business process (the .bpel file), as well as the accompanying WSDL and Java files.
Deployable services

As the name suggests, this folder holds a view of the resources that are deployable within this enterprise application.

Package explorer view

This view is good for seeing the Java-specific resources associated with the project. This is the best view to use if you want to work manually on the Java source files. Figure 5-12 on page 104 is an example of this view.
Figure 5-12  Package explorer view

J2EE hierarchy view

This view provides a hierarchical view of the content models. This is the view where you would go to work on Enterprise Archive (EAR) files, for example, to change the deployment descriptor of the enterprise application. A sample if this view can be seen in Figure 5-13.
Other views
The other views in this perspective are standard views throughout WebSphere Studio, and details can be found in the online help. The only other two views that we are interested in are the WDSL editor and the BPEL editor. We deal with these later in the chapter.

5.4.2 Tools on the toolbar

There are many functions and features available within the business integration perspective that are easily available through the numerous icons on the toolbars. Table 5-3 lists the icons and what they do, for the features that are particular to creating integration services. The other icons are common among other perspectives and their behavior can be found in the online help.

Table 5-3  Business integration icons in the business integration perspective

<table>
<thead>
<tr>
<th>Toolbar Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Open business integration perspective.</td>
<td>Brings the business integration perspective to the foreground if multiple perspectives are open.</td>
</tr>
<tr>
<td></td>
<td>Create service project.</td>
<td>Creates a new service project.</td>
</tr>
<tr>
<td></td>
<td>Create empty service.</td>
<td>Creates an new WSDL file of an empty service.</td>
</tr>
<tr>
<td></td>
<td>Create service interface.</td>
<td>Creates a new WSDL file that contains port types and message definition.</td>
</tr>
<tr>
<td></td>
<td>Create java skeleton.</td>
<td>Generate an enterprise service and java skeleton based on an interface service definition.</td>
</tr>
<tr>
<td></td>
<td>Create EJB skeleton.</td>
<td>Generate an enterprise service and EJB skeleton based on an interface service definition.</td>
</tr>
<tr>
<td></td>
<td>Create java service.</td>
<td>Create an enterprise service from a java class.</td>
</tr>
<tr>
<td></td>
<td>Create EJB service.</td>
<td>Create an enterprise service from an EJB.</td>
</tr>
<tr>
<td></td>
<td>Create business process.</td>
<td>Create a BPEL (or FDML) process. Opens the appropriate editor.</td>
</tr>
</tbody>
</table>
5.4.3 Editors

As in other perspectives, there are numerous editors available that will be used depending on the type of file you are opening. There are two editors we discuss here that are really relevant to business integration:

- **WSDL editor**
  
  The WSDL editor has been updated in this release, with the purpose of making it easier to use. There is an overview on using the updated editor in “Using the WSDL editor” on page 106.

- **BPEL editor**

  The BPEL editor is a new editor that comes into this release with the support for BPEL. You will find an overview on using the new editor in “Using the BPEL editor” on page 119. The old FDML editor is still available and works the same way as the previous release, so it will not be discussed in this publication.

5.5 Using the WSDL editor

In this section we describe the elements of the editor and list the some of the elements in a WSDL. We show how to create and change these elements within the editor. It is assumed that the reader has basic knowledge of WSDLs and how they should be used, as the descriptions we provide of WSDL elements are quite brief. For an overview of WSDL see “Web Service Description Language (WSDL)” on page 43.
5.5.1 WSDL elements

The WSDL elements are:

- Service
  The service element is a collection of related ports and specifies the location of the service.

- Port
  A port defines an individual endpoint by specifying a single address for a binding. The port contains a 'binding' attribute that references a binding and an address element that provides a specification for the endpoint.

- Port type
  A port type is a named set of abstract operations and the abstract messages involved. Each operation refers to an input message and output messages. A port type is referenced by a binding object. Each binding references exactly one port type. Since each port refers to exactly one binding, each port has exactly one port type.

- Binding
  A binding is a top-level WSDL object that provides a concrete specification regarding the transmission of messages to and from a Web service. A binding references exactly one port type. The structure of a binding corresponds very closely to that of the port type. The binding contains extensibility elements (for example, soap, http, and mime) that specify protocol-specific details. Each port within a service references exactly one binding.

- Operation
  An operation names the operation and lists the expected inputs and outputs. The operation element may also contain a fault sub-element that describes any error data the operation may return.

- Message
  Messages represent an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system. WSDL messages are top-level objects that can be referenced by an operation's input, output, and fault elements (within a port type).

- Part
  All messages contain one or more parts. Parts are a flexible mechanism for describing the logical abstract content of a message. The message definition associates each part with a type using a message-typing attribute.
5.5.1 Import statement
The import statement is used to import other WSDL documents or XML schemas into your WSDL document. This enables you to have more modular WSDL documents. The use of the import statement allows the separation of the different elements of a service definition into independent documents, which can then be imported as needed. This technique helps in writing clearer service definitions by separating the definitions according to their level of abstraction.

5.5.2 WSDL editor views
The WSDL editor contains two views:

- Source view
In this view you can see the XML source code that is generated in the editor. You can directly edit the source code of the WSDL in this view.

- Graph view
This view is a graphical representation of the WSDL file, which allows you to create or change a WSDL without needing to edit the source XML directly. There are three different panes within this view and they are discussed in “Editor panes” on page 108.

In practice you generally use these two views in conjunction. The graph view is easy to use but sometimes it can be quicker to just edit the source code. Also if you are a little unfamiliar with the graph view, it can be easier to find some settings by glancing at the source.

5.5.3 Editor panes
Figure 5-14 on page 109 is an example of the graph view in the new WSDL editor.
You can see in the example that there are three panes associated with the editor:

**Elements pane**
This is the top pane and holds the graphical representation of the elements of the WSDL. We discuss this pane in more detail below.

**Details pane**
This is the bottom left-hand pane and generally shows the attributes of the selected element in the elements view. Only attributes from the active WSDL are able to be changed here; attributes from imported WSDLs must be changed in their respective files.

**Documentation pane**
This is the bottom right-hand pane and allows you to set and view documentation about the selected element.

**Elements pane**
The elements panes consist of seven containers, mostly corresponding to the elements of a WSDL previously described in “WSDL elements” on page 107.

- Imports
- Type
- Services
- Bindings
- Port Type
- Messages
User-defined types

User-defined types show the details for those types that are not built in. You will only see this container if there are user-defined types, and it will appear to the far right of the elements pane.

Imported files

When another file is imported, you will see the definitions from that file in the relevant container as if it was defined in the active WSDL file. You cannot update or change the elements from an imported file; they are essentially opened as read only.

In the editor you can tell the elements from an imported file because they are greyed out. This can be seen in Figure 5-15 where the only elements that are actually defined in the active WSDL are under the services container.

![Figure 5-15](import.png) Import elements are greyed out in the active WSDL

5.5.4 Tasks in the WSDL editor

This section describes some standard tasks you will need to perform within the WSDL editor. They are grouped by the element pane in which the task is generally done.

Port types

This section describes common tasks for port types.

Create port type

To create the port type:

1. Right-click in the port type container and choose Add Child → Port Type, as shown in Figure 5-16 on page 111.
2. Give the port type a name and click OK.
Add an operation to a port type

To add an operation to a port type:

1. Right-click the port type you want to add the operation to and choose Add Child → Operation.
2. Give the operation a name and click OK.

Add input/output/fault to an operation

To add input/output/fault to an operation:

1. Right-click the operation you want to choose Add Child → Input or Add Child → Output or Add Child → Fault.
2. If you choose fault, you must also supply a name for the fault then click OK.
**Associating a message to input/output/fault**

To associating a message to input/output/fault, right-click the input, output or fault you wish to associate a message to and choose **Set message**. You now have three choices:

- **Create a new message** - This creates a new message with the name you specify in the name field. The default name given will be `<operation name>Request` for an input, `<operation name>Response` for an output, and `<operation name>NewFault` for a fault. Take the default name or type in a different name, then click Finish.

- **Select an existing message** - This gives you a list of the messages currently available in the WSDL for you to choose from. Select a message and click Finish.

- **Import message from a file** - You must browse to a file with the message you want, then select one of the available messages and click **Finish**.

You should now see a link to the message, as in Figure 5-17.

![Figure 5-17 Showing link from input to associated message](image)

**Messages**

This section describes common tasks for messages.

**Create new message**

To create a new message:

1. In the messages container right-click and choose **Add Child → message**.
2. Give the message a name and click **OK**.

**Add a part to a message**

To add a part to a message:

1. Right-click the message and choose **Add Child → part**.
2. Give the part a name and click **OK**.
**Assigning an XSD element to a part**

To assign an XSD element to a part, right-click the message part and choose **Set Element**. You now get the following three options:

- **Create a new element** - This creates a new element with the name you specify in the name field. Take the default name or type in a different name, then click **Finish**.

- **Select an existing element** - This gives you a list of the elements currently available in the WSDL for you to choose from. Select an element and click **Finish**.

- **Import element from a file** - You must browse to a file with the element you want, then select one of the available elements and click **Finish**.

**Setting the type of a part**

Right-click the message part and choose **Set Type**. You now get the following three options:

- **Create a new type** - This creates a new type with the name you specify in the name field. Take the default name or type in a different name, then click **Finish**.

- **Select an existing type** - This gives you a list of the types currently available in the WSDL for you to choose from. Select a type and click **Finish**.

- **Import type from a file** - You must browse to a file with the type you want, then select one of the available types and click **Finish**.

**Bindings**

This section describes common tasks for bindings.

**Create binding**

To create a binding:

1. Right-click in the bindings container and choose **Add Child** → **binding**.
   a. Type in a name for the binding.
      
      If you click **Finish** now, you can later set a port type and protocol. If you already have a port type created it is better to choose it now because you also get the binding content generated for you.
   b. Choose a port type from the drop-down list, which will contain the available port types for the WSDL.
   c. Choose a protocol from the drop-down list.

2. Click **Finish**.

Figure 5-18 shows two binds created in this way, but using different protocols.
Figure 5-18  Bindings

**Set a port type**

If you want to change the port type of a binding, right-click the binding and choose **Set Port Type**. You now get the following three options:

- Create a new port type - This creates a new port type with the name you specify in the name field. Take the default name or type in a different name, then click Finish.
- Select an existing port type - This gives you a list of the port types currently available in the WSDL for you to choose from. Select a port type and click Finish.
- Import port type from a file - You must browse to a file with the port type you want, then select one of the available types and click Finish.

**Services**

This section describes common tasks for services.

**Create a service**

To create a service:

1. Right-click in the service container and choose **Add Child → Service**.
2. Give the service a name, then click **Finish**.

**Add port to a service**

To add a port to a service:

1. Right-click the service and choose **Add Child → Port**.
   a. Give the port a name.
   b. Choose the binding you want to use for this port. Also choose the protocol you want the port to use.
c. If you choose a protocol that is different from the protocol used in the binding, you will get errors in the task view. The protocols need to match.

2. Click **Finish**.

### 5.5.5 Creating a sample WSDL

We now create a simple WSDL file from scratch to demonstrate these tasks. The service we describe is a simple hello service. When this service is called you pass a name to the service and then you receive a greeting back that uses the name you pass in.

1. From the Services view in the business integration perspective, right-click the services project you wish to create the WSDL in and choose **New → Other**.
2. In the Web Services wizards choose **WSDL** and click **Next**.
3. In the file name give the WSDL the name Sample.wsdl, then click **Next**.
4. Take the defaults on this page and click **Finish**.
5. The WSDL editor now opens with the newly created blank WSDL.

First we will create a port type called Hello_PortType.

1. Right-click in the port type container and choose **Add Child → portType**.
2. Type in the name **Hello_PortType** and click **OK**.

We now create an operation against this new port type.

1. Right-click **Hello_PortType** and choose **Add Child → Operation**.
2. Call the operation sayHello and click **OK**.

We now need to give the operation an input and an output element.

1. Right-click the **sayHello** operation and choose **Add Child → Input**.
2. Do the same again choosing **Add Child → output**.

The WSDL should currently look like Figure 5-19.

---

Figure 5-19  Sample WSDL after creating operation
Our next step is to create message to go with our input and output.

1. Right-click **Input** and choose **Set Message**. Take the default of create message and also leave the default name of sayHelloRequest. Click **Finish**.

2. Right-click **Output** and choose **Set Message**. Take the default of create message and also leave the default name of sayHelloResponse. Click **Finish**.

Now that the message are created, we need to add some substance to them. Right-click the **sayHelloRequest** message and choose **Add Child → Part**. Give the part a name of firstName and click **OK**.

We now set an element for the firstName part. Right-click firstName and choose **Set Element**. Select **Create a new element** and give it a name of firstName (same as the part name for convenience), then click **Finish**.

By default the element is create as type xsd:string which is what we want.

You now need to do the same thing for sayHelloResponse. Right-click the **sayHelloResponse** message and choose **Add Child → part**. Give the part a name of greeting and click **OK**.

We now set an element for the greeting part. Right-click **Greeting** and choose **Set Element**. Select **Create a new element** and give it a name of greeting (same as the part name for convenience), then click **Finish**.

Again by default the element is created as type xsd:string, which is what we want.

The WSDL should now look like Figure 5-20, which is showing with the input selected. You should see a similar screen if you select output, but with the connection going to sayHelloResponse and the greeting element.

![Figure 5-20](image)

Figure 5-20 WSDL with message created and operation input selected

We now need to create a binding.

1. Right-click the bindings container and choose **Add Child → binding**.
   a. Give it the name Hello_Binding.
   b. Choose **tns:Hello_PortType** in the Port Type drop-down.
c. Choose **SOAP** in the protocol drop-down.

2. Click **Finish**.

You should now see that the WSDL looks like Figure 5-21. You can see that because we chose the port type when creating the binding. It has automatically created the binding operation and attached it to the sayHello operation in the port type.

![Figure 5-21 WSDL with binding defined](Image)

We now need to create a service.

1. Right-click the services container and choose **Add Child → service**.
2. Call the service **Hello_Service** and click **OK**.

A port now needs the be added to the **Hello_Service**.

1. Right-click the **Hello_Service** and choose **Add Child → port**.
   - Call the port **Hello_Port**.
   - From the binding drop-down choose **Hello_Binding**.
   - From the protocol drop-down choose **SOAP** to match the protocol of the binding chosen.
2. Click **Finish**.
3. Click **Save** or press Ctrl+S to save the WSDL.

We have now created the WSDL and it should look like Figure 5-22 on page 118.
Currently this WSDL is defined to be called via the SOAP protocol. We could also make it available via HTTP by creating a new service and binding as follows.

1. Right-click the bindings container and choose **Add Child → binding**.
   a. Give it the name `Hello_Binding_HTTP`. We added HTTP to the name to differentiate from the original binding, and also to let use know at a glance what the protocol this binding uses is.
   b. Choose `tns:Hello_PortType` in the Port Type drop-down.
   c. Choose `HTTP` in the protocol drop-down.

2. Click **Finish**.

We now need to create a service for this protocol.

1. Right-click the services container and choose **Add Child → service**.
2. Call the service `Hello_Service_HTTP` and click **OK**.

A port now needs to be added to the `Hello_Service_HTTP`.

1. Right-click the `Hello_Service_HTTP` service and choose **Add Child → port**.
   a. Call the port `Hello_Port_HTTP`.
   b. From the binding drop-down choose `Hello_Binding_HTTP`.
   c. From the protocol drop-down choose `HTTP` to match the protocol of the binding chosen.

2. Click **Finish**.

You will now see that the WSDL looks like Figure 5-23 on page 119. You can see that both services are attached to the same port type but coming in through different protocols. This allows you to offer your Web service over as many protocols as needed and supported.
5.5.6 Best practices with WSDL files

The best practices with WSDL files are:

- Put only one service definition into a WSDL file.
- Put only one port type into a WSDL file unless it is a flow, in which case, you will inline several required port types.
- Use separate WSDL files for service, binding, and interface.
- Only provide one binding per WSDL file. If you need to expose a service via multiple bindings, you should have one file per binding and a file with a service that aggregates the bindings and endpoints.

Attention: A service deployed as IBM Web Service or IBM Web Service over JMS service must have its complete service definition in a single WSDL file.

5.6 Using the BPEL editor

This section explains how to use the new BPEL editor by explaining what the icons mean, and by showing how to perform the more common tasks.

We then finish up by showing how to create a simple business process.

5.6.1 Basic terminology used in BPEL

Here is a very brief overview of the basic term used by BPEL when describing a process.

- **Partners** are the external users or services that interact with the process.
- **Activities** are the individual business tasks within the process that compose the larger business goal.
- **Elements** supplement activities, and assist them in accomplishing their tasks. They are nested within the activities with which they interact.
- **Variables** store the messages that are passed between these activities and partners.
- **Staff assignment** sends a task out to a human for interaction.
- **Compensation** returns the business process to a balanced state if something happens during execution to upset that balance.
- **Fault handling** identifies possible problems ahead of time, and tells the process how to deal with them.
- **Correlation sets** identify tokens that allow two participants in a conversation to identify each other in subsequent communications.

### 5.6.2 Anatomy of the BPEL editor

![Figure 5-24 The BPEL editor](image)
The BPEL editor can roughly be split up into eight separate areas, labeled 1 to 8 in Figure 5-24 on page 120.

1. Palette - Icons of the activities that can be dragged to the process area. Note that some of the icons on the palette actually open drop-downs when clicked. An example of this can be seen in Figure 5-26 on page 122.

2. Variables - Defined variables for the process.

3. Process area - Section on the canvas to visually arrange the activities.

4. Action bar - Series of actions related to the currently selected activity.

5. Partner links - Partner links that are the external users or services that interact with this process.

6. Canvas - Area that contains all visual elements of the editor.

7. Details - Situated below the canvas, allows the currently select activity to be configured. Reflects the parameters available for the selected activity.

8. Correlation sets - Shows the defined correlation sets for the process.

![Figure 5-25 Drop-down list from the palette](image)

### 5.6.3 Activities and their icons

An overview of the available BPEL4WS process activities is given in Table 5-4 on page 122.
The icons given in the table are what appear in the process area of the canvas. The icon you would choose in the activities palette looks like the small icon in the top center of these icons.

Figure 5-26 shows the invoke activity to demonstrate this. You can see that the small icon on the right, which you choose in the activity palette, is the same as the small upper center of the icon that appears on the process area.

Table 5-4 WebSphere Process Choreographer activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Invoke" /></td>
<td>The invoke activity performs an operation. The operation is defined by a partner link and may be synchronous or asynchronous.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Receive" /></td>
<td>The receive activity waits for an external input to the process before continuing. The operation supported by the receive activity is defined by a partner link.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Reply" /></td>
<td>The reply activity sends a message to the partner defined by a partner link. This is typically used in processes that need to return a message to the partner that instigated the process.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Pick" /></td>
<td>The pick activity waits for an incoming message and selects a path appropriate to the first message received. A time-based path can be configured to manage situations where no message is received. A partner link is associated with each message path.</td>
</tr>
<tr>
<td><img src="image5.png" alt="Staff" /></td>
<td>The staff activity delegates a task within the process to a human. The user interface in this case is either a custom application based on the process choreographer API or the Web client provided that comes with WebSphere Business Integration Server Foundation.</td>
</tr>
<tr>
<td><img src="image6.png" alt="Transform" /></td>
<td>The transformer activity maps the contents of one or more message types to the contents of another.</td>
</tr>
<tr>
<td>Activity</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Assign</strong></td>
<td>The assign activity copies information from one part of the process to another.</td>
</tr>
<tr>
<td><strong>Switch</strong></td>
<td>The switch activity evaluates the conditions on a series of control paths and follows the first one that matches.</td>
</tr>
<tr>
<td><strong>While</strong></td>
<td>The while activity repeats the activities that it contains as long as a condition is met.</td>
</tr>
<tr>
<td><strong>Wait</strong></td>
<td>The wait activity stops the process until a point in time has occurred or a time interval has elapsed.</td>
</tr>
<tr>
<td><strong>Sequence</strong></td>
<td>The sequence activity defines a serial control path within a process.</td>
</tr>
<tr>
<td><strong>Flow</strong></td>
<td>The flow activity defines a potentially parallel control path within a process.</td>
</tr>
<tr>
<td><strong>Terminate</strong></td>
<td>The terminate activity stops the process immediately without performing any compensation or fault handling. The behavior of this activity depends on the location within the process.</td>
</tr>
<tr>
<td><strong>Throw</strong></td>
<td>The throw activity signals that an error has occurred. This is typically handled by a fault handler element associated with a higher level of process structure.</td>
</tr>
<tr>
<td><strong>Empty</strong></td>
<td>The empty element does nothing. It can be used as a placeholder during process design, and then changed to the appropriate activity when the process is implemented.</td>
</tr>
<tr>
<td><strong>JavaSnippet</strong></td>
<td>Java code can be embedded into the process using the JavaSnippet activity. While it is possible to embed business logic into this type of activity it is not advisable, as it removes the clarity of the process modelling. Snippets are designed to perform lightweight utility activities such as data mapping.</td>
</tr>
</tbody>
</table>
5.6.4 Tasks

This section provides an overview of some of the common tasks you can perform using the BPEL editor.

**Create a new business process**

To create a new business process:

1. Switch to the Business Integration Perspective in WebSphere Studio Application Developer Integration Edition.

2. Create a Service project by clicking File → New → Service Project from the main menu.

3. Select this newly created Service project and click File → New → Business Process.

4. In the New Business Process window, specify a package and a name for the new process, and click Next.

5. In the Choose Process Type window choose from the options listed below, and click Finish.
   - Flow-based BPEL process - Use this setting if you want that control dictated by control links.
   - Sequence-based BPEL process - Use this setting if you want your process controlled by structured activities.

This creates a process, and launches it in a new process editor. It also generates a few files that you can view in the Services view. The file with the .bpel extension contains the code that describes this process, and the WSDL file describes the service interface.

Screenshots of this task can be seen in the sample BPEL section in “Common tasks” on page 165.

**Partner links**

This section provides an overview of how to work with partner links in the BPEL editor.
**Create a partner link**

To create a partner link:

1. In the Partner Links area on the canvas, click the plus icon (+).
2. Give the partner link an appropriate name.
3. Click the **Implementation** tab in the Details area.

   Figure 5-27 shows where you can find these fields and icons in the editor.

   ![Image](image1.png)

   **Figure 5-27  Create and name a new partner link**

   Figure 5-28 shows the fields in the implementation tab of the new partner link.

   ![Image](image2.png)

   **Figure 5-28  Implementation tab for a partner link**

4. If you would like to reference a WSDL interface file (such as the one that was automatically created by the Process editor), click **Browse**.
5. To launch the New Partner Link Type window, either click **New**, or select a Partner Link Type from the drop-down list and click **Edit**. You will see a dialog, as in Figure 5-29.

![New Partner Link Type](image)

**Figure 5-29  New partner link type window**

6. Modify the fields in the New Partner Link Type window as follows:
   - If this is a new partner link, then you can choose to reference an existing WSDL interface file (click **Browse**), or create a new one (click **New**).
   - Select the number of roles that this partner link will expose.
   - Name the roles, browse to the WSDL interface file that defines the port type that you want to associate with the role, and select it.

7. Click **OK**.

**Variables**
This section provides an overview of how to work with variables in the BPEL editor.

**Create a variable**
To create a variable:
1. In the Variables area on the canvas, click the plus symbol.
2. Give the variable an appropriate name.
Figure 5-30 shows where these fields can be found in the editor.

Figure 5-30   Create a new variable

Configure it as follows:
1. Click the Message tab in the Details area.
2. Browse to a WSDL file that has at least one message type and part defined.
3. Select an appropriate Message type and part.

Figure 5-31 shows the Message tab fields.

Figure 5-31   Message tab fields

**Activities**

This section provides an overview of how to work with activities in the BPEL editor.
Create an activity
To create an activity:

1. To use the Palette to add an activity:
   a. In the palette, click an activity’s icon.
   b. Click a position in the process editor where you want the activity to appear.

2. To use the context menu to add an activity:
   a. On the canvas, decide where you want to place your new activity, and select the existing activity that is directly below that position.
   b. Press SHIFT+F10 or right-click to launch the context menu.
   c. From the menu, select Insert before and select the desired activity.

Figure 5-32 shows the Insert before selection in the context menu. Note that the top of the context menu has been removed for the figure.

5.6.5 Sample BPEL process
We now create a sample business process from scratch. The sample will be a simple calculator application.
Initial setup
For the initial setup:

1. Create a service project called BPEL Sample.
2. Create three packages in this project with the following names:
   - itso.demo
   - itso.demo.processes
   - itso.demo.services
3. Import Calculator.java into the itso.demo package.

Note: Calculator.java is included in our redbook additional material. To obtain the additional material follow the instructions in Appendix B, “Additional material” on page 319. For more details on working with the additional material see Appendix A, “Working with the redbook sample code” on page 315.

Create service from the Java file
Calculator.java is a simple Java class that we use to create our sample process. Creating a service from the Java file is a prerequisite for us to use the class in the BPEL process, as it creates a WSDL interface to the class that makes it easy for us to access it from a BPEL process. To create the service using the Business Integration perspective in WebSphere Studio Application Developer Integration Edition:

1. Expand the tree Service Projects → BPEL Sample → itso.demo and select Calculator.java.
2. Generate a service from Calculator.java. The service generation can be started by:
   a. Choosing the menu option File → New → Service built from.
   b. Right-click Calculator.java and choose the context menu New → Service built from.
   c. Click the toolbar Service built from icon.
3. On the Create Service page make sure Java is selected and click Next. See Figure 5-33 on page 130.
4. Select all methods in Calculator.java and click Next. See Figure 5-34 on page 131.

**Tip:** Click the *What does this diagram mean* link shown in Figure 5-33 to launch the WebSphere Studio help and get a detailed description of what is being generated by this wizard and how this relates to the Services run-time environment. The main point to note is that with this wizard we are performing the role of the service provider in the service-oriented architecture (SOA) and generating the service implementation.
5. On the Add Java Service page shown in Figure 5-35 on page 132 change the package name to itso.demo.services, accept all other default values for WSDL files to be generated, and click Finish.

This will create three WSDL files in the itso.demo.services package of the EnterpriseServices project:

- Calculator.wsdl is the service interface.
- CalculatorJavaBinding.wsdl is the Java binding for the service.
CalculatorJavaService.wsdl is the Java service that uses the service interface and the Java binding.

The WSDL editor will open on the CalculatorJavaBinding.wsdl file as shown in Figure 5-36 on page 133.
Create the process interface WSDL

Use the WSDL editor to create a WSDL file that will be the interface to our BPEL business process. The following steps describe at a high level the WSDL to build:

1. Create a new service interface that will be the definition of our process. From the services view select the package itso.demo.services in the EnterpriseServices project, right-click and choose New → Service Interface or use the toolbar icon Create a service interface.

2. Enter CalculatorInterface.wsdl as the file name, accept the other default values, and click Finish.

3. The WSDL editor will open on the CalculatorInterface.wsdl file.
Note that the file will have a port type called CalculatorInterface already set up for us. The WSDL that we need to create is as follows:

- **Port type - CalculatorInterface**
- **Operation - DoCalculation**
  - input - Using message CalculationRequest
  - output - Using message CalculationResponse
- **Message - CalculatorRequest**
  - part - operation of type xsd:string
  - part - inputX of type xsd:int
  - part - inputY of type xsd:int
- **Message - CalculatorResponse**
  - part - intResult of type xsd:int
  - part - floatResult of type xsd:float
  - part - message of type xsd:string
- **Message - CalculatorErrors**
  - part - errorOperation of type xsd:string
  - part - errorX of type xsd:int
  - part - errorY of type xsd:int
  - part - errorMessage of type xsd:string

Example 5-1 shows the source of the completed CalculatorInterface.wsdl.

**Example 5-1  CalculatorInterface.wsdl source**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CalculatorInterface"

targetNamespace="http://services.demo.itso/CalculatorInterface"

xmlns="http://schemas.xmlsoap.org/wsdl/"

xmlns:tns="http://services.demo.itso/CalculatorInterface"

xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<message name="CalculatorRequest">

<part name="operation" type="xsd:string"/>

<part name="inputX" type="xsd:int"/>

<part name="inputY" type="xsd:int"/>

</message>

<message name="CalculatorResponse">

<part name="message" type="xsd:string"/>

<part name="intResult" type="xsd:int"/>

<part name="floatResult" type="xsd:float"/>

<part name="message" type="xsd:string"/>

</message>

<message name="CalculatorErrors">

<part name="errorOperation" type="xsd:string"/>

<part name="errorX" type="xsd:int"/>

<part name="errorY" type="xsd:int"/>

<part name="errorMessage" type="xsd:string"/>

</message>

</definitions>
```
Figure 5-37 shows the completed CalculatorInterface.wsdl in the graph view of the WSDL editor.
Create the business process

To create the business process:

1. Right-click the package itso.demo.processes and choose New → Business Process, as shown in Figure 5-38.

![Figure 5-38 Create business process](image)

2. Give the process a file name of CalculatorProcess and make sure the package name is itso.demo.processes, as shown in Figure 5-39 on page 137.

3. Click OK.
4. Select the **Sequence-based BPEL process** radio button as seen in Figure 5-40.

5. Click **Finish**.

---

**Figure 5-39  Name the process**

**Figure 5-40  Choose process type**
6. By default, the process is created with a variable and a partner link that we will not use and need to delete.

7. Right-click the variable and select **Delete** from the menu, as shown in Figure 5-41.

---

![Figure 5-41 Delete default variable](image)

8. Right-click the partner link shown in Figure 5-42 on page 139 and select **Delete**.
Create partner links

We now need to create a partner link for the Calculator Interface.

1. Using the left mouse button, drag the CalculatorInterface .wsdl file from the itso.demo.services package in the service view, onto a blank space on the BPEL editor canvas.

2. Select a port type of CalculatorInterface, as shown in Figure 5-43.

3. Click OK.
4. Make sure the calculatorInterface partner link is selected under partner links.
5. In the details area, click the **Implementation** tab. See Figure 5-44.

Figure 5-44  In partner role name

As can be seen in Figure 5-44, the partner link is currently set up with a partner role. We need this to be moved to a process role.

6. Click the <-> button between the two roles. This will shift the role to the process role, as can be seen in Figure 5-45 on page 141.
We now need to set the variable in the Receive activity.

1. Select the receive activity on the canvas.
2. Select the **Implementation** tab in the details area.
3. Make sure the partner link drop-down says CalculatorInterface.
4. Make sure the operation drop-down is showing DoCalculation.

We now need to make a new request variable.

1. Click **New**, which is on the right-hand side of the details area next to the request drop-down.
2. Give the variable a name of CalculatorRequest.
3. Click **OK**. The screen should now look like Figure 5-46 on page 142.
4. Click the newly created variable, under variables on the canvas.

5. Click the **Message** tab in the details area.

As Figure 5-47 on page 143 shows, you can see that the message parts are automatically imported and understood by the BPEL editor.
We now need to do the same thing for the Reply activity.

1. Select the reply activity on the canvas.
2. Select the **Implementation** tab in the details area.
3. Make sure the partner link drop-down says CalculatorInterface.
4. Make sure the operation drop-down is showing DoCalculation.
5. Set the reply type to Normal.

We now need to make a new response variable.

1. Click **New**, which is on the right-hand side of the details area next to the response drop-down.
2. Give the variable the name CalculatorResponse.
3. Click **OK**.

**Figure 5-47  Shows parts in the request message**
We also need to create the Calculator partner link.

1. Drag and drop the itso.demo.services.Calculator.wsdl file onto the BPEL editor canvas. You will be prompted with the dialog in Figure 5-48.

2. Select the port type of **Calculator**.

![Select Service dialog](image)

*Figure 5-48 Selecting port type for the calculator partner link*

**Create switch activity**

Click the switch icon in the palette, then insert the switch between the receive and reply activities. You can see the switch has been added in Figure 5-49 on page 145.

We now need to add case statements to the switch.

1. Right-click the switch activity on the canvas and choose **Add Case**, as shown in Figure 5-49 on page 145.
2. Set the name of the case in the Description tab of the details area to **AddCase**, as shown in Figure 5-50 on page 146.
We now need to set the condition statement of this case.

1. On the Condition tab, click **Condition**, as shown in Figure 5-51 on page 147.
2. Expand CalculatorRequest in the right hand pane as shown in Figure 5-52 on page 148.
3. Select operation, also shown in Figure 5-52 on page 148.
The right-hand pane now changes to show the valid methods that can be called.

1. Scroll down to `equalsIgnoreCase` and select it. See Figure 5-53 on page 149.
Figure 5-53  Choosing the method for the comparison

2. Click anotherString, as shown in Figure 5-54 on page 150.
3. Click string in the right-hand pane.
4. Enter the text add.
5. Click OK.
6. Add three more case activities to the OperationSwitch. Name the cases:
   - SubtractCase
   - DivideCase
   - MultiplyCase

7. Add conditions to each case:
   - SubtractCase -
     CalculatorRequest.operation.equalsIgnoreCase("subtract")
   - DivideCase - CalculatorRequest.operation.equalsIgnoreCase("divide")
   - MultiplyCase - CalculatorRequest.operation.equalsIgnoreCase("multiply")
Create the invoke activities

Next we add an Invoke activity to call the Calculator partner link.

1. Select an Invoke activity from the palette and drag it to the canvas between the Receive and Reply activities. Rename the activity InvokeAdd. See Figure 5-55.

2. On the Implementation tab for the InvokeAdd activity select Partner Link Calculator, Port Type Calculator, and Operation Add. See Figure 5-56 on page 152.
3. Click **New** and add a request variable called AddRequest. Figure 5-57 on page 153 shows the new variable.
4. Click New and add a response variable called AddResponse. Figure 5-58 on page 154 shows the new variable.
5. Follow the same steps to create invoke activities as follows:
   - InvokeSubtract
   - InvokeDivide
   - InvokeMultiply

6. Set the implementation for each invoke activity to call the correct operation on
   the Calculator partner link and create new request and response variables for
   each operation:
   - InvokeSubtract
     - Partner link - Calculator
     - Port type - Calculator
     - Operation - subtract
     - Request - SubtractRequest
     - Response - SubtractResponse
– InvokeDivide
  • Partner link - Calculator
  • Port type - Calculator
  • Operation - divide
  • Request - DivideRequest
  • Response - DivideResponse

– InvokeMultiply
  • Partner link - Calculator
  • Port type - Calculator
  • Operation - multiply
  • Request - MultiplyRequest
  • Response - MultiplyResponse

**Create assigns**

To create assigns:

1. Select **Assign** from the palette and add Assign activities.

2. Add assign activities between the case activities and invoke activities in the switch block. Name the assigns:
   - SetupAdd
   - SetupSubtract
   - SetupDivide
   - SetupMultiply

See Figure 5-59 on page 156.
3. Use the Implementation tab for each assign activity to copy from the data from the process input CalculatorRequest message to the correct message for each operation on the Calculator partner type:
   - SetupAdd
     - Copy variable or part CalculatorRequest.CalculatorRequest.inputX to AddRequest.addRequest.x.
     - Copy variable or part CalculatorRequest.CalculatorRequest.inputY to AddRequest.addRequest.y.
– SetupSubtract
  • Copy variable or part CalculatorRequest.CalculatorRequest.inputX to SubtractRequest.subtractRequest.x.
  • Copy variable or part CalculatorRequest.CalculatorRequest.inputY to SubtractRequest.subtractRequest.y.
– SetupDivide
  • Copy variable or part CalculatorRequest.CalculatorRequest.inputX to DivideRequest.divideRequest.x.
  • Copy variable or part CalculatorRequest.CalculatorRequest.inputY to DivideRequest.divideRequest.y.
– SetupMultiply
  • Copy variable or part CalculatorRequest.CalculatorRequest.inputX to MultiplyRequest.multiplyRequest.x.
  • Copy variable or part CalculatorRequest.CalculatorRequest.inputY to MultiplyRequest.multiplyRequest.y.

4. Add assign activities between the invoke activities and the process reply. Name the assigns:
   – SetupAddReply
   – SetupSubtractReply
   – SetupDivideReply
   – SetupMultiplyReply

See Figure 5-60 on page 158.
5. Use the Implementation tab for each assign activity to copy the data from the output message of the operation on the Calculator partner type to the process reply CalculatorResponse message.
   - SetupAddReply
     - Copy variable or part AddResponse.addResponse.result to CalculatorResponse.CalculatorResponse.intResult.
– SetupSubtractReply
  • Copy variable or part SubtractResponse.subtractResponse.result to CalculatorResponse.CalculatorResponse.intResult.
  • Copy fixed value Subtract Successful to CalculatorResponse.CalculatorResponse.message.

– SetupDivideReply
  • Copy variable or part DivideResponse.divideResponse.result to CalculatorResponse.CalculatorResponse.floatResult.
  • Copy fixed value Divide Successful to CalculatorResponse.CalculatorResponse.message.

– SetupMultiplyReply
  • Copy variable or part MultiplyResponse.multiplyResponse.result to CalculatorResponse.CalculatorResponse.intResult.
  • Copy fixed value Multiply Successful to CalculatorResponse.CalculatorResponse.message.

6. Save the Calculator process

**Generate deploy code**
As can be seen in Figure 5-61 on page 160, there will be a warning message associated with the CalculatorProcess. This warning is simply telling us that the deployment code for this process must be generated.
To generate deploy code:

1. Right-click the **CalculatorProcess.bpel** and choose **Enterprise Services** → **Generate Deploy Code**, as can be seen in Figure 5-62 on page 161.
Figure 5-62  Selecting generate deploy code for the process

2. Click Calculator as shown in Figure 5-63 on page 162, because we need to set a valid port for the partner interface, now that we are at deploy time.

Note: We could have set this earlier, but we wanted to demonstrate that we can leave this binding to deploy time.
3. Click **Browse** to set the WSDL file to use.

4. Choose **CalculatorJavaService.wsdl** as seen in Figure 5-64 on page 163.

5. Click **OK**.
6. Choose the Service and Port entries, as shown in Figure 5-65 on page 164.
7. Click **OK**.
8. The code generation will run for a little while. When completed you will see in the J2EE hierarchy view that a number of Web and EJB modules are generated. An example of this can be seen in Figure 5-66 on page 165.
5.7 Common tasks

This section provides details of how to do a number of key tasks in WebSphere Studio Application Developer Integration Edition. The tasks are chosen because they represent actions you will often need to take as you work with the samples in our redbook. This section is not provided as a detailed guide to WebSphere Studio Application Developer Integration Edition, but to aid you when you use our redbook.
5.7.1 Create service project

To create a service project:

1. Two ways to create a service project using the business integration perspective are:
   - Click the **Create a Service Project** icon on the toolbar.
   - Click **File → New → Project** then choose **Business Integration** and **Service Project**.

2. Both of these options will then take you to the screen shown in Figure 5-67.
   a. Give the project a name and a directory. Using the default directory is generally a good choice. Click **Next**.

![New Project](image)

*Figure 5-67  Create service project - Name page*

3. You will now see the project properties page, like Figure 5-68 on page 167. You can add any libraries you expect to reference, as well as other projects. You can edit this later so it is fine to just click **Finish** now.
4. You now have a new service project, which you will be able to see in the services view.

5.7.2 Create test server

We will only describe how to create an Integration test server here, as that is the only test server that has the business process engine (needed to run business processes).

1. Change to the Servers perspective.

2. In the Server Configuration view, right-click Servers and choose New → Server and Server Configuration.

3. Give the server a name and choose Integration Test Environment under the WebSphere Version 5.1 folder in Server Type. You can leave the rest as defaulted. You should now have a screen like Figure 5-69 on page 168.
4. Click **Next**.

You will now see a page where you can set the HTTP port number the server will use for incoming requests as in Figure 5-70 on page 169. The default of 9080 is fine, although you can use any valid port number available.

5. Click **Finish**.
5.7.3 Enabling security on a test server

To enable security on a test server:

1. Open the test server properties page by double-clicking the test server in the Server Configuration view of the Servers perspective.

2. Click the Security tab along the bottom of the properties window. Figure 5-71 shows the security page with the tab highlighted.
3. Make sure **Enable Security** is ticked and that Enforce Java 2 Security is *not* ticked.

**Important:** If you tick Enforce Java 2 Security, you will get a lot of authorization errors on files when trying to start the test server.

Java 2 Security is not needed except in the highly secure scenarios, which are beyond the scope of this publication.

4. You need to enter a user ID that has administrator access to the local operating system. Also, if you are running under windows, you should also make sure the user ID has the Act as operating system privilege.

5. Enter the password twice in the given fields.

You will also need to update the localhost/BPEAuthDataAliasEmb localhost_server1 entry JAAS Authentication Entries table to give it a valid user ID and password.

6. Click the alias and then click **Edit**. You will be presented with a screen like Figure 5-72 in which you should use the same user ID and password entered on the last page. It is not mandatory to use the same user ID, but it makes administration easier.

![Edit JAAS Authentication Entry](image)

**Figure 5-72** Updating the JAAS authentication alias user ID and password

7. Click **OK**.

8. Now save the test server properties by clicking **Save** or pressing Ctrl+S.
5.7.4 Adding and removing applications to the test server

To deploy or remove an enterprise application to the test server, do the following:

1. Go to the Servers perspective. In the Server Configuration view, right-click the test server then choose **Add and remove projects**, as shown in Figure 5-73.

   ![Figure 5-73 Deploying to test server](image)

2. This will bring up the dialog in Figure 5-74 on page 172, which can be used to either add or remove an application from the server.

3. The **Available projects** pane contains the applications in this workspace that can be deployed to the server (but are not yet deployed). The **Configured projects** pane contains those application in the workspace that have already been deployed to the server.

4. To deploy an application, simply select it in the **Available projects** pane and click **Add >**.

5. To remove an application simply select it in the **Configured projects** pane and click **< Remove**.

6. You can also add or remove all applications by clicking the respective Add/Remove all buttons.
7. When you have finished adding/removing projects, click **Finish**.

8. You should now be able to see the deployed applications by expanding the test server in the Server configuration view, as in Figure 5-75.

5.7.5 Setting JMS properties in test server

These properties are all set in the test server configuration properties.

This can be accessed by either double-clicking the test server in the server configuration view, or right-clicking it and choosing **Open**.
Create queue
To create a queue:

1. In the JMS tab you will see a list of queue names similar to those shown in Figure 5-76.

![Figure 5-76 JMS queue list](image)

2. Click Add.
3. Give the queue a name and then click OK.

Create queue connection factory
To create a queue connection factory:

1. In the JMS tab part way down the page you will see a list of Queue Connection Factories, as in Figure 5-77.

![Figure 5-77 JMS QCF list](image)

2. Click Add.
3. You need to fill in the following two fields:
   - Name
   - JNDI Name
4. The rest can be left as defaulted. See Figure 5-78 on page 174 for a sample.
5. Click **OK**.

![Image of WASQueueConnectionFactory](image.png)

**Figure 5-78 Create QCF**

**Create queue destinations**

To create queue destinations:

1. In the JMS tab, scroll near to the bottom and you will see the list of JMS Destinations, which will look like Figure 5-79 on page 175.

2. Click **Add**.

3. You need to fill in the following two fields:
   - Name
   - JNDI Name

4. The rest can be left as defaulted. See Figure 5-78 for a sample.
5. Click OK.

![Add WASQueue](image)

Figure 5-80  Create JMS Destination

Create listener ports
To create listener ports:
1. In the EJB tab you will see a list of listener ports similar to Figure 5-81.

![Listener Ports](image)

Figure 5-81  Listener port list

2. Click Add.
3. You need to fill in three fields in this dialog:
   - Name - Can be anything.
   - Connection Factory JNDI Name - Choose an already defined connection factory from the drop-down.
   - Destination JNDI Name - Again choose from the drop-down of already defined destination queues.

   See Figure 5-82 for a sample.

4. Click **OK**.

![Add Listener Port](image)

**Figure 5-82 Create listener port**

### 5.7.6 Setting a project build path

WebSphere Studio projects that include Java code have a property called Java build path. You use this property so that WebSphere Studio can locate all the Java classes it needs to reference when it builds a project. The build class path is a list of paths visible to the Java compiler when building the project.

To change this build path:

1. Select a project, right-click, and choose **Properties**, as shown in Figure 5-83 on page 177.
Figure 5-83  Project properties

2. Select **Java Build Path** from the list of properties, as shown in Figure 5-84 on page 178.
3. Note that there are four tabs on the Java build path property view. They are:
   - **Source**
     
     This tab is used to add or remove source folders from the build path. Source folders contain .java files that will be compiled to .class files when the project is built. See Figure 5-85 on page 179.
Figure 5-85  Source folders for build path

- Projects

This tab is used to add or remove projects from the build path. You can choose WebSphere Studio projects from the current workspace. See Figure 5-86 on page 180.
Figure 5-86  Projects for build path

- **Libraries**

  This tab is used to add or remove libraries from the build path. Libraries can be:
  
  - **Internal JAR files**, which are JAR files managed by the WebSphere Studio workbench
  
  - **External JAR files**, which are JAR files from the file system outside of direct WebSphere Studio control
  
  - **Predefined system libraries** such as the WebSphere Studio defined JRE System library
  
  - **Class folders**, which are folders containing .class files produced when the compiler builds a WebSphere Studio project
  
  - **Variables** that are an indirect way of referring to JAR files so that local file system paths are not embedded in the project build path
See Figure 5-87.

Figure 5-87 Libraries for build path

- Order and Export

  This tab allows you to change the order of source folders, projects and libraries that are in the build path. Any selected build path entries in this list will be exported, which means that they will be visible to projects that require the current project. See Figure 5-88 on page 182.
4. Example steps to take to add a variable to the project build path from the Libraries tab are:
   a. Click **Add Variable**.

   Variables are created using the WebSphere Studio preference dialog and they may point directly to a JAR file or to a folder that contains JAR files.

   b. Select a variable as shown in Figure 5-89 on page 183.
c. The variable we have selected points to a folder, so we click Extend, which allows us to select a JAR file contained in the folders referenced by the variable. See for Figure 5-90 an example.

d. Click OK to complete the variable extension.
e. Click **OK** to save the new build path properties.

### 5.7.7 Exporting code for deployment

You can export resources from WebSphere Studio in many different formats. Exporting resources moves resources from the workspace used by WebSphere Studio to a target location.

Within WebSphere Studio the code of our sample application is organized in an enterprise application project, but to deploy the solution to WebSphere Business Integration Server Foundation we want to create EAR files externally to WebSphere Studio.

To create an EAR file we export our project from WebSphere Studio.

1. To start the export process we can:
   a. Choose **File → Export** as shown in Figure 5-91 on page 185.
b. In this case we need to select an export destination type of **EAR file**, as shown in Figure 5-92 on page 186.

**Note:** EAR file will already be selected if we have previously selected an enterprise application project in WebSphere Studio.
c. We need to use the drop-down list on the Export page as shown in Figure 5-93 on page 187 to choose the project we want to export.
2. Select the EAR file we wish to export, right-click, and choose Export → Export EAR File, as shown in Figure 5-94 on page 188.
3. Choose a destination for the exported EAR file. You can either enter a file name or click **Browse** to choose an export destination. You can also decide whether you want to include source files in the export and whether to overwrite the existing file. See Figure 5-95 on page 189.
4. You can click **Advanced** to select advanced export options. This will bring up the dialog shown in Figure 5-96 on page 190.
5. Click **Finish** to complete the export.
Deploying the existing sample solution

This chapter takes the sample application described in Chapter 3, “Solution design and technical overview” on page 29, and shows how to deploy and test it in two different environments.

We first deploy in the development environment of WebSphere Studio Application Developer Integration Edition V5.0. We import a predefined workspace, which already contains the application, then we deploy this to the built-in WebSphere Application Server Enterprise V5.0 test server.

Once the application is working in the development environment, we then deploy the application into a simplified production environment using WebSphere Application Server Enterprise V5.02.
6.1 Sample code

Chapter 3, “Solution design and technical overview” on page 29, describes the Private Exchange application as you would deploy it in a real production environment. The objective of this chapter is to demonstrate business processes in action, and not the accompanying real-world systems that they would talk to.

With this in mind we have simplified the surrounding code, while maintaining the core aggregation service as described in Chapter 3, “Solution design and technical overview” on page 29.

6.1.1 Obtaining the sample code

The sample code described in this chapter is supplied as part of our redbook additional material. For instructions on obtaining the additional material refer to Appendix B, “Additional material” on page 319. For details of how to work with a workspace that contains the sample code relevant to this chapter see Appendix A, “Working with the redbook sample code” on page 315.

Workspace
To use the workspace, just start WebSphere Studio Application Developer Integration Edition, and when prompted for a workspace choose the Version 5.0 workspace provided as part of our redbook additional material.

EAR files
You can also set up your own workspace and import the sample application, as a number of EAR files that are also provided in the redbook additional material.

To do this, start WebSphere Studio Application Developer Integration Edition with whatever workspace you like, then import the following Version 5.0 EAR files:

- AggServiceEAR
- EIS1EAR
- EIS2_SRVEAR
- EIS3_SRVEAR
- ExchangeStub2EAR
- MessageTester
- PEUtil (this is a JAR file not an EAR)

6.1.2 Overview of code

The following sections provide a brief overview of each component and what we have changed to simplify the sample.
**AggServiceEAR**
This is the core of the sample application. Here you will find the business processes and core implementation of the aggregation and approval process. This code is essentially unchanged from that described in Chapter 3, “Solution design and technical overview” on page 29.

**EIS1EAR**

**Attention:** EIS1 can only be used in the production environment. It will not work under WebSphere Studio Application Developer Integration Edition if you are using the internal JMS provider.

You will also have to export this application before installing it in production because you must make a change to the source.

EIS1 is meant to be any external application that can be used to create an ITEM, which will then undergo the approval process for publishing to the Exchange Server and any other commerce application.

**Attention:** It is not mandatory that you install EIS1 because MessageTester is an even simpler application that can be used instead. EIS1 is a more flexible application that will allow you to much more easily create ITEMS for the approval process.

We recommend installing EIS1 for this flexibility.

Our implementation here is a fairly simple Web application, utilizing servlets and JSPs that allow a user to create an ITEM from a simple Web page.

Once deployed it can be run by going to the following URL:

http://host_machine:9080/EIS1/MQServletAInputForm.html

Currently the queue manager name that houses the PEAGGIN queue is hardcoded into the EIS1 application. To use EIS1 with a different queue manager name (which you almost certainly will need to do), you must update com.ibm.ssys.mq.servlets.MQServletBean.java in the EIS1 project.

You will see a line in the execute() method that looks like:

```
myMQConnSend.setQManagerName("WAS_BPE_MKA0KLFX");
```

You must change this to match your queue manager name.
You will need to export the EIS1EAR project before installing to production so that this change will take effect. You can export by right-clicking the EIS1EAR project and choosing Export.

Choose EAR file as the export type and click Next, then designate the name and place of the file you want to export to and click Finish. You will then use this file when installing the application on the production server.

**EIS2_SRVEAR**
EIS2 is used by the aggregation process to retrieve the price of a given ITEM or a list of all ITEMS and their prices from the database. We have simplified the code by removing the need for an external data source and simply hardcoding a set of items into an internal data structure to the code.

You can modify the sample data simply by editing the initialize() method in class org.tempuri.EIS2_EJBServiceBean.java in the EIS2_SRVEJB project, then compile and redeploy.

**EIS3_SRVEAR**
EIS3 is similar to EIS2 except that it retrieves the price for a given ITEM. It also only returns the price for a specified ITEM and not a list of all ITEMS.

As with EIS2, we have significantly simplified the code by keeping the data hardcoded into an internal data structure.

You can modify the sample data simply by editing the initialize() method in class eis3Send.EIS3SendServiceBean.java in the EIS3_SRVEJB project, then compiling and redeploying.

**ExchangeStub2EAR**
This EAR provides a test version of the exchange functionaliy.

**MessageTester**
Due to the way EIS1 has been implemented and because we are using the Internal JMS provider, we are unable to use EIS1 to create new ITEMS. MessageTester is a very simple application that uses JMS to place a hardcoded ITEM onto the PEAGGIN queue.

There is a small servlet contained in itso.demo.WriteQ.java, which is called from a simple HTML page. To change the data that the ITEM is created with, you need to edit the init() method in WriteQ.java, as shown in Example 6-1 on page 195. To change the ITEM data just edit the String variables that are highlighted.
Example 6-1  init() method of WriteQ.java

public void init() throws ServletException{
    super.init();
    getQCF();

    String ItemNumber = "IN1111111";
    String Supplier = "IBM";
    String PartNumber = "1234567";
    String Category = "Test";
    String Name = "part1";
    String Description = "The First Part";
    String Weight = "2.0";
    String ImageURL = "something";

    theMessage = "<?xml version="1.0" ?>\n<ExchangeDBRoot><Part><ItemNumber>";
    theMessage += ItemNumber;
    theMessage += "</ItemNumber><Supplier>");
    theMessage += Supplier;
    theMessage += "</Supplier><PartNumber>";
    theMessage += PartNumber;
    theMessage += "</PartNumber><Category>";
    theMessage += Category;
    theMessage += "</Category><Name>";
    theMessage += Name;
    theMessage += "</Name><Description>";
    theMessage += Description;
    theMessage += "</Description><Weight>";
    theMessage += Weight;
    theMessage += "</Weight><ImageURL>";
    theMessage += ImageURL;
    theMessage += "</ImageURL></Part></ExchangeDBRoot>";
}

There is also a servlet itso.demo.ReadQ.java that is also call from the same URL. This gets a message from PEAGGOUT and can be used to retrieve messages to see that the processes have completed correctly and placed their output on the PEAGGOUT queue.

PEUtil
This is a utility class used by a number of the other projects, mostly for passing messages between components. A JAR file should be created and put in the classpath in order to be able to run the sample.
6.1.3 Deployment architecture

As well as simplifying the code, we are also simplifying the way in which we will deploy the sample. The intended scenario is for each EIS to be housed on separate machines, as well as enabling clustering of WebSphere MQ and remote instances of the database.

We will have one single machine running one instance of an application server, using a local database and non-clustered WebSphere MQ running locally.

Each application will be installed in the single application server instance (server1), and security will be enabled using the local operating system.

Figure 6-1 shows this simple deployment architecture.

![Deployment architecture diagram]

Figure 6-1  Deployment architecture

6.2 Sample code deployment in WebSphere Studio Application Developer Integration Edition V5.0

In this section, we import the sample into Studio and deploy it to the built-in WebSphere Application Server Enterprise V5.0 test environment. This is done with the following steps:

1. Import the code into Studio.
2. Create a test server.
3. Deploy code to the test server.
4. Configure the test server.
5. Publish and start the test server.
6. Test the deployed application.

### 6.2.1 Environment

We deployed this sample using the following applications and options:

- WebSphere Studio Application Developer Integration Edition V5.02
- Internal JMS provider

We initially tried to use Embedded MQ, but when security was turned on, we were unable to get our Message Driven Beans to work. The basic error message was unable to authenticate user when trying to access JMS.

### 6.2.2 Importing the sample code

Although the sample application can be imported into an existing workspace, by far the simplest approach is to utilize the predefined workspace provided.

Start Studio and when prompted for a workspace, point it to the workspace named *PE Sample 5.0*.

When the workspace has opened up, you may see a large number of errors in the Tasks view. In our experience these errors are usually due to classpath references, and can be fixed as follows:

1. From the menu bar choose **Project** → **Rebuild All**.
2. If there are still some errors when the rebuild is finished, you should once again choose **Project** → **Rebuild All**.

We found after this second rebuild, we had an error-free workspace.

If you do still have errors with classpath references, then you will need to correct the classpath of the offending projects in the project properties dialog.

From the Java perspective, right-click the project in error, then choose **Properties** (the last item at the bottom on the menu). This brings up the properties dialog. You choose **Java Build Path** in the left pane then the **Libraries** tab in the right pane, as can be seen in Figure 6-2 on page 198. Make any necessary changes to point to the correct libraries.
6.2.3 Create a test server

To test the application within Studio, we use the built-in WebSphere Version 5.0 EE Test Environment. This means we need to create a sever and server configuration:

1. From the Server perspective, right-click Servers in the Server Configuration view. Then choose New → Server and Server Configuration, as in Figure 6-3 on page 199.
Figure 6-3  Create a new server and server configuration

2. You may be asked Do you want to create a new server project with the name Servers? If you see this question click Yes.

3. In the Server and Server Configuration dialog, give the server a name (we used test) and make sure you select **WebSphere version 5.0 EE Test Environment** as in Figure 6-4 on page 200, then click **Next**.

    **Important:** You must choose the EE Test Environment in order to get the **Business Process Environment** (BPE), which is the container that business processes run in.
Using BPEL Processes in WebSphere Business Integration Server Foundation - Business Process Integration and Supply Chain Solutions

6.2.4 Deploy the application EAR files

Now that we have the test environment installed and configured, it is time to deploy the EAR files that make up the application to the newly created server.
As described earlier, there are five EAR files that make up this sample application:

- AggServiceEAR
- MessageTester
- EIS2_SRVEAR
- EIS3_SRVEAR
- ExchangeStub2EAR

1. From the Server perspective right-click the test server in the Server Configuration view, then click Add → **AggServiceEAR**, as in Figure 6-6.

![Figure 6-6 Deploy the Aggregation service to the test server](image)

The other four EAR files are deployed in a similar manner to the AggServiceEAR and should also be deployed at this time.

2. Once all the EAR files are deployed, you now need to deploy the business processes to the server. In the Servers view right-click the server and select **Deploy Process**, as in Figure 6-7 on page 202.

In our sample this deploys the processes that are found in the AggServiceEAR, but if there had been processes in the other EARs then they would also have been deployed in this step.
6.2.5 Configure the test server

To run the applications, we need to configure the following in the test server:

- Enable security.
- JMS connection factory.
- Destination queues.
- Listener ports.

Most of these properties can be set in the test server configuration. This can be opened from the Server Configuration view; right-click the test server and choose Open (you can also just double-click the test server). This will open the test server configuration file and should look something like Figure 6-8 on page 203.
Figure 6-8  Test server properties file

Enable security
To enable security:

1. In order to use staff activities, security must be enabled. This is set under the Security tab. Enable security on this page, as in Figure 6-9 on page 204.

   **Important:** Do not select J2EE security. This is difficult to configure and not needed in for our installation.

2. Make sure the Enable Security checkbox is ticked. In the User ID and Password fields, use an id that has administrator access to the system.

   **Attention:** On Windows, this user ID must also have the “Act as part of the operating system” privilege. This allows WebSphere Application Server to validate the user ID and password with the operating system.
3. While on this page, you also need to set the JAAS Authentication Entry. There should already be an entry with an alias of BPEAuthDataAliasEmb_localhost_server1. You should edit this entry and use the same user ID and password entered when enabling security above.

4. You can also create a new alias by clicking Add and filling in the fields as appropriate. In most places throughout the server configuration, when you are asked for component or container managed authentication alias, your new alias will appear in a drop-down list for selection.

**JMS connection factory**

For the JMS connection factory:

1. The application uses a connection factory named jms/QCF, which should be installed and configured by default. If this connection factory is not already configured then you need to configure it manually.

2. This is set in the properties file under Websphere JMS Provider Options, which you can get to by clicking the JMS tab along the bottom of the server configuration editor. Scroll about half way down the page and you should see the JMS Connection Factories section. If jms/QCF is already defined you will see something like Figure 6-10 on page 205.
Figure 6-10  JMS Connection Factories list with jms/QCF defined

3. If jms/QCF is not already defined, click Add and configure it as in Figure 6-11 on page 206, then click OK.

**Note:** If you created an authentication alias with a different name, then you should choose that in the authentication fields.
Figure 6-11  Configuration of jms/QCF

**Destination queues**

For the destination queues:

1. At the bottom of the page in the JMS tab of the server properties, will find the JMS Destinations. You need to configure two destinations, jms/PEAGGIN and jms/PEAGGOUT.

2. To create the PEAGGIN destination click **Add** and fill in as in Figure 6-12 on page 207 (using the left-hand screenshot), then click **OK**.

3. You should then create PEAGGOUT in the same way, using the right-hand screenshot in Figure 6-12 on page 207 for the definition.
Listener ports
The listener ports are set in the EJB Container under the EJB tab of the properties file.

1. You need to have a listener port named SenderQListenerPort. This is usually created by default when you create the test server. If it is already created then you need to modify it to point to a different queue. If it is not already defined then you need to create it.

2. If the SenderQListenerPort is already defined as in Figure 6-13 on page 208, then you need to modify it by clicking SenderQListenerPort, then click Edit. You should then choose jms/PEAGGIN from the Destination JNDI Name pull-down, as in Figure 6-14 on page 208, and click OK.
3. If the SenderQListenerPort is not already defined click Add and fill in the values as seen in Figure 6-14, then click OK.

6.2.6 Publishing and starting the server

To publish and start the server:

1. In the Server view, you will now see that the system tells you that the server should be republished under the Server State heading, as shown in Figure 6-15 on page 209.
2. To publish the application to the server, you right-click the test server and select **Control → Publish**, as in Figure 6-16.

3. When the application has been published to the server, you should see the **Server State** changed to **Server is synchronized**.

4. If the **Status** of the test server is **Stopped**, then start the server by right-clicking the server in the **Server view** and choosing **Start**.

5. If the **Server State** needs to be restarted, then right-click the server in the **Server view** and choose **Restart**. See Figure 6-17 on page 210.
6.2.7 Testing the deployed application

There are two scenarios you should test to confirm that the deployment has been successful.

**Testing the approval process**

To test the approval process:

1. Create a new ITEM on the PEAGGIN queue.
2. Open a Web browser using http://<hostname>:9080/MessageTesterWeb/index.html, which will take you to the page shown in Figure 6-18 on page 211.
Chapter 6. Deploying the existing sample solution

3. When you click **Send Message** a XML message is placed on the PEAGGIN queue.

4. The Aggregation application has a Message Driven Bean with a listener attached to the PEAGGIN queue. It picks up the message and starts the Approval business process, which requires the created ITEM to be either approved or rejected.

5. Approve the ITEM using the BPE Web Client.

6. Before starting the BPE Web client, you must first create a user ID in the local operation system called **aggclerk** with any password you choose. For simplicity, this should be granted administrator access.

7. To see a list of ITEMS awaiting approval, open a Web browser and go to http://<hostname>:9080/bpe/webclient. You will initially be presented with a dialog for user ID and password. Enter the user ID **aggclerk** and password that you set for this ID earlier. You should now see a page similar to Figure 6-19 on page 212, with the staff activities awaiting your approval.
8. If there are no staff activities listed it may just be that the process has not yet created the staff activity. Wait a few moments and click My To Dos in the left-hand pane to refresh the screen (you may need to do this several times).

9. When the staff activity is in the page, click the Staff link for the activity you want to work on. You will now see a page similar to Figure 6-20 on page 213.
10. Click **Claim** to assign the call to the aggclerk ID, which will allow you work on the activity.

11. When you have claimed the activity to work on, you are now presented with a screen that looks similar to Figure 6-21 on page 214. On this page you are presented with the information about the **ITEM**, which you should see corresponds to the data that was hardcoded in itso.demo.WriteQ.java.

12. If you scroll to the bottom of this page, you will see two radio buttons, **Approve this part** and **Reject this part**. Click **Approve this part**, then scroll back to the top of the page and click **Complete Activity**.

13. You will now be back at the My To Dos page and that activity will no longer be available.

14. By completing the activity you have resumed the approval business process, which is now creating entries to send to the Exchange server as well as to the PEAGGOUT queue, which would feed to a commerce server a real-world completed application.

If you rejected the part, the activity would still not be available in the My To Dos, but unlike approval, the part would not be sent to the exchange server or PEAGGOUT.
15. View the new ITEM in the exchange server.

   To view ITEMs that have been sent to the exchange server, open a Web browser and go to http://<hostname>:9080/ExchangeStub. You will see a page similar to Figure 6-22 on page 215.

16. Make sure the **getAllDescriptions** radio button is selected, then click **Submit Query**. You should see the part you just approved.

   **Attention:** You may not see the nicely formatted table at the bottom of the Web page for the approved part. There is a bug in the sample code, which means that sometimes you just see the raw data presented as one long line. You can still see the data you approved, it is just a little harder to pick out.
Figure 6-22  Exchange server

When you see your created ITEM go all the way through the approval process and then get displayed in the exchange server, you have successfully deployed the approval process and related applications.

Testing the mass update process
There are only two steps to testing the mass aggregation process:

1. Schedule mass update to run.

To schedule the mass update process to run, open a Web browser and go to http://<hostname>:9080/SchedulerStatus.jsp. This will bring you to the page showing the current schedule for running mass aggregation. A schedule has yet to be set, so you should see a page like Figure 6-23 on page 216.

2. To set a schedule for mass update to run click Schedule Task.
3. Figure 6-24 shows the scheduling page. Filling in a starting time of 10 seconds will make the process run 10 seconds. Putting a repeat interval of 10 days will have it run every 10 days after the first invocation.

4. You can enter any times you like, but these are good values for testing because the process will start soon and only run once. We suggest that if you want to run it again you should purge the task and re-enter it as shown in Figure 6-24. Purging the task is discussed later in this chapter.
5. Once you are happy with the times click **Submit**, which will submit the schedule and should take you to the page shown in Figure 6-25.

![New Schedule Submitted](image)

**Figure 6-25  A successfully submitted schedule**

6. If you now click **View Scheduler Status** you will be taken to a page similar to Figure 6-26, which shows the schedule set.

7. You also now have a link for **Purge Scheduled Task**, which if clicked will purge the task and take you back to the screen in Figure 6-23 on page 216, from which you can set a new schedule.

![Scheduled Process Status](image)

**Figure 6-26  View scheduler status and purge task**
8. Check the PEAGGOUT queue to see that the data has been sent.

9. The best way to see if the mass update process is running is to look at the messages on stdout (see “Tip: For viewing stdout:” on page 220). You should see a number of messages similar to those shown in Example 6-2.

**Example 6-2  Sample stdout message for mass update process**

<table>
<thead>
<tr>
<th>SystemOut</th>
<th>0 Mass Update: Preparing to get all parts from EIS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Initializing data</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Data initialized</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Item Passed in &lt;%&gt;</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Found key: IN1111113</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Found key: IN1111112</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Found key: IN1111111</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: Returning the following XML string: &lt;?xml version=&quot;1.0&quot;?&gt;&lt;ExchangeDBroot&gt;&lt;Part&gt;&lt;ItemNumber&gt;IN11111113&lt;/ItemNumber&gt;&lt;Price&gt;23.98&lt;/Price&gt;&lt;/Part&gt;&lt;Part&gt;&lt;ItemNumber&gt;IN11111112&lt;/ItemNumber&gt;&lt;Price&gt;10.20&lt;/Price&gt;&lt;/Part&gt;&lt;Part&gt;&lt;ItemNumber&gt;IN11111111&lt;/ItemNumber&gt;&lt;Price&gt;2.05&lt;/Price&gt;&lt;/Part&gt;&lt;/ExchangeDBroot&gt;</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 -EIS2: getPartPrice finished</td>
</tr>
<tr>
<td>SystemOut</td>
<td>0 Mass Update: processing item 1 of 3</td>
</tr>
</tbody>
</table>
| WebGroup  | I SRVE0180I: [ExchangeStub2Web] [/ExchangeStub] [Servlet.LOG]:
| rpcrouter: | init                                                  |
| SystemOut | 0 Attempting to read ExchangeDBFile.xml               |
| SystemOut | 0 Aggregation: aggregating item: IN1111113 (null)....|
| SystemOut | 0 -EIS3: Initializing data                             |
| SystemOut | 0 -EIS3: Data initialized                              |
| SystemOut | 0 -EIS3: Returning the following XML string: <?xml version="1.0"?><ExchangeDBroot><Part><ItemNumber>IN11111113</ItemNumber><Quantity>34</Quantity></Part></ExchangeDBroot> |
| SystemOut | 0 -EIS3: getPartQuantity finished                     |
| SystemOut | 0 -EIS2: Item Passed in <IN11111113>                  |
| SystemOut | 0 -EIS2: Found price <23.98                          |
| SystemOut | 0 -EIS2: Returning the following XML string: <?xml version="1.0"?><ExchangeDBroot><Part><ItemNumber>IN11111113</ItemNumber><Price>23.98</Price></Part></ExchangeDBroot> |
| SystemOut | 0 -EIS2: getPartPrice finished                        |
| SystemOut | 0 Aggregation: EIS3 result: 34                        |
| SystemOut | 0 Aggregation: EIS2 result: 23.98                     |

<Messages deleted>
If no error messages are shown and you see the last line saying Mass Update: loop complete, then the process worked.

10. You can also check the PEAGGOUT queue by going to http://<hostname>:9080/MessageTester/index.html and clicking Get Message. This will read the first message off the PEAGGOUT queue, which should correspond to one of the three parts that the mass update handled. You should be able to run Get Message three times to see all three messages. Figure 6-27 shows an example.

![Figure 6-27 Message off PEAGGOUT](image-url)
6.3 Sample code deployment on WebSphere Application Server Enterprise V5.02

We now deploy the sample application onto WebSphere Application Server Enterprise V5.02 using a single machine for all components, as described in “Deployment architecture” on page 196.

6.3.1 Prerequisite software and settings

This section details the required software and configuration settings need to install and test the sample application on WebSphere Application Server Enterprise.

Software
To be able to deploy and test the sample code, you need to following software installed on your machine:

- WebSphere Application Server Enterprise V5.02
- WebSphere MQ V5.3
- IBM DB2 V8.1

For testing, you also need a Web browser. We used Microsoft Internet Explorer V6 sp1, but any modern Web browser should also work.

Settings
You need several user IDs defined to the local operating system:

1. The first user ID needed is for the security setting within the application server and can be called anything you like and should have administrator privileges.

Tip: For viewing stdout:

During all of these processes, lots of messages are sent to stdout by the application server. To view these under WebSphere Studio Application Developer Integration Edition, you should look at the Console view. You will see lots of debug messages that will also show you that the process is working (or error messages showing you it is not).

To view stdout under the production environment you should look at the file WAS_HOME/logs/<servername>/SystemOut.log. It is best to use a viewer that automatically updates when the file has changed, or else you will need to keep closing and reopening the file.
2. The second user ID is for the approval process and must be called aggclerk. We also gave this user administrator access to make security easier to configure.

**Important:** The user ID used for security settings should also have the Act as part of operating system privilege. We found that if this was not done, you get errors saying the application server could not verify user IDs and passwords.

### 6.3.2 Preparing the application server to run processes

Before installing any business processes on an application server, you must first configure the WebSphere Business Process Container. This must be done on each application server you wish to run business processes on.

We provide a brief overview of the process to configure the Business Process Container in this chapter. For detailed instructions see the WebSphere Application Server Information Center under **WebSphere Application Server Enterprise** → **All topics by feature** → **Applications** → **Service choreography** → **Using Process Choreographer**.

**Planning**

Before configuring the Business Process Container, you must consider the following issues:

- Which database system will be used.
- Which machine will host the database.
- If not creating a WebSphere cluster setup, you need to decide which JMS messaging service you will use:
  - The embedded messaging in WebSphere (needs to have been selected at install time)
  - An external message product (such as WebSphere MQ)
- If you plan to run in a WebSphere cluster environment, there are other planning considerations for Business Process Choreographer. Consult the WebSphere Information Center for details.
- Decide whether to use the Install Wizard or manually configure the Business Process Container.

**Tip:** We found the Install Wizard much simpler to use.
For our simplified architecture, we use DB2 locally and WebSphere MQ. The instructions below are an abbreviated version of those found in the WebSphere Information Center.

### Configuring DB2
This assumes DB2 is already installed and ready to run.

1. Open a DB2 command window.
2. Change to the directory with the Business Process Choreographer setup scripts.
   ```bash
   cd %WAS_HOME%\ProcessChoreographer
   ```
3) Create a database named BPEDB.
   ```bash
   db2 -tf createDatabaseDb2.ddl
   ```
4) Make sure the CLI packages are bound to the database.
   ```bash
   db2 connect to BPEDB
   db2 bind %DB2PATH%\bnd\@db2cli.lst blocking all grant public
   ```
3. Create the tablespace
   ```bash
   db2 -tf createTablespaceDb2.ddl
   ```
4. Create the schema
   ```bash
   db2 -tf createSchemaDb2.ddl
   ```

### Configuring WebSphere MQ
To configure WebSphere MQ:

1. Make sure that your user ID has the authority to create WebSphere MQ queues.
2. Create the queue manager and queues:
   ```bash
   cd %WAS_HOME%\ProcessChoreographer
   createQueues.bat <queueManager>
   ```
   Where `<queueManager>` is the name of an existing queue manager, or the name to be given to a new queue manager. If the named queue manager already exists, it will be used to create the queues. If the queue manager does not exist, it will be created and started before the default queues are created.

   We created a new queue manager with the name WAS_BPE_<hostname>.
3. You also need to create two queues that will be used by the processes later:
   - PEAGGIN
   - PEAGGOUT
4. Start WebSphere MQ Explorer by clicking **Start → Programs → IBM WebSphere MQ → WebSphere MQ Services**. Now right-click the queue manager and choose **Properties**. On the General tab there is a field called Startup; change this to Automatic, then click **OK**.

### Installing the Business Process Container

Now that DB2 and WebSphere MQ are both set up, you need to run the Business Process Container Install Wizard.

To start the wizard do the following:

1. Start the WebSphere Administrative Console (this assumes that you have already started the WebSphere Application Server from either first steps or the command line).
2. Click **Servers → Application Servers → <server_name>**.
3. In the Additional Properties section, click **Business Process Container**.
4. Near the bottom of the page click the link for the **Business Process Container Install Wizard**.

There are three steps that the wizard requires to be completed.

**Attention:** In the following configuration, we used the same user ID that had administrator privileges for all instances that needed a user ID and password.

The first step for database configuration and should be configured as follows (see Figure 6-28 on page 224 as a sample for our configuration):

1. Select **Create a new XA datasource**.
2. In the drop-down list, select the database you are using.

3. For the Implementation class name use the default class name provided for the JDBC driver implementation.

4. For Classpath enter the location of the Java archive or zip file for the DataSource.

5. The DataSource user name must be a user ID that has the authority to create and administer the database.

6. Enter the DataSource password for the DataSource username.

7. Click **Next** to go to the next step in the Install Wizard.

---

<table>
<thead>
<tr>
<th>Step 1: Select the Database Configuration for the Business Process Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select an existing resource or enter information for new resources which will be created by the wizard.</td>
</tr>
<tr>
<td>- Create a new XA datasource</td>
</tr>
<tr>
<td>- Implementation Classname</td>
</tr>
<tr>
<td>- Classpath</td>
</tr>
<tr>
<td>- Datasource User Name</td>
</tr>
<tr>
<td>- Datasource Password</td>
</tr>
<tr>
<td>- Custom Properties</td>
</tr>
<tr>
<td><img src="image" alt="Image of configuration screen" /></td>
</tr>
</tbody>
</table>

**Figure 6-28 Business Process Container install wizard Step 1**

The second step is for JMS provider and security configuration and should be configured as follows (see Figure 6-29 on page 225 as a sample for our configuration):

1. In the drop-down list for JMS provider, select the messaging service that the Business Process Container will use.

2. For the Queue manager, use the default provided (WASQM or WAS_hostname_server).

3. You must also define the WebSphere environment variable `${MQ_INSTALL_ROOT}`. Make sure that the Classpath points to the WebSphere MQ Java lib directory.
4. For the JMS user ID, enter a user ID that has administration rights for the messaging service.

5. For the JMS password, enter the password for the JMS user ID.

6. For the Scheduler calendar field, if you have your own scheduler calendar, enter its JNDI name. Otherwise, if you leave it blank, the default value, BPEScheduler, will be used.

7. For the Security role mapping, enter the user or group from your user registry that is to be mapped onto the role of Business Process Administrator.

8. For the JMS API user ID, enter the user ID that is to be used when processing asynchronous API calls.

9. For the JMS API password, enter the password for the JMS API User ID.

10. Click Next to go to the next step in the Install Wizard.

---

**Figure 6-29** Business Process Container install wizard Step 2

Step 3 is for JMS resource configuration and should be configured as follows (see Figure 6-30 on page 226 as a sample for our configuration):

1. Select existing JMS resources.

2. Use the Connection Factory drop-down list to select **BPECF**.
3. Use the Internal queue drop-down list to select **BPEIntQueue**.
4. Use the External request processing queue drop-down list to select **BPEApiQueue**.
5. Use the Hold queue drop-down list to select **BPEHldQueue**.
6. Use the Retention Queue drop-down list to select **BPERetQueue**.
7. Click **Next**.

![Figure 6-30 Business Process Container install wizard Step 3](image)

The last step provides a summary of what you have entered as well as a reminder that there are some things you need to manually configure. We have already done all the things that need manual configuration when configuring DB2 and WebSphere MQ. You can now click **Finish** to install the container.

The progress is shown on the Installing page:

1. If the container did not install successfully, check for any error messages that can help you correct the problem, then repeat the installation by starting the wizard again.
2. If the container was installed successfully, click **Save Master Configuration**, then click **Save**.

### 6.3.3 Configuring the application server

We now need to configure the application server to enable security, set up server resources, and more. This is all done from within the WebSphere Administrative
Console, which can be started either from the First Steps panel, or by opening a Web browser and going to http://<hostname>:9090/admin. This assumes that you have first started the application server.

## Enable security

There are two steps to enabling security:

1. **Set up the User Registry for local OS.**
   a. Go to **Security → User Registries → localOS**.
   b. You will be presented with a page where you need to type a user name and password that has administrator rights on their local operating system. We also suggest that this user ID also has the Act as operating system privilege as well. See Figure 6-31.

   ![Figure 6-31 Setting user registry](image)

2. **Enable security:**
   a. Go to **Security → Global Security**.
   b. The only thing you need to do is make sure **Enabled** is selected and **Enforce Java 2 Security** is not selected. The page should look like Figure 6-32 on page 228.
   c. Click **OK**.

   **Important:** When you select **Enabled**, **Enforce Java 2 Security** will automatically become selected as well. You **must** clear **Enforce Java 2 Security**.

   If you leave **Java 2 Security** selected, you will have a lot of issues with access to files when you to and start the application server, and the server will not be able to start.
Figure 6-32  Enable security

Queue connection factories
For queue connection factories:

1. Go to Resources → WebSphere MQ JMS Provider, scroll down the page, and click WebSphere MQ Queue Connection Factories.

2. We need to create a new connection factory called QCF, so click New. You can leave most fields as defaulted with the following exceptions:
   - Name: QCF
   - JNDI Name: jms/QCF
   - Queue Manager: Put in the name of the queue manager that you previously created.

3. Now click OK.
Queue destinations
For queue destinations:

1. Go to Resources → WebSphere MQ JMS Provider, scroll down the page, and click WebSphere MQ Queue Destinations.

2. Here we need to create destination queues that correspond to the two queues you created in WebSphere MQ earlier. Click New and again leave the defaults with the following exceptions (Figure 6-33 on page 230 has an example page for creating PEAGGIN):
   - Name: PEAGGIN
   - JNDI Name: jms/PEAGGIN
   - Base Queue Name: PEAGGIN

3. Click OK.

4. Do the same steps for the other required queue.
   - Name: PEAGGOUT
   - JNDI Name: jms/PEAGGOUT
   - Base Queue Name: PEAGGOUT
Figure 6-33   Creating PEAGGIN destination queue
Message listener port

For the message listener port:

1. Go to **Servers → Application Servers → <servername>**, scroll about one third of the way down the page, and click **Message Listener Service**.

2. Click **Listener Ports**.

3. We want to create a listener port called SenderQListenerPort, so click **New**. Fill in the fields as shown in Figure 6-34. Then click **OK**.

![General Properties](image)

**Figure 6-34  Create SenderQListenerPort**

Save the configuration

Now that you have configured the application server, you should save to the Master Configuration by clicking **Save** at the top of the admin console. Then click **Save** again when prompted.

6.3.4 Deploy the applications

In this environment you must install the five applications as described in the test environment discussion in 6.2, “Sample code deployment in WebSphere Studio Application Developer Integration Edition V5.0” on page 196. You also need to install the EIS1 application. You do not need to install both MessageTester and EIS1EAR, but, as explained in “EIS1EAR” on page 193, we recommend
installing both applications, for the flexibility this provides. The EAR files to install are:

- AggServiceEAR
- EIS2_SRVEAR
- EIS3_SRVEAR
- ExchangeStub2EAR
- MessageTester
- EIS1EAR

We will explain how to install AggServiceEAR in detail. The other applications use the same technique, just using a different EAR file.

1. In the WebSphere Administrative Console click **Applications → Install New Applications**. You will see a page like Figure 6-35.

![Figure 6-35 Installing and application - Specify ear file](image)

2. You need to specify the location and name of the EAR file. *Local Path* is local to the machine you are running the admin console on. *Server Path* is a path on the server the application server is installed on. When done click **Next**.

3. Leave the defaults on this screen and click **Next**.

4. You need to make sure both **Deploy EJBs** and **Deploy Web Services** are selected, as in Figure 6-36 on page 233. Click **Next**.
Figure 6-36  Application install Step 4

5. You can leave the defaults for the rest of the steps, so continue to click Next until you get to step 12, which should be the summary and look similar to Figure 6-37 on page 234.
You should now click Finish to install the application.

7. Install the other applications in the same manner as this one (there is an additional step for EIS1 as detailed below). You can take the defaults on all pages except remember to select Deploy EJBs and Deploy Web Services.

8. When installing EIS1, there is an additional step for mapping security roles to users. EIS1 uses a user role called EIS1UserRole, which must be mapped to allow users to access the Web page. Step 4 during the install allows you to do this. We granted access to everyone, as can be seen in Figure 6-38.

![Figure 6-37 Install Step 12 summary](image1)

![Figure 6-38 Mapping user role to everyone](image2)
6.3.5 Starting and stopping the server and applications

Now that everything is configured and deployed, you should save to Master Configuration one last time, and then reboot the server.

**Starting**
When the server has been rebooted, you can now start the application server again. This can be done in the normal way from either First Steps or from the command line with the following command:

```
startserver <servername>
```

**Stopping**
With security enabled, you must now supply the administrator user name and password that you defined when enabling security. This can be done from the command line with the following command:

```
stopserver <servername> -user <admin user> -password <admin password>
```

6.3.6 Test sample code

Testing can be done with the same procedure as previously seen in “Testing the deployed application” on page 210. The one additional step here is the use of EIS1 to initiate the approval process.

**Testing with EIS1**
To test with EIS1:

1. Open a Web browser and go to:

   `http://<hostname>:9080/EIS1/MQServletAInputForm.html`

   This should bring up a page similar to Figure 6-39 on page 236.
2. Enter the specified data (no data checking is done, so anything can be entered). Make sure Create is chosen at the bottom of the page and click Submit.

3. This produces the same effect as running Send Message in MessageTester, so you can follow the rest of the instructions in testing under the development environment.

### 6.4 Troubleshooting

This section is a short list of problems we ran into and what we did to fix them. They may not be directly applicable to the sample, as we fixed all of the problems we encountered, but are included to illustrate some of the common problems we ran into.
6.4.1 Changing deployment descriptors for the Web service

The exchange server was originally designed to be accessed via a Web Services Gateway. We changed this to simplify the installation. The following sections describe what we did to make this change, and also what you could do to change it back to using a Web Services Gateway.

Removing the AggService application

To remove the AggService application:

1. The first thing you have to do is stop the AggServerEAR application.
2. Using the WebSphere Administrative Console choose Applications → Enterprise Application, select the AggServiceEAR application, then click Stop.
3. Once the application is stopped, you must stop the templates.
4. Go to Applications → Enterprise Application → AggServiceEAR then scroll down and choose Business Process Modules.
5. Click AggService.far.
6. Click Templates. This will bring you to a page that should look like Figure 6-40.

![Figure 6-40 Stopping templates](image)

7. Make sure all the templates are selected and then click Stop. When all the templates are stopped, you should save the change to the Master Configuration.

You must now delete some rows out of the process_instance_t table in the BPEDB database.

1. Start a DB2 command prompt by choosing Start → Programs → IBM DB2 → Command Line Tools → Command Window.
2. Enter the following DB2 commands:

   `db2 connect to bpedb`
   `db2 delete from process_instance_t where application_context254 like 'AggServiceEAR%'
   db2 connect reset`

3. The last step is to actually uninstall the AggServiceEAR. Go to **Applications → Enterprise Applications**. Tick the AggServiceEAR and click **Uninstall**.

4. When the uninstall is complete save to the Master Configuration.
Migrating to WebSphere Business Integration Server Foundation

In this chapter we discuss how to migrate FDML business processes from WebSphere Studio Application Developer Integration Edition v5.0 into the newest BPEL business process specification supported in WebSphere Studio Application Developer Integration Edition V5.1. We cover the initial steps of migrating the workspace, using the automated migration tooling to convert our process, any additional manual steps required for the transition, and how to deploy BPEL processes to the WebSphere Studio Application Developer Integration Edition V5.1 run-time test server.
Note: This chapter describes the steps that we took to migrate our specific sample application. In our redbook additional material we do provide a workspace with the original application so that you can look at following the steps in our migration process. However, please remember that migration is a complex process and it does help if you have a good understanding of the original application. While we have tried to record all the steps of our migration, this was a lengthy process and it will not be easy for you to repeat every step we document. We do not expect that many readers will actually want to migrate our application; rather it is our intention that you use this chapter to get a feel for the migration issues and their solutions that you may encounter in your own work.
7.1 Importing an existing workspace

Before tackling the more challenging issues of migrating our existing FDML process we must first address any possible low-level breakages that may take place as we migrate to a WebSphere Studio Application Developer Integration Edition V5.1 environment.

The most common migration issues in any project are those related to changes in external jars and libraries, or those problems related to deprecated, unsupported, or newly created API functions. We address these issues one at a time as they relate to our example so that in later sections we have the correct environment necessary to successfully migrate our business processes.

7.1.1 Opening the Version 5.0 workspace

We begin by migrating the existing WebSphere Studio Application Developer Integration Edition V5.0 workspace. Upon starting WebSphere Studio Application Developer Integration Edition V5.1 you will be prompted for the directory location of the workspace with which you want to invoke the development environment. Enter the location for the directory where your WebSphere Studio Application Developer Integration Edition V5.0 workspace exists, for example, “C:\PE Sample 5”. Click OK to continue.

Note: A sample workspace containing the existing application is included in our redbook additional material. For details on obtaining the additional material see Appendix B, “Additional material” on page 319. Further details about using the sample workspace are in Appendix A, “Working with the redbook sample code” on page 315.

During the startup of Studio, an informational box will be displayed informing you that the workspace you have referenced was created with a different version of Studio other than V5.1. From this window you can choose to continue loading Studio, with the added change that your workspace will be migrated to the current V5.1 level. See Figure 7-1 on page 242.
7.1.2 Resolving broken library links and external JAR file locations

After your updated workspace loads, you will most likely be presented with at least some errors or warnings displayed in the Tasks view. For example, the view of your newly migrated workspace should look similar to Figure 7-2 on page 243.

**Note:** This workspace migration is irreversible, and the updated workspace will not be compatible with Version 5.0 WebSphere Studio environments. In the event that migration fails, or if you have a need to use the Version 5.0 environment, it is a good idea to make a backup copy of your workspace before attempting this migration.
The easiest of these errors to resolve are those related to broken library links and other entries in a project's classpath that may no longer resolve correctly. Before resolving any of the errors seen, select Project → Rebuild All to refresh the project space and remove any trivial errors.

1. Next, we break down fixing the errors into the projects they belong to.

2. In the ExchangeStub2Web project you will notice two warning messages regarding the com.ibm.soap.server.http.WASRPCRouterServlet and com.ibm.soap.server.http.WASMessageRouterServlet classes. The warnings for these classes can be resolved by including the wssoap.jar library in the ExchangeStub2Web build path. The wssoap jar is located in the WAS_EE_V51/lib directory. For details on how to add JARs to a project's build path see 5.7.6, “Setting a project build path” on page 176.

3. You will notice multiple errors for the resources in the EIS1 project. Although there are multiple errors they can all be resolved by two changes:
   a. First, add the XSDBEANS variable to the EIS1 build path.
   b. Second, add the WAS_EE_V51 variable to the build path of EIS1 and extend the variable with /mqjms/java/lib/com.ibm.mq.jar by clicking
Extend, and navigating to and selecting `mqjms/java/lib/com.ibm.mq.jar`. When you are finished, the EIS1 build path should look like the one shown in Figure 7-3.

![Figure 7-3 Migrated EIS1 build path](image)

### 7.1.3 Unnecessary BPE projects

Delete the following unnecessary BPE projects from your workspace:

- BPEContainer
- bpecontainer_ejb
- BPERemoteDeploy
- bperemotedeploy_ejb
- bpessoapclient
- bpewebclient
- compensate_ejb

We are not using the capabilities provided by `compensate_ejb` for process compensation. The remaining BPE-related projects of our WebSphere Studio Application Developer Integration Edition test run time will be generated for us automatically as they are needed.
7.2 Migrating the aggregation and approval business processes

The next task is to migrate the aggregation and approval business processes.

7.2.1 Automatic migration of the aggservice project

For automatic migration of the aggservice project:

1. With the PE Sample 5 workspace loaded into WebSphere Studio Application Developer Integration Edition you must first navigate to the Java perspective and right-click the AggService project.

2. From the displayed menu, click the Migrate command as shown in Figure 7-4.

![Figure 7-4: Invoking the auto-migration of FDML business processes](image)

3. Since you selected to run the migrate command against an entire project, you are presented with a list of all of the business processes defined in the project, as seen in Figure 7-5 on page 246. All these processes need to be migrated.

Attention: If the migrate command fails to display the window above, it may have to do with the V5.0 to V5.1 workspace migration you first performed. To resolve the issue delete the AggService project from your workspace, taking care not to delete the project from the file system itself. After doing so, import the AggService directory back into your workspace as an already existing project, accepting all of the defaults. Perform a rebuild of all projects in the workspace and try the migrate command again.
4. Click **Next** to continue.

5. At the next screen, Figure 7-6 on page 247, you have the option to specify your own project name for the newly migrated project, or to accept the default project name. The default project name is generated by appending MigrationV5 to the original name of the project.
6. Finally, click the Finish button to execute the automatic migration process.

7.3 Resolving possible migration errors

After running the above automatic migration command, depending on the functions used and the design of your FDML process, there may be unresolved errors or warnings listed in the Tasks window.

In our example there will be three errors displayed after the migration. The errors shown can be broken down into two different types, staff activity related errors and response variable related errors.

7.3.1 Staff activities

Staff activities, as they are known in BPEL processes, are by nature synchronous activities. Under FDML, processes with these activities were defined as interruptible. Since the activity is a staff activity it also dictates whether a process is long or short running. To resolve the staff activity error we have received we must manually set the process as long running.

1. Double-click the BPED0251E: Non-interruptible process ‘approval’ must not contain asynchronous activity ‘Staff’ error message listed in the Tasks
window to display the approval.bpel process. From the process screen, click the approval process name, and then click the Server listing at the bottom of the window frame. Check the second checkbox entitled “Process is long-running” to mark the process as long running, as seen below in Figure 7-7.

Figure 7-7 Enable a long-running process

2. Save the process once you have made these modifications.

7.3.2 Missing activity output variables

Another error that may be introduced in the conversion of FDML processes to BPEL processes is due to differences in requirements of the processes. In FDML it was possible to invoke external activities without assigning the returning output result to a local variable inside the calling process. It created no errors and would not result in abnormal behavior. For BPEL processes, the interfacing between various activities has been changed. According to the BPEL specifications, if an activity is a two-way process, meaning both a request and response are passed
between the activities, you must assign the output of the activity that you are calling to a local variable inside the calling process. You cannot invoke a two-way process if you do not assign a variable to accept the response that the process will generate. For instance, you will notice that there are two errors listed similar to the following format:

“BPED0264E: Mandatory output variable for activity ‘Aggregation’ of process ‘massUpdate1’ is missing.”

To correct this problem:

1. Double-click the error message to open the BPEL process model in the BPEL editor.

2. Click the activity inside the process that has an error, and select the Implementation details tab for that activity. See Figure 7-8.

3. The Response field is listed as None, indicating that no variable has been assigned to this activity’s response message.

![Figure 7-8 Aggregate activity without an associated response variable](image)
4. Click the plus sign next to the Variables listing in the BPEL editor.

5. Enter in a name for the new variable. In our case we will use aggregateResponse, as in Figure 7-9.

![Figure 7-9 Creating a new BPEL variable](image)

6. This variable does not need to correspond to any critical value since in our previous FDML implementation this response value was simply being ignored. In essence, we call the aggregation process not for its return value, but for the actions it performs to the data in our environment.

7. After creating the new variable, you should see its name added to the Variables list and you can now assign it a message type.


9. Select the Int message type, as shown in Figure 7-9.

10. Return to the Implementation tab for the aggregate process and select the aggregateResponse variable for the process’s Response field. Save the BPEL process, and your output should look similar to that in Figure 7-10 on page 251.
11. Repeat this process for the remaining activities that do not have variables associated with their output.

### 7.3.3 Mass update process names

Ensure that the mass update process name matches the name being used in the WSIF call in the MassUpdatePortTypeServiceHome.java file inside AggSchedulerEJB. If these names are different you will get an error when the process is run.

1. To check for this condition open the massUpdate1.bpel process file, located inside AggServiceMigrationV5 in the BPEL editor.
2. Select the **massUpdate1** process name icon and inspect the Name field inside the Description tab.
3. If it is listed as massUpdate1, as shown in Figure 7-11, change the name to massUpdate so that it matches the invocation found in MassUpdatePortTypeServiceHome.java.

![Diagram of massUpdate1 BPEL process]

Figure 7-11 Incorrect massUpdate name “massUpdate1”

7.3.4 Resolving approval process invocation errors

In the previous FDML versions of our processes, the aggregation and approval processes shared a common WSDL description, the /wsdl/ProductAggregation.wsdl file inside AggServiceMigrationV5. The two processes require the same inputs, generate similar outputs, and can share all of the same message types. It was previously not a problem for two processes that are bound to different transports to share one port type, but after the migration to BPEL this is a problem.

Since the aggregationMDB Message Driven Bean is responsible for invoking the approval process and because it is generated code, we try to keep changes to it as small as possible.

1. We will only change the WSDL port types we have defined for the aggregation and approval processes.

2. Back up the aggregation Message Driven Bean WSDL files before making any changes.

3. We will edit the WSDL file describing the approval activity, which is also imported by WSDL files in the aggregation MDB project. Before we edit the approval activity WSDL, we first copy ProductAggregation.wsdl and...
BuiltinMessages.wsdl found in AggService/wsdl and AggService/com/pe/agg respectively, into AggregationEJB/ejbModule/com/pe/agg. Note that these are the WSDL files from the original FDML project. ProductAggregation.wsdl in AggServiceMigrationV5 is based on the original file.

4. After copying the files, update the import reference of the AggregationEJB in the AggregationBinding.wsdl and ProductAggregation.wsdl files to reference the local copies of the WSDL files.

Creating an Approval port type
The approval process is bound to JMS, and the aggregation process is bound as an EJB. To reflect this, we must add an approval port type to the AggServiceMigrationV5 ProductAggregation.wsdl file and change the associations of the BPEL files accordingly.

1. Open AggServiceMigrationV5/wsdl/ProductAggregation.wsdl and create a port type named Approval.
2. Under the new Approval port type, create an operation named aggregate.
3. From the aggregate operation create and link the input element to AggregateRequest, and create and link the output element to Int.
4. Save the changes to the WSDL file.

Associate BPEL processes to the new port type
After creating the new Approval port type we must refer to it in our BPEL processes.

1. We begin by opening the approval business process in the BPEL editor.
2. Select the approvalInvoker entry under the Partner Links listing.
3. With this item selected, you should be able to view its implementation details under the Implementation tab.
4. From this tab the Partner Link Type field should be filled in with approvalLT.
5. Click Edit next to the Partner Link Type field to display the Edit a Partner Link Type window, as shown in Figure 7-12 on page 254.
6. Make sure to change the Port Type entry to Approval. This allows the partner link for approvalInvoker to refer to the new port type we created.
7. Once you have selected the partner link and partner link type through the window in Figure 7-12, click <-> to set the partner link Process Role Name and Process Port Type. See Figure 7-13 on page 255 for the completed screen.
8. We must reflect the changes we just made to the Input and Output nodes.

9. Click the Input icon in the approval process, and navigate to its Implementation tab.

10. Ensure that the checkbox labeled “Create a new Process instance if one does not already exist” is checked.

11. Also be sure that the Operation field points to the aggregate operation, as shown in Figure 7-14 on page 256.

12. Similarly for the Output node, make sure that the Operation field also points to the aggregate operation.
13. Ensure that the aggregation partner link and the aggregate process still point to the correct port types and operations.

14. From the approval process click the Implementation tab for the aggregation partner link. Although the implementation file is shown as /AggServiceMigrationV5/com/pe/agg/aggregation.bpel, refresh the approval BPEL process by clicking the Browse button and navigating to and selecting the same aggregation.bpel file again. In the Partner Link field be sure aggregationInvoker is displayed.

15. Verify the aggregate activity by opening its Implementation tab and ensuring that aggregation is listed as the Partner Link, Aggregation is listed as the Port Type, and aggregate is indicated as the Operation.

16. Save the BPEL process when you are finished.
7.3.5 The exchange partner link

The exchange service is invoked in our business processes as an EJB-bound Web service. In the current release of the BPEL run-time environment (shipped as part of WebSphere Business Integration Server Foundation V5.1) we experienced problems trying to integrate the exchange service. The exchange was originally written to be invoked over WSIF, while in BPEL all of the Web services calls are performed with JAX-RPC. We tried to perform a conversion between the invocation methods, but this caused some difficulties. Since the Web service invocation was not our primary concern, we chose to invoke the exchange using a different format.

We chose to create a client proxy service to the WSIF-enabled exchange service, and then interface locally with the client proxy from the BPEL process. Since the client code would be local to our processes we also chose to use an EJB binding method.

To implement our alterations, follow these instructions:

1. Select the Exchange-service.wsdl file located in the ExchangeStub2Web project.
2. Right click the Enterprise Services menu and select Generate Deploy Code.
3. Select the EJB binding type, as opposed to SOAP. Keep all the provided default options, as shown in Figure 7-15 on page 258, and click Finish.
4. After this step completes you may notice an error message. It is safe to continue in the code generation by clicking OK.
5. After the generation has completely finished, you will notice errors relating to WSDL file imports. Copy and paste the Exchange-binding.wsdl and Exchange-service.wsdl files from the ExchangeStub2Web project's webApplication/wsdl folder into the webApplication.wsdl package located under the ExchangeStub2WebEJB project.

6. The exchange file imports used for the approval and mass update business processes will also have to be updated to reflect this change. Copy the ExchangeEJBBinding.wsdl and ExchangeEJBService.wsdl files from

7. For both the approvalGen.wsdl and massUpdateGen.wsdl files located in AggServiceMigrationV5/com/pe/agg replace the two import statements 
   ./../webApplication/wsdl/Exchange-binding.wsdl and
   ./../wsdl/Exchange/no_wsgw/Exchange-bind.wsdl with one import to
   ./../wsdl/Exchange/no_wsgw/ExchangeEJBBinding.wsdl, with a namespace of

8. After these updates have been made make sure that the partner links defined in the approval and mass update partners reflect the changes. Open each process in the BPEL editor and check that the Exchange partner link references the appropriate massUpdateGen.wsdl or the approvalGen.wsdl file and that the partner link type fields refer to the Exchange-binding.wsdl file found in the ExchangeStub2WebEJB/webApplication/wsdl directory. After you are certain of this, check the Implementation tab for any exchange activities in the processes, namely createEntry and getDescription. The appropriate operation should be displayed, either createEntry or getDescription.

9. You will also notice one remaining error regarding the Manifest class-path of ExchangeStub2WebEJB. WebSphere Studio Application Developer Integration Edition incorrectly assumed that ExchangeStub2Web should be packaged as a JAR file, when in fact it should be a WAR file.

10. Using the source tab view of ExchangeStub2WebEJB’s MANIFEST.MF file, change the file ending to the correct value.

11. Add the ExchangeStub2Web project as a module inside the ExchangeStub2WebEAR and AggServiceMigrationV5EAR application deployment descriptors.

7.3.6 Integrate AggServiceMigrationV5 with the other projects

Now that AggServiceMigrationV5 is equivalent in functionality to the original AggService project, we remove the original AggService project from our workspace and replace any references to it with the new AggServiceMigrationV5 project.

Before we delete the AddService project there is a section of code that is needed by AggregationServiceMDB that we must save. This Message Driven Bean invokes the approval process and was generated against the FDML version of the process. Instead of regenerating the class we make one small alteration to it. The code would otherwise need to be regenerated because of the fact that we are now invoking a different instance of the approval business process than the one that the class was generated against.
In most cases, you would normally regenerate this Message Driven Bean from the new approval BPEL process we have created.

1. Open the approval.java file found in AggService/com/pe/agg. Select and copy the getCorrelationSetFromAggregateRequest method inside of the class into AggregationServiceMDB.java located under AggregationEJB.

2. Go to line 135 of AggregationServiceMDB.java and remove the com.pe.agg.approval package name in the call to getCorrelationSetFromAggregateRequest so that the invocation refers to the local copy of this function.

3. Now we are ready to delete the AggService and AggServiceEAR projects. Right-click both projects and click Delete. This will leave us with a variety of errors, most dealing with project dependencies and references to AggService.

4. For any projects with an error message indicating Missing required Java project: AggService we must change the required project references to point to the new AggServiceMigrationV5 project:
   a. From the Java view select any of the projects showing this error and click Properties.
   b. Go to the Java Build Path tab and select the Projects tab.
   c. As shown in Figure 7-16 on page 261, the AggService project is indicated as being missing.
   d. Uncheck the AggService project and select the new AggServiceMigrationV5 project.
   e. Click OK.

5. Repeat these steps for any other projects that have the same error message.
6. Add all of the projects we want included in the AggServiceMigrationV5 EAR. These will be all of the projects originally included in the AggService EAR plus one or two additional projects.

7. Open the Application Deployment Descriptor for the AggServiceMigrationV5EAR project, add the following modules to it, and click Save:
   - AggServiceMigrationV5EJB
   - AggServiceMigrationV5Web
   - massUpdateEJB
   - MessageTesterWeb
   - ApprovalWeb
   - AggregationEJB
   - AggSchedulerEJB
   - AggSchedWeb

The Module tab of the application deployment descriptor should look similar to the one shown in Figure 7-17 on page 262.
8. Now fix the IWAE0024W Manifest Class-Path errors related to the AggService.jar. These errors will be found for both massUpdateEJB.jar and AggregationEJB.jar.

9. Double-click the error messages to bring up the JAR Dependency Editor.

10. Unselect the AggService.jar entry, and instead check the AggServiceMigrationV5.jar entry.

   The JAR dependency editor window should look similar to Figure 7-18 on page 263.
7.4 Running BPEL processes in WebSphere Studio V5.1

This section assumes that migration is complete. Your project should be free of any errors. We can now configure WebSphere Studio Application Developer Integration Edition to run our BPEL processes. All of the steps we follow in this section are directly or indirectly related to steps that must be taken at deployment time when the application would be installed on a WebSphere Business Integration Server Foundation server.

7.4.1 Generating BPEL process deploy code

We must create BPEL deployment code for each of our three processes.

1. Right-click each BPEL process file, and select Enterprise Services → Generate Deploy Code.

2. For each process, follow the specific deploy instructions shown in Table 7-1 when entering options on the Generate BPEL Deploy Code window.

<table>
<thead>
<tr>
<th>BPEL process</th>
<th>Options when deploying</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregation</td>
<td>Use all default options. No changes necessary.</td>
</tr>
<tr>
<td>BPEL process</td>
<td>Options when deploying</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>approval</td>
<td>Use all default options except for the referenced exchange partner. For the exchange reference ExchangeEJBService.wsdl found in ExchangeStub2WebEJB/webApplication/wsdl. See Figure 7-19 as an example.</td>
</tr>
<tr>
<td>mass update</td>
<td>Use all default options except for the referenced Exchange partner. For the exchange reference ExchangeEJBService.wsdl found in ExchangeStub2WebEJB/webApplication/wsdl. See Figure 7-19 as an example.</td>
</tr>
</tbody>
</table>

**Note:** Generating the deploy code will create additional AggService projects in your workspace. The AggregationServiceMigrationV5EJB project may be generated with errors due to it missing the XSDBEANS.jar package. Add the XSDBEANS variable to the project’s build path if this is the case.

3. After generation, any warnings regarding EJB links to ejb/compensation/CoordinatorHome, ejb/compensation/RunProcletHome, or ejb/compensation/StandardExecutorHome can be safely ignored. Our processes do not take advantage of compensation functionality provided by Process Choreographer, and thus we do not need the EJBs required for compensation.
7.4.2 Create a test server

To perform this step, refer to 6.2.3, “Create a test server” on page 198, which details creating and configuring a WebSphere Studio Application Developer Integration Edition V5.0 test environment. In this section we are performing our testing in a V5.1 environment, but any differences between the two are minor and the steps performed are largely the same. The steps listed here are brief, and for additional detail you can refer to Chapter 6, “Deploying the existing sample solution” on page 191.

1. Switch to the Server view inside of Studio and select Create a new Server and Server configuration.
   The server type must be V5.1 Integration Test Environment, and for our example we name the server testServer.
2. After this information has been entered click Finish.

7.4.3 Configure the test server

In the following sections we list all of the configuration options that need to be set in the test server for the proper deployment of our applications. As before, see Chapter 6, “Deploying the existing sample solution” on page 191, for more details on each configuration section.

**Add projects to the server configuration**

To add projects to the server configuration:

1. Right-click the server you created.
2. Select Add and remove projects.
3. Add the AggServiceMigrationV5EAR, EIS2_SRVEAR, EIS3_SRVEAR, and MessageTester projects, as shown in Figure 7-20 on page 266.
To configure security:

1. Enable security for the server by using the server configuration editor in WebSphere Studio.

2. Click the **Security** tab, and select **Enable security**. Do *not* enable J2EE security. Be sure the user ID you provide to the WAS run time has enabled for it “act as part of the operating system” as set from the windows local security settings panel.

3. Create an aggclerk user ID on the machine for the approval process Web client to use.

**Configure server classpath**

To configure a server classpath:

1. Change to the Environment tab of the server configuration editor.
2. In the Class Path entry box enter the PEUtil project and 
$WSADIE_HOME/wstools/eclipse/plugins/com.ibm.etools.xsdmodel_5.1.1/jars/xsdbeans.jar to your server's classpath.

Create required JMS objects
To create required JMS objects:
1. Change to the JMS tab of the server configuration editor.
2. Create two queue names, PEAGGIN and PEAGGOUT, in the JMS Server Properties configuration.
3. In the JMS Destinations configuration, create two queue destination entries:
   – PEAGGIN, with a JNDI name of jms/PEAGGIN
   – PEAGGOUT, with a JNDI name of jms/PEAGGOUT
4. Finally, under the EJB tab of the server configuration editor, alter the SenderQListenerPort so that its Destination JNDI Name is jms/PEAGGIN.
5. Save all of your changes to the server configuration.

7.4.4 Starting the test server
One of the last steps in deploying the sample code to your test server involves publishing the latest copy of the code to the test server. Publishing copies the sample code into the run-time environment. To publish the code:
1. Change to the WebSphere Studio Server perspective.
2. You will notice that the Server window probably indicates that the server needs to be republished.
3. Right click the server and select Publish.
4. When the publishing is complete and the server is up to date, the server state will be listed as Server is synchronized.

Restriction: Be aware of a windows path length limitation when publishing. If while using the file copy transfer method to publish, some of the class files in the AggService EJB project fail to copy, the installation path of WebSphere Studio or the project’s name may have to be shortened.

A further step that is normally performed at application install time is to create database tables necessary to store the run-time data for our processes. This is required before the business processes can run. When the test server is still stopped, right-click the test server and select Create tables and datasources.
7.5 Testing the business processes

Once the migrated application is deployed to the test server, you should test that
the application behaves correctly and that its function is the same as the original
application. Section 6.2.7, “Testing the deployed application” on page 210,
describes the main scenarios of the application, and you should test that the
migrated application still functions the same way.

Test that:

- The approval process of the migrated applications works as described in
  “Testing the approval process” on page 210.
- The mass update process of the migrated applications works as described in
  “Testing the mass update process” on page 215.

7.6 Preparing the application for deployment

After we have successfully deployed and tested the scenario and its components
under the WebSphere Studio Application Developer Integration Edition V5.1 test
environment, we need take to export the projects from WebSphere Studio and
make them ready for deployment.

7.6.1 Deployment to a server environment without Cloudscape™

If you will be deploying your business processes to a server environment that
does not include the Cloudscape datastore system, you need to remove the data
maps currently created for the Cloudscape datastore. For example, if you deploy
the processes to a server environment that uses DB2 as its database, as we
describe in Chapter 8, “Deploying sample on WebSphere Business Integration
Server Foundation” on page 271, it is necessary to make the following changes:

1. Switch to the J2EE perspective in WebSphere Studio.
2. Expand the EJB project containing your generated process code; in our case
   this is AggServiceMigrationV5EJB.
3. Expand the Maps section of the EJB project, right-click the Cloudscape
   entry, and select Delete.
4. Delete both the map and the schema.

**Note:** This step is also detailed in the WebSphere Application Server v5.1
Release Notes, found in the WebSphere Application Server v5.1 Information
Center.
7.6.2 Export the process EAR file

To export the process EAR file:

1. To finally transfer the business process resources from WebSphere Studio into a deployable EAR file, export the AggServiceMigrationV5EAR project.
2. Right-click the AggServiceMigrationV5EAR project, and choose Export.
3. Choose EAR file as the type of export and click Next.
4. Complete the export wizard and export the EAR file to a directory of your choice.

For a more detailed description of the export process see 5.7.7, “Exporting code for deployment” on page 184.
Deploying sample on WebSphere Business Integration Server Foundation

In this chapter we discuss how we set up our WebSphere Business Integration Server Foundation V5.1 environment to run our sample private exchange application. Since the actual installs of the base products are fairly routine, the primary focus is on how to configure the Business Process Container and the installation and validation of our application.
8.1 Deployment considerations

Deployment for WebSphere Business Integration Server Foundation V5.1 is little changed from earlier versions of WebSphere Application Server Enterprise. For our purposes, what differences there are stem mainly from the change to BPEL.

**Note:** If you have deployment procedures that modify deployment artifacts you may have to take these changes into account in ways we did not.

Our sample application has no production requirements so we did not consider operational concerns such as load balancing or highly available configurations. Consequently, we did not use WebSphere Application Server Network Deployment, and installed everything on a single server (no network deployment, no cells, no clusters).

8.2 Prerequisite software

To deploy our application we built a server with these products installed:

- DB2 V8.1
- WebSphere MQ V5.3
- WebSphere Business Integration Server Foundation V5.1

The Business Process Container and the resources it needs are not automatically installed or configured when you install WebSphere Business Integration Server Foundation. Once you have your platform installed you need install the Business Process Container, BPE, in a separate step.

- EAR file for our sample application

System House designed our sample application to operate in a production style environment. We installed only the minimum necessary set of products to be able to run our application and let us focus on the migration and configuration issues of moving to WebSphere Business Integration Server Foundation V5.1.

8.3 Installing prerequisite products

We made default installs of the core products with minor exceptions.

- DB2

The directory we used for DB2 8.1 installation was C:\SQLLIB.
8.4 Installing the Business Process Container

The Business Process Container is a pure Java implementation and is a J2EE application. The APIs that clients may call to start business processes and claim and complete work items are in bpecontainer.jar, which is an EJB module. The APIs themselves rely on internal classes, which are part of the WebSphere Business Integration Server Foundation V5.1 product. The synchronous session bean and asynchronous message-driven bean interfaces are also in bpecontainer.jar.

Another JAR file, compensate_ejb.jar, is part of the installation. As its name implies, this file contains the Compensation Coordinator. (We did not use compensation in our sample application.)

Aside from the container itself, you also need:

- A relational database and a DataSource (to hold process state). You can use Cloudscape or choose among DB2, Oracle, Sybase, or DB2 for z/OS. We used DB2.
- A JMS provider and JMS resources. Any compliant JMS provider, including WebSphere embedded messaging or WebSphere MQ, is acceptable. We used WebSphere MQ.

The Business Process Container does not come installed with WebSphere Business Integration Server Foundation V5.1; you have to install it. There are two ways you can do this: You can use the wizard that comes with the WebSphere Administrative Console or use a script and wsadmin. The preferred way is to use the wizard. Naturally, the scripted method would be the obvious
choice for production environments. Either way creates the DataSource and JMS resources you need.

When you use the wizard to do the install you use data definition files (DDL) provided for the supported relational databases. There is also another script to create the JMS resources for WebSphere MQ. (You do not need this if you use embedded messaging.)

8.4.1 Preparing to install the Business Process Container

The Business Process Container requires a minimum environment of 32000 bytes, and you want your server running when you configure it. Therefore the first two steps are:

1. Increase your environment space. Open a DOS prompt window and enter the following command:
   ```
   set ComSpec=%SystemRoot%\system32\cmd.exe /e:32000
   ```

2. Start your application server, for example:
   ```
   %WAS_HOME%\bin\startserver server1
   ```
   Wait for the server to start. Once it does, you will see the “ready for e-business” message that looks like this:
   ```
   ADMU3000I: Server server1 open for e-business: process id is nnnn
   ```

   **Tip:** There are two topics in the WebSphere Business Integration Server Foundation V5.1 Information Center that provide complete references on settings for your Business Process Container and its resources. These are:
   - Using the Install Wizard to configure the Business Process Container
   - Business Process Container Install Wizard settings: WebSphere Business Integration Server Foundation

   These topics cover all the possible settings for any resources, such as databases, you may want to use.

   In the next section, we describe how we installed the Business Process Container for our application.

   **Note:** We installed our server in C:\was. Of course, your install may be to some other location, so substitute your location as necessary. Also, if you are unsure, the default server name is normally server1. You can easily locate any running servers with the `serverstatus` command. Enter it like this:
   ```
   C:\was\bin\serverstatus -all (CR).
   ```
At this point the server should start with no exceptions. Sometimes it is possible for the server to reach the ‘ready for e-business’ state even with exceptions. If there are any, clean these up before you continue.

8.4.2 Installing the Business Process Container using the WebSphere Administrative Console wizard

To install the Business Process Container using the WebSphere Administrative Console proceed as follows:

1. Log onto the WebSphere Administrative Console:
   a. Open a Web browser and enter the following URL (assuming you are at the server; if not, replace localhost with your server’s host name):
      \[
      \text{http://localhost:9090/admin}
      \]
   b. In the Login window, enter <yourID> as the user. Click OK. Any user ID can be used, as it simply identifies your session on the WebSphere Administrative Console. If you can, use the user ID and password you want to use with your message resources. Then these values will be defaulted for you when you configure the Business Process Container.

   **Tip:** You may want to check to see if the Business Process Container is already installed at this point. If it is, you should remove it before you proceed. You cannot modify its settings. Refer to 8.4.6, “Removing the Business Process Container” on page 299.

2. Start the Install Business Process Container wizard.
   a. In the WebSphere Administrative Consoles tree view, expand **Servers**.
   b. Select **Application Servers**.
   c. Select **server1**.
   d. Scroll down and select **Business Process Container**.
   e. Scroll down again and select **Business Process Container Install Wizard**.

3. Install the Business Process Container using the wizard.
   – In Step 1, shown in Figure 8-1 on page 276, select your database provider from the drop-down list and use a user ID that has administrative privileges. We used:
     – JDBC Providers: DB2 UDB 8.1 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA))
- Datasource User Name: sah407r (Note that if you used this name to log on to the WebSphere Administrative Console this name is already filled in. You can override this, and you must override it if you logged on with something that is not a valid DataSource user name.)

- Datasource Password: <password>

Figure 8-1 Installing the Business Process Container wizard, Step 1

- Click Next.

4. In Step 2, shown in Figure 8-2 on page 277, you select the JMS provider and set the parameter values needed to authenticate JMS messages.
   a. For JMS providers we used WebSphere MQ JMS Provider.
   b. In this case the JMS user ID and password are not used, but the security role mapping ID and JMS API parameters are, so we used:
      - Security Role Mapping: <userid>
      - JMS API User ID: <userid>
      - JMS API Password: <password>
c. Click Next.

5. Step 3 creates the JMS resources. We want to use the defaults, so make sure this radio button is selected. We also want to create the Web client, so we left this option checked. Figure 8-3 on page 278 shows an example. Click Next.
6. Step 4 gives you a summary of the Business Process Container configuration that you will install. Inspect this and make sure it is what you intend. Once the container is installed you cannot simply change these values (you have to uninstall and reinstall the container to change its configuration). Our configuration is shown in Figure 8-4 on page 279.
Chapter 8. Deploying sample on WebSphere Business Integration Server Foundation

Figure 8-4 Installing the Business Process Container wizard setting verification

a. If everything is in order, click **Finish**.

At this point you should see a series of logging and status messages. If all goes well, these will include two entries in the WebSphere Administrative Console, as shown in Figure 8-5. If you get these messages, you have successfully installed the Business Process Container and the BPE Web client.

ADMA5013: Application BPEContainer_k8brmp_server1 installed successfully.
ADMA5013: Application BPEWebClient_k8brmp_server1 installed successfully.

If you want to start the application, you must first save changes to the master configuration.

**Save to Master Configuration**

Figure 8-5 Installing the Business Process Container with the wizard and successful completion
b. If your Business Process Container install is successful, click **Save to master configuration**.

**Note:** Your Web client install will fail if it is already installed.

c. Click **Save** again on the work area panel.

7. The next task is to define the path for your DataSource (see Figure 8-6). From the home page of the WebSphere Administrative Console:
   a. Expand **Environment** and click **Manage WebSphere Variables**.
   b. Scroll down and click **DB2UNIVERSAL_JDBC_DRIVER_PATH**.
   c. In the Value field, enter `C:\SQLLIB\java\`, as shown in Figure 8-6.

d. Click **OK**.

e. Click **Save** in the upper frame and then **Save** again to save your configuration changes.

f. Click **Logout** on the top horizontal bar to log out from the WebSphere Administrative Console.

8. Next, create the Business Process Container database and tables. A DDL file is provided, which will create the BPEDB database for you. Modify the DDL file to connect to the database using your user ID (we used sah407r for our user ID).

**Tip:** For production databases, where you need to control and manage the tablespaces, there are additional DDL files to help you create these, the tables and the schema. These are located in `%WAS_HOME%\ProcessChoreographer` and are well documented with comments.
a. In a Windows Explorer window, navigate to the DDL file and make a copy of it before you do anything else. The file is:

   %WAS_HOME%\ProcessChoreographer\createDatabaseDb2.ddl

b. In the DDL file, locate the following line near the beginning of the file:

   CONNECT TO BPEDB;

c. Modify this line to the following, but substitute <userid> for the Windows user account you used and give your password:

   CONNECT TO BPEDB USER <userid> USING <password>;

d. Save and close the DDL file.

9. Run the DDL script you just modified:

   a. Using a command prompt change to the following directory:

      %WAS_HOME%\ProcessChoreographer\

   b. Start a DB2 command window by entering the command:

      db2cmd

c. In the DB2 command window, enter the following command to create the tables:

      db2 -tf createDatabaseDb2.ddl

      

      **Tip:** A quick way to do this is to open a DB2 command window from the Windows Start menu, and issue this command using the fully qualified name of the target DDL file:

      C:\sqllib\db2>db2 -tf
      C:\<WSIF_Root>\ProcessChoreographer\createDatabaseDb2.ddl

   d. Bind the DB2 CLI programs against the new database by issuing the following commands, substituting <userid> for your Windows user account and <password> for your password:

      db2 connect to BPEDB user <userid> using <password>
      db2 bind C:\SQLLIB\bind\@db2cli.lst blocking all grant public
      db2 connect reset
      exit

10. Create a WebSphere MQ queue manager and resources for the Business Process Container using the `createQueues.bat` command.

   a. Change the directory to:

      %WAS_HOME%\ProcessChoreographer
b. At the command line, run the `createQueues` command, giving the name of your queue manager. We used ka6brmp because KA6BRMP was our node name:

```
>createQueues.bat WAS_BPE_ka6brmp.
```

c. Set your queue manager to start automatically (the default is to start manually). Use the WebSphere MQ Services tool and the General tab as shown in Figure 8-7:

i. Select **Start → Programs → WebSphere MQ → WebSphere MQ Services**.

ii. Select the **General** tab, and in the Startup drop-down list select **Automatic**.

iii. Click **OK**. Close the WebSphere MQ Services tool.

### 8.4.3 Business Process Container commands

Several commands related to installation and configuration of the Business Process Container are supplied with WebSphere Business Integration Server
Foundation V5.1. These are .jacl files, and you run them as JACL commands, usually with the wsadmin command utility.

Among these commands are three we found useful for Business Process Container configuration management. You use these to:

- Configure the Business Process Container (bpeconfig.jacl).
- Remove configuration settings (bpeunconfig.jacl).
- Check on or verify the settings in effect (bpecheck.jacl).

**Business Process Container configuration with bpeconfig**

You can create and configure the Business Process Container entirely from the command line. You run the bpeconfig.jacl script in wsadmin. This accomplishes everything, including:

- Installs the Business Process Container application (EAR)
- Configures the Business Process Container Service
- Creates the DataSource definition
- Creates the database
- Configures the J2C Authentication Alias for Business Process Container and messaging

You have to respond to prompts to use bpeconfig.jacl, so before you run it have available the information you will need:

- Node name where Business Process Container will install
- Server name
- Virtual host name
- User ID and password for database and Business Process administrator
- Database you will use (Cloudscape, DB2, Oracle, Sybase or DB2 zOS)
- JMS provider (embedded messaging or WebSphere MQ)

The `bpeconfig` command can be run repeatedly until everything works; it is not an all or nothing process, so you can run it, fix what did not work, and run it again. Once a task is done you can say No when prompted if you want to do it again. You can proceed by elimination, but it helps to set as many of your parameters as you can beforehand. The Business Process Container settings we used are shown in Table 8-1 on page 284. We show our security parameters here, but you do not want to code and save yours in plain text.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>servername</td>
<td>server1</td>
<td>Application server where Process Choreographer will be installed</td>
</tr>
<tr>
<td>hostname</td>
<td>default_host</td>
<td>Virtual host for Web client</td>
</tr>
<tr>
<td>enableSecurity</td>
<td>Yes</td>
<td>Enables global security</td>
</tr>
<tr>
<td>serverUser</td>
<td>sah407r</td>
<td>Security user name</td>
</tr>
<tr>
<td>serverPwd</td>
<td>itso4u</td>
<td>Password for global security user ID</td>
</tr>
<tr>
<td>createDB</td>
<td>Yes</td>
<td>Whether to create the Process Choreographer database</td>
</tr>
<tr>
<td>dbType</td>
<td>&quot;DB2&quot;</td>
<td>Type of database</td>
</tr>
<tr>
<td>dbHome</td>
<td>C:sqllib</td>
<td>Home directory of database; $(dbHome)java must exist</td>
</tr>
<tr>
<td>dbVersion</td>
<td></td>
<td>Not used with DB2</td>
</tr>
<tr>
<td>dbUser</td>
<td>sah407r</td>
<td>User ID for database</td>
</tr>
<tr>
<td>dbPwd</td>
<td>itso4u</td>
<td>Password for database user ID</td>
</tr>
<tr>
<td>dbAdmin</td>
<td>sah407r</td>
<td>User ID of database administrator</td>
</tr>
<tr>
<td>dbServerName</td>
<td></td>
<td>Sybase only</td>
</tr>
<tr>
<td>dbServerPort</td>
<td></td>
<td>Not used with DB2</td>
</tr>
<tr>
<td>dbTablespaceDir</td>
<td></td>
<td>Not used</td>
</tr>
<tr>
<td>dbInstance</td>
<td></td>
<td>Informix® only</td>
</tr>
<tr>
<td>dbName</td>
<td>BPEDB</td>
<td>Database name</td>
</tr>
<tr>
<td>dbSubSystem</td>
<td></td>
<td>z/OS only</td>
</tr>
<tr>
<td>dbStorageGroup</td>
<td></td>
<td>z/OS only</td>
</tr>
<tr>
<td>dbJava</td>
<td></td>
<td>iSeries only</td>
</tr>
<tr>
<td>createQM</td>
<td>yes</td>
<td>Whether to create the BPE Queue Manager or not</td>
</tr>
</tbody>
</table>
Table 8-1 on page 284 shows the values we used. For the default values see the documentation in the `bpeconfig.jacl` command.

The `bpeconfig.jacl` script did not use the entered name for the queue manager, but it used the default queue manager name of the form `WAS_<nodename>_<servername>`. A rerun explicitly to change this parameter had no effect. Consequently, our listing of the command output in Example 8-1 on page 287 shows the default name.

You can correct the queue manager name in the WebSphere Administrative Console where it is part of the definition of the queue connection factory. We did this using server scope, not node scope, since no resources are defined at node scope in our environment. After you enter the correct name remember to save your change and restart the server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mqType</td>
<td>MQSeries</td>
<td>Type of MQ provider, “Embedded” or “MQSeries”</td>
</tr>
<tr>
<td>mqHome</td>
<td>C:\MQ</td>
<td>MQ installation home directory</td>
</tr>
<tr>
<td>mqUser</td>
<td>sah407r</td>
<td>WebSphere MQ JMS user ID (not used with embedded messaging)</td>
</tr>
<tr>
<td>mqPwd</td>
<td>itso4u</td>
<td>MQ User password</td>
</tr>
<tr>
<td>qmNameGet</td>
<td>WAS_BPE_ka6brmp</td>
<td>WebSphere Queue Manager name for ‘Get’</td>
</tr>
<tr>
<td>qmNamePut</td>
<td>***</td>
<td>MQ Clusters only</td>
</tr>
<tr>
<td>mqClusterName</td>
<td>***</td>
<td>MQ Clusters only</td>
</tr>
<tr>
<td>adminUsers</td>
<td>***</td>
<td>Users to be assigned ‘BPESystemAdministrator’ role</td>
</tr>
<tr>
<td>adminGroups</td>
<td>***</td>
<td>Groups to be assigned ‘BPESystemAdministrator’ role</td>
</tr>
<tr>
<td>jmsRunAsUser</td>
<td></td>
<td>Run-as user for JMS API</td>
</tr>
<tr>
<td>jmsRunAsPwd</td>
<td>***</td>
<td>Password for run-as user for JMS API</td>
</tr>
</tbody>
</table>
To run the bpeconfig script:

1. Change to:
   ```
   %WAS_HOME%\ProcessChoreographer\sample
   ```

2. Enter:
   ```
   %WAS_HOME%\bin\wsadmin -f bpeconfig.jacl
   ```

   **Note:** The `bpeconfig` command we show here assumes server1 is running, but it does not have to be. You can run “headless” (that is, with server1 not running), in which case the `wsadmin` format is:
   ```
   >wsadmin -conntype NONE -f bpeconfig.jacl
   ```

In practice, doing your configuration this way is a more reliable and repeatable process, especially if you refine your own version of the command to reflect your installation. However, getting it working properly does take some interaction since the command contains a number of switches you have to get used to in order to complete the process. You do not get to override all the prompts (unless you want to rewrite the script). If you are comfortable with the problem solving needed to get `wsadmin` commands working, we highly recommend this approach. Otherwise, stick with the WebSphere Administrative Console procedure to create the Business Process Container.

Some problems we had were:

- At one point an exception was occurring when the script attempted to create our DB2 database and do the binds. The command involved was visible in the command window, however, so we just ran it itself. This worked and the next time we ran `bpeconfig.jacl` we just replied No when prompted to create the DB2 database.

- At first, the Web client was installed but could not start, as localhost was not defined. The solution for this was to rerun the command and respond Yes when prompted to install `processportal.ear` (the first time we said No). We then used `Default_host` when prompted and it started just fine. Although it is not self-evident, `processportal.ear` is the Web client application.

- The script for `bpeconfig.jacl` does not seem to be able to locate the install directory for WebSphere MQ correctly. Fix this if you do the install WebSphere MQ resources section by simply giving your install root explicitly (that is, even if it looks OK, do not use the default value).

Our results from several iterations through `bpeconfig.jacl` are shown in Example 8-1 on page 287.
Example 8-1  Results from bpeconfig command

C:\was\ProcessChoreographer\sample>\was\bin\wsadmin -conntype NONE -f bpeconfig.jacl

WASX7357I: By request, this scripting client is not connected to any server process. Certain configuration and application operations will be available in local mode.

*****************************************************************************
* This script allows to install bpecontainer.ear, install the Web client,       *
* create the Process Choreographer DB, create the WebSphere MQ queues, and   *
* update WCCM with the DataSource, ListenerPorts, and Scheduler required by  *
* Process Choreographer.                                                     *
* Supported databases are Cloudscape, DB2, Informix, Sybase, Oracle, and     *
* SQL Server; supported JMS providers are WebSphere Embedded Messaging and   *
* WebSphere MQ.                                                              *
* The prerequisite software must already be installed.                       *
* -------------------------------------------------------------------------- *
* You will be prompted for the required information at each step. The default value is always listed first in a prompt, you can select it by simply pressing the 'Enter' key. *
* -------------------------------------------------------------------------- *
* DISCLAIMER: This sample script is provided AS-IS for your convenience. It can only create simple configurations. If you need to create more complex configurations, please follow the instructions provided in the InfoCenter. *
*****************************************************************************

Configuring: Node 'ka6brmp' / Server 'server1'
bpecontainer.ear already installed, skipping
processportal.ear already installed, update [No/yes]? No
  => No
Create the Process Choreographer database [Yes/no]? no
  => no
Create the DataSource for the Process Choreographer database [Yes/no]? no
  => no
Use Embedded Messaging or WebSphere MQ [Embedded/MQSeries]? MQSeries

Note: If you are not working with WebSphere Business Integration Server Foundation you can still work out the parameters for your bpeconfig.jacl command because it is packaged with WebSphere Studio Application Developer Integration Edition. It is in:

%WAS_HOME%\v5.1\runtimes\ee_v5\ProcessChoreographer\sample
MQSeries
Create the Process Choreographer Queue Manager and Queues [Yes/no]? Yes

Queue Manager name [WAS_ka6brmp_server1]:
WAS_ka6brmp_server1

Will the Queue Manager join a WebSphere MQ cluster [No/yes]? No

"d:\programs\IBM\WebSphere MQ\bin" does not exist, enter the WebSphere MQ installation directory [d:\programs\IBM\WebSphere MQ]: C:\MQ

WebSphere MQ queue manager created.

Creating or replacing default objects for WAS_ka6brmp_server1.

Default objects statistics : 31 created. 0 replaced. 0 failed.

Completing setup.

Setup completed.

WebSphere MQ queue manager 'WAS_ka6brmp_server1' started.

Create the ListenerPorts [Yes/no]? Yes

Create Queue Provider and MQ Queue name for Process Choreographer
Create settings for Server server1(cells/ka6brmp/nodes/ka6brmp/servers/server1:server.xml#Server_1)
WebSphere MQ JMS Provider already exists
MQ_INSTALL_ROOT = C:\mq
Create BPECF Queue Connection Factory:
BPECF(cells/ka6brmp/nodes/ka6brmp/servers/server1:resources.xml#MQQueueConnectionFactory_1084297281226)
Create BPECFC Queue Connection Factory:
BPECFC(cells/ka6brmp/nodes/ka6brmp/servers/server1:resources.xml#MQQueueConnectionFactory_1084297283199)
Create BPEIntQueue
Create BPEInternalListenerPort
Create StateManageable: initialState = START
Create BPEApiQueue
Create BPEApiListenerPort
Create StateManageable: initialState = START
Create BPEHldQueue
Create BPEHoldListenerPort
Create StateManageable: initialState = START
Create BPERetQueue
Save Admin changes

Queue and listener configuration
Server: server1
Node: ka6brmp
Path to MQ: C:/MQ
Queue Manager for 'GET': WAS_ka6brmp_server1
Queue Manager for 'PUT': WAS_ka6brmp_server1
IntQueueName: BPEIntQueue
ApiQueueName: BPEApiQueue
HldQueueName: BPEHldQueue
RetQueueName: BPERetQueue

Create the Scheduler for Process Choreographer [Yes/no]? Yes
===> Yes
Locating WorkManagerProvider
Create BPESchedulerWorkManager in
WorkManagerProvider(cells/ka6brmp/nodes/ka6brmp/servers/server1:resources-pme.xml#WorkManagerProvider_1)
Create BPEScheduler

Scheduler configuration
Scheduler name: BPEScheduler
JNDI name: BPEScheduler
Datasource JNDI name: jdbc/BPEDB
Table prefix: SCHED_
Poll interval: 10
Security Role: S:Everyone
Work Manager: BPESchedulerWorkManager

**************************************************************************
* WARNING: Global security is turned off, staff-related functions like the
* Web client will not work properly.
**************************************************************************

Enable global security using the Local OS user registry [Yes/no]? Yes
===> Yes
Server user ID [sah407r]:
===> sah407r
sah407r's password []: itso4u
===> itso4u
Global Security is now enabled.
Set 'com.ibm.SOAP.loginUserid' in soap.client.props [Yes/no]? Yes
===> Yes
Delete the temporary directory C:\tmp\bpeconfig [Yes/no]? Yes
===> Yes
Process Choreographer configuration finished. See bpeconfig.log for details.
C:\was\ProcessChoreographer\sample>
After you have run `bpeconfig.jacl` successfully, both the Web client and Business Process Container applications will be installed and started. Verify this from the WebSphere Administrative Console in the normal way, or else use the `bpecheck.jacl` command.

### 8.4.4 Verifying the Business Process Container configuration

Your Business Process Container is now installed. By now you are also aware that besides the Business Process Container application EAR itself, Business Process Choreographer relies on several other system components. These are:

- An instance of WebSphere Business Integration Server Foundation
- Listener ports
- Message beans
- JDBC providers
- DB2 (or some other database product)
- JMS resources (queue connection factories and queues)
- The Message Listener Service
- Scheduler (usually BPEScheduler unless you use something else)

You need to take the time to verify your installation and configuration of each of these components. The main reason is this: To make changes to your Business Process Container configuration after you run the wizard you have to uninstall the container and then reinstall it. You want to do this before you have data in your application if you have to do it at all, because once you have running processes you have to consider the impact on them if you remove the Business Process Container. At this point it is relatively easy to do the configuration over again. Once you have process data in the BPEDB, to make configuration changes you must remove your process templates from the database. Before you can remove the templates you have to manually stop any long-running processes you may have in flight. In short, the sooner you verify the configuration the better.

There are two main ways to proceed: You can use the WebSphere Administrative Console and proceed by inspection, or you can use the `bpecheck.jacl` command. What you see actually varies a bit so we describe both methods.
Verifying using WebSphere Administrative Console

Here are the steps to follow to verify your Business Process Container configuration:

1. Restart the server. Using a command prompt, navigate to where your server is installed, for example, C:\WebSphere\Appserver\bin, and issue the following commands:
   
   - stopserver server1
   - startserver server1

   Wait until your server restarts and gives you the “ready for e-business message”. Check that no exceptions were thrown and there are no qualifications on the ‘ready’ line.

   **Tip:** Servers can start and be ‘ready for e-business’ even though resources necessary to run Process Choreographer did not start successfully. If you are not absolutely certain you have a clean start, the time you take to verify this will be more than repaid later because it will save you the effort to determine at run time why your processes are failing. The best way to verify that your start is clean is to inspect your System.out log. We found it useful to delete this file before recycling our server in order to isolate the results of each cycle. Your System.out file can be found at:

   
   C:\<install root>\logs\server1\System.out.

   Only look for exceptions. If you are unclear as to their cause, also inspect the binds in the log. All of these must succeed for a clean start. It appears that WebSphere Business Integration Server Foundation V5.1 starts also produce a number of fix-ups to resource connectors for demo applications. You can ignore these.

2. Start the WebSphere Administrative Console.
   
   a. Open a Web browser and point it to:

      - http://localhost:9090/admin

      The login prompt will appear (see Figure 8-8 on page 292) and it should be the insecure challenge prompt that requires only your user ID. Enter any ID and click **OK**.
3. View the installed applications.
   a. On the left-hand side of the WebSphere Administrative Console, expand Applications and select Enterprise Applications.
   b. From the list of installed applications, click BPEContainer_<NodeName>_server1. Click the link and do not check the check box. The BPEContainer application is the Business Process Container that executes the business processes.
   c. Scroll down and click Map security roles to users/groups (Figure 8-9). On the Mapping Users to Roles page, you will see information about the BPESystemAdministrator, JMSAPIUser and WebClientUser. Click the BPEContainer_<NodeName>_server1 link at the top of the page.

![Figure 8-8 WebSphere Administrative Console logon](image)

**Note:** Once you have security enabled on your server, your WebSphere Administrative Console logon will also prompt for your password. Security must be enabled before you can run our private exchange application. How to do this is described as part of the install procedure. See “Installing ExchangeStub2EAR” on page 309.

![Figure 8-9 Verify security roles BPE_Wizard_Step_9](image)
d. Scroll down and click **Provide Listener Ports for Messaging Beans**. On the Binding Enterprise Beans to Listener Port Names page, shown in Figure 8-10, you will see information about Message Driven Beans and Listener Ports relative to the BPEContainer applications. Click the **BPEContainer_<NodeName>_server1** link at the top of the page.

![Listener Ports for Messaging Beans](image)

**Figure 8-10 Listener Ports for messaging beans**

4. Look at the JDBC resources. These are used to persist the state of long-running business processes when execution is halted.

   a. Expand **Resources** and click **JDBC Providers**.

   b. In the Scope section, select **Server** and click **Apply**.

   c. Once the screen has refreshed, click the **DB2 Legacy CLI-based Type 2 JDBC Driver (XA)** JDBC provider.

   d. Scroll down and click **Data Sources**. You should see the DataSource named BPEDataSourceDb2. Click **BPEDataSourceDb2** to see the details of the DataSource. Notice the JNDI name of jdbc/BPEDB. See Figure 8-11 on page 294.
5. Examine the JMS resources that were created when you ran the bpeconfig.jacl script.
   a. Expand Resources and click WebSphere MQ JMS Provider.
   b. In the Scope section, select Server and click Apply.
   c. Scroll down and click WebSphere Queue Destinations. You should see several queue destinations defined. All of them are named BPE*Queue (where * is a wildcard). As shown in Figure 8-12, these queue destinations have been defined by the Business Process Container JACL script.
d. Click the browser back arrow to go back one page. Click **WebSphere Queue Connection Factories**. You will see connection factories BPECF and BPECFC as they appear in Figure 8-13.

![WebSphere MQ JMS Provider:]

**WebSphere MQ Queue Connection Factories**

A queue connection factory is used to create connections to the associated JMS provider or JMS queue destinations, for pointing WebSphere MQ Queue Connection Factory administrative objects to manage queue connection factories for the WebSphere MQ J.

<table>
<thead>
<tr>
<th>Name</th>
<th>JNDI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPECF</td>
<td>jms/BPECF</td>
<td>Queue Connection Factory for BPE Container</td>
</tr>
<tr>
<td>BPECFC</td>
<td>jms/BPECFC</td>
<td>Queue Connection Factory for BPE Container</td>
</tr>
</tbody>
</table>

**Figure 8-13 Verify JMS Connection Factories**

6. Verify the defined Listener Ports (see Figure 8-14 on page 296).
   a. Expand **Servers** and click **Application Servers**.
   b. In the right-hand pane, click **server1**.
   c. Scroll down and click **Message Listener Service**.
   d. Click **Listener Ports**. You should see three listener ports defined as BPEApiListenerPort, BPEHoldListenerPort, and BPEInternalListenerPort. By clicking each one of them, you can see the Connection factory and Queue destination JNDI names.
7. Inspect the Scheduler configuration (Figure 8-15).
   a. From the Navigation pane, expand Resources and click Work Manager.
   b. In the Scope section, select Server and click Apply.
   c. You will see the BPESchedulerWorkManager workmanager.

Figure 8-14 Verify Internal Listener Port

Figure 8-15 Verify BPESchedulerWorkManager
8. Finally, logout from the WebSphere Administrative Console by clicking **logout** on the top horizontal bar.

**Verifying your Business Process Container with bpecheck**

When you verify your configuration with the bpecheck.jacl command the results you see vary somewhat by how you run the command and the environment you are using, but the command is able to check the integrity of your settings in ways the WebSphere Administrative Console does not. For example, it checks that there are no invalid (multiply defined) JNDI names, as shown in Example 8-2.

**Example 8-2  Output of headless bpecheck command**

C:\was\ProcessChoreographer>\was\bin\wsadmin -conntype NONE -f bpecheck.jacl
WASX7357I: By request, this scripting client is not connected to any server process. Certain configuration and application operations will be available in local mode.

Installed Product
---------------------------------------------------------------------------

-  
Name  IBM WebSphere Application Server  
Version  5.1.0.2  
ID  BASE  
Build Level  
Build Date  

Installed Product
---------------------------------------------------------------------------

-  
Name  IBM WebSphere SDK  
Version  1.4.1.1  
ID  JDK  
Build Level  
Build Date  

Installed Product
---------------------------------------------------------------------------

-  
Name  IBM WebSphere Business Integration Server Foundation  
Version  5.1.0  
ID  PME  
Build Level  a0411.05  

Chapter 8. Deploying sample on WebSphere Business Integration Server Foundation  297
Build Date 3/17/04

Process Choreographer-Version: 5.1.0

Logging Global Security attributes
  Global Security enabled
  Active User Registry is LocalOS
Logging Global Security attributes finished

authDataAlias-Check
authDataAlias-Check finished

BPE unique Deployment and DataSource Check
BPE unique Deployment and DataSource Check finished

Oracle oci8 check
Skipping Oracle oci8 check, because no Oracle database is used by BPEContainer.

tmp path is set to C:\DOCUME~1\sah407r\LOCALS~1\Temp\
tmp pathlength is 34 -> OK

JNDI-Check
  Check ka6brmp
    Checking server server1:
      Only unique JNDI-Names found. OK
      No datasources in node ka6brmp, skipping check
    Finished check ka6brmp
    No datasources in cell ka6brmp, skipping check
JNDI-Check finished

ClassLoader-Check
  Found installation of BPEWebClient
  BPEWebClient deployed on node ka6brmp, server server1
  Application classloader policy set to "Multiple". OK
ClassLoader-Check finished

Variable-Check for provider classpaths
  Variable CLOUDSCAPE_JDBC_DRIVER_PATH in classpath of Cloudscape JDBC Provider (XA) is set
    File exists: C:/was/cloudscape/lib/db2j.jar
  Variable CLOUDSCAPE_JDBC_DRIVER_PATH in classpath of Cloudscape JDBC Provider is set
    File exists: C:/was/cloudscape/lib/db2j.jar
  Variable DB2_JDBC_DRIVER_PATH in classpath of DB2 Legacy CLI-based Type 2 JDBC Driver (XA) is set
    File exists: C:/SQLLIB/java/db2java.zip
Variable-Check for provider classpaths finished
No Network Deployment, skipping Deployment Manager classpath-entries check

Logging attributes
Finished logging attributes

Finished Configuration Check successfully
C:\was\ProcessChoreographer>

8.4.5 Backing up your server configuration files

Once you are satisfied with your configuration, we suggest that you create a backup of the files. To do this:

1. From a DOS command prompt, change to the following directory.
   C:\<install root>\bin
2. Stop the server:
   stopserver server1
3. Back up your configuration:
   backupConfig.bat backupBPEConfig.zip

See Example 8-3 for an example of the output from a backupConfig command.

Example 8-3   Results of the backupConfig command

C:\was\bin>backupConfig.bat BPEConfig6May.zip
ADMU0116I: Tool information is being logged in file
  C:\was\logs\backupConfig.log

-------------------------------------------------------------------------------
ADMU5002I: 358 files successfully backed up

Note: The backupConfig.bat puts your entire configuration into the zip file it creates. This produced a 32-MB file in our simple environment (because our two largest EAR files used up more than 21 MB).

8.4.6 Removing the Business Process Container

To remove the Business Process Container, which you need to do in order to reconfigure it, do the following:

1. In the WebSphere Administrative Console, stop and delete the Business Process Container application.
   a. Start the WebSphere Administrative Console.
b. Click **Enterprise Applications**.

c. Click **Installed Applications**.

d. Check the checkbox for Business Process Container.

e. Optionally, check WebClient as well.

f. Click **Stop**.

g. After the application stops, click **Delete**.

2. Remove the DB2 tables.

3. Remove the WebSphere MQ JMS resources.

   a. Start the WebSphere MQ Explorer using **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**.

   b. Select **Queue Managers**.

   c. Stop and delete the queue manager you created earlier. For this publication, our queue manager was WAS_BPE_KA6BRMP. (If you used the default name it will be of the form WAS_<NodeName>_<servername>. However, if you did this you may have to modify one of the application EAR files that contains a hard-coded reference to the queue manager name. This is discussed in 6.1.2, “Overview of code” on page 192.)

   d. Go to the WebSphere Administrative Console and delete the WebSphere MQ JMS resources you defined for BPE.

### 8.5 Configuration for the sample application

There are two ways you can install an application into the Business Process Container: You can use the application EAR file if you have one, and this is the usual approach if you use WebSphere Studio Application Developer Integration Edition; or you might only have the BPEL and WSDL files of an application, and in this case you can use the Standalone Deployment Tool. Either way, you end up with a J2EE application with its business process installed in the Business Process Container. Also, the process template is inserted into the BPE DB by the installation process.
8.5.1 Resources used by the private exchange application

At this point our Business Process Container is installed and configured. (If the Business Process Container is not installed and properly configured you must do that before you continue.) Now you follow the general procedure for installing BPEL-based process applications found in the WebSphere Business Integration Server Foundation InfoCenter. Generally the steps are these:

1. Create your queue managers and JMS resources.
2. Define the Message Listener ports.
3. Install the EAR file.
4. Enable security (if needed, and if you have interruptible processes in your application, you do need security enabled).
5. Start the application.

Create the WebSphere MQ resources

Our sample application needs an input and an output queue. Use the WebSphere MQ Explorer to create the required queues (you also have to configure them for WebSphere, but you do that separately).

1. Start the WebSphere MQ Explorer and select the queue manager you are using (ours was WAS_BPE_ka6brmp).
   a. Create the inbound queue. Only the name, PEAGGIN, is needed. See Figure 8-16 on page 302.
a. Click **OK**.

b. Create the outbound queue, **PEAGGOUT**. Once again you only need to enter the queue name; see Figure 8-17 on page 303.
2. Create the JMS resources. Start the WebSphere Administrative Console and log on.
   a. Click Resources → WebSphere MQ JMS Provider.
      i. The queue connection factory is QCF.
      ii. JNDI name is jms/QCF.
      iii. Queue manager name is WAS_BPE_KA6BRMP.

3. Create the queue destinations:
   a. Click Resources → WebSphere MQ JMS Provider.
   b. Click WebSphere MQ Queue Destinations.
   c. Click New.
   d. Enter the required fields as:
      i. Name: PEAGGIN.
ii. JNDI Name: jms/peaggin.
iii. Base Queue Name: PEAGGIN.
iv. Click **OK**.

e. Click **New** again and enter:
   i. Name: PEAGGOUT for the output queue.
   ii. JNDI Name: jms/peaggout.
   iii. Base Queue Name: PEAGGOUT.
   iv. Click **OK**.

4. Define the Listener Ports (shown in Figure 8-18).
   a. Click **Message Listener Service**.
   b. Click **Listener Ports**.
   c. Create a new listener port called SenderQListenerPort.

   **Figure 8-18  Listener port definition**

d. Enter connection factory JNDI name jms/QCF.

e. Enter destination JNDI name jms/PEAGGIN.

f. Click **OK**.

g. Click **Save** and **Save** again. Stay in the WebSphere Administrative Console.
8.5.2 Set authentication alias for the JDBC DataSource

Security is enabled for the server, so we need to provide aliases for component and container-managed resources.

Here is a procedure to follow:

1. Start the WebSphere Administrative Console.
2. Expand Resources and click **JDBC Providers**.
3. On the JDBC Providers page, click **DB2 Legacy CLI Based Type 2 JDBC Driver(XA)**.
4. At the bottom of the DB2 Legacy CLI Based Type 2 JDBC Driver(XA) page, click **Data Sources**.
5. On the Data Sources page, click **BPEDataSourceDb2**.
6. On the BPEDataSourceDb2 page, locate the Component managed Authentication alias drop-down list, and select **BPEAuthDataAliasDb2_ka6brmp_server1**.
7. On the same page and next below Component-managed Authentication alias there is another drop-down list, called Container-managed Authentication alias. Go to this list and select **BPEAuthDataAliasDb2_ka6brmp_server1**.

At this point your alias settings should be as in Figure 8-19.

![Component and container-managed alias settings](image)

Figure 8-19 Component and container-managed alias settings.

8. We want to remain in the page, so at the bottom, click **Apply**.
9. Now verify that the database connection is valid and works. At the top of the page locate the test connection button and click **Test Connection**.

   This should result in a confirmation that the database connection is valid as in Figure 8-20 on page 306.

10. Finally, at the top of the page click **Save** and then again click **Save**.
At this point your server should start and be Ready for e-Business. You should verify that this is the case before you install the application EAR files. To do this stop and restart the server at the command line:

> %WAS_HOME%\bin\stopserver server1.
> %WAS_HOME%\bin\startserver server1

No matter what results you get in your command window, inspect the SystemOut.log to be sure there are no exceptions between your startserver command initiation and the final ready for e-Business message. The log is located at:

%WAS_HOME%\logs\server1\SystemOut.log.

### 8.6 Install the private exchange application

Our sample application consists of four EAR files. We only have one process map to think about and it is in ExchangeStubEAR.ear. Since the install procedure for the EARs without .far files in them is simpler we do these first.

We install:

- EIS1EAR.ear
- EIS2_SRVREAR.ear
- EIS3_SRVREAR.ear
- ExchangeStub2EAR.ear
8.6.1 Installing the application EAR files

In this section we discuss installing the application EAR files.

Setting the JVM classpath for our .jar files
Two JAR files are needed to run our sample application, and these need to be located by the JVM when the server starts. The JAR files are PEUtil.jar from our sample application and the xsdbbeans.jar that comes with WebSphere Business Integration Server Foundation. Add these JARs to the JVM classpath:

1. Start the WebSphere Administrative Console.
2. Click **Servers → Application Servers → Server1**.
3. In the Additional Properties list scroll down and click **Java Virtual Machine**.
4. In the Configuration General Properties classpath field, enter:
   
   `C:\was\eclipse\plugins\com.ibm.etools.xsdmodel_5.1.1\jars\xsdbbeans.jar
   R:\sah407r\jason\deploy\Peutil.jar`

   Notice that you can put each entry on a separate line since no semi colon is required. The location where you have PeUtil.jar will depend on your choice of working directory for this exercise.
5. Click **OK**.
6. Save the changes by clicking **Save** and **Save** again.

Installing EIS1EAR, EIS2_SRVREAR and EIS3_SRVREAR
Three of our EAR files are very simple to install because they do not require security to be enabled and none of them contain any .far (process) components. To install each of these, proceed as follows:

1. Start the WebSphere Administrative Console.
2. Click **Enterprise Applications** (Figure 8-21).

   ![Figure 8-21 Install the EAR file](image)

3. From here on you can default the steps of the process since we are not going to change anything. Click **Step 6, Summary**. This should look like Figure 8-22 on page 308.
4. Click **Finish**.

5. Make sure you get the message:

   Application `<EAR file name>` installed successfully.

6. Click **Save to Master Configuration**, then click **Save** again.

7. At this point you can return to Enterprise Applications and you should see your EAR file installed but not running. Start it if you want, or go on to install the remaining EARs.

   **Note:** You will get an Application Security Warning for the `ExchangeStub2EAR.ear` file. You can ignore this.

When you are finished installing the EAR files you can check that they are all present in the WebSphere Administrative Console. They should be listed as in Figure 8-23 on page 309.
8. When they are all installed start the EAR files either by checking them and clicking **Start**, or by stopping and restarting the application server.

### Installing ExchangeStub2EAR

This is the EAR file that contains our business process (in the .far file). The install process is different than for EARs that do not have .far files. (For more information on this step consult the WebSphere Business Integration Server Foundation V5.1 Information Center.) Start out as usual, in the WebSphere Administrative Console. Then:

1. Click **Applications** → **Install New Applications** in the navigation pane.
2. Click **Browse**, locate ExchangeStub2EAR, and select it.
3. Click **Next**.
4. In the Provide options to perform EJB deploy step, make sure you select **DB2**.

**Important:** The database system must be the same as the one that is used for the DataSources for the Business Process Container. A mismatch here will cause your install to fail.
5. (This step is optional.) An application that has an interruptible process, such as ours does, has the option to automatically create database tables for the process. (In our case, these were normally created when the application was built in WebSphere Studio Application Developer Integration Edition. To be sure we regenerated them here.) This option is only available if an application contains an interruptible process and you have the authority to create new tables.

   a. To take this optional route, click **Automatically create tables for Process Entity Beans**.

   The EAR file contains the required relational database (RDB) mapping for process entity beans for the chosen database system. We generated these mappings in the J2EE perspective of the EJB module (Generate → EJB to RDB mappings).

   b. Go to the Summary step and click **Finish**.

   At this point the process template is put into the start state, but there are no template instances because these get created when you start the application. If you do not create a database table for the process entity bean in this step, the creation of the process instance will fail since it is interruptible.

6. If the database tables have not been created automatically in the last step, you have to export the DDL file and apply it to the process choreographer database.

   A table.dll file is generated for each module. This file is located in the back-end directory for DB2.

   In our case we only have one database to apply the table.dll file to, so:

   a. Click **Applications → Enterprise Applications** in the WebSphere Administrative Console navigation pane.

   b. Select the check box next to the application for which you want to export the DDL file and click **ExportDDL**.

   c. Click the link to the data definition language (DDL) file and save it to a temporary directory.
d. Apply the DDL file to the BPEDB database.

Create the user for the sample application
At runtime the Business Process Choreographer work item manager fulfills claims by users for work items. To succeed, a request for a work item must come from a valid potential owner, who is a user ID with claiming rights for that class of work item. Our sample application defines a single user as the sole potential owner for our work items. Before you can test the application this user ID must be created.

Note: If you followed the steps given in Chapter 6, “Deploying the existing sample solution” on page 191, and created the user name aggclerk and gave it administrative rights, there is no need to do this again.

Create a user named aggclerk as a member of the administrators group. Assign a password to the user.

Staff support deployment
Understanding how to define roles and support work item claiming behavior is important if you plan to implement a production application with Business Process Choreographer. In our redbook we discuss only a single user, but this does not cover the implications of a production environment with multiple roles and users. For production systems you should understand:

- How to define multiple roles and users
- The role of the Work Item Manager
- How to work with user registries such as Active Directory or Lightweight Directory Access Protocol (LDAP)

Further details of these topics are covered in the WebSphere Business Integration Server Foundation V5.1 InfoCenter, and in the redbook WebSphere Business Integration Server Foundation V5.1, SG24-6318.

8.6.2 Release compatibility and versions
The WebSphere Business Integration Server Foundation V5.1 Business Process Container can run processes you created in earlier versions of WebSphere Application Server Enterprise. As an exercise, you may want to install and execute the EAR files from the V5.0.2 installs discussed in Chapter 6, “Deploying the existing sample solution” on page 191. If you do try this, some key points are:

- At deploy time process names must be unique (because the Template name in the BPEDB must be unique). The full name of your process template is
formed by the combination of the process name and the value of the ValidFrom parameter. At deploy time any name clash will cause the application install to fail. To avoid this, either:

- Rename one of the processes before you install the application.
- Deploy the application with a non null ValidFrom value.

Interruptible processes are long running by definition. If you modify the run-time environment while instances of a process are active, or in flight, errors may occur at item claim time or later. To avoid this you have to preserve the run-time environment for each generation of your process. Only after all in-flight process instances of a generation (that is, all instances with the same ValidFrom parameter value) complete, can you remove elements of the environment that are specific for that generation. Evaluate the ways you might do this based on your business requirements.

### 8.6.3 Installation using the Standalone Deployment Tool

**Note:** This section provides an overview of the standalone deploy capability WebSphere Business Integration Server Foundation V5.1. We did not fully investigate standalone deploy as part of our redbook project, but it is important that you are aware of the availability of the Standalone Deployment Tool.

You can create an EAR file using the process choreographer Standalone Deployment Tool (SDT). Its purpose is to allow you to deploy a business process that was *not* created with WebSphere Studio Application Developer Integration Edition. The process has to be BPEL based.

Functionally the tool, is a JACL script invoked at the command line using wsadmin. Among the inputs are a BPEL file with your business process definition, WSDL, and associated XSD files. Output is an EAR file that contains your business process. Once you have created the EAR, you install the EAR file in the normal manner.

The SDT is located in:

```
%WAS_HOME%\ProcessChoreographer\util\deployBPEL.jacl.
```

You run the `deployBPEL` command with an options file. The options file allows you to:

- Reassign partner endpoint references in the WSDL port binding table.
- Define the installation target server (or servers if you have a production environment).
- Specify the binding parameters for the application.
This capability provides one form of late binding for your processes. The implications for release management by operations engineers depend on how releases are conducted. You can complete your testing of a new release of your process choreographer application, certify it to go live, and then use the `deployBPEL` command to reset your binding parameters and service endpoints without any impact on the validated EAR file.

**Important:** This assumes you have no code artifacts in the EAR; these are packaged separately in this scenario.

You would install the EAR file you create with a ValidFrom parameter set by operations and you’re done with the new release.

The `deployBPEL` command is well documented in the InfoCenter.
Working with the redbook sample code

This appendix describes how to work with the redbook sample code supplied as part of the additional material of our redbook. For details of how to obtain the redbook additional material refer to Appendix B, “Additional material” on page 319.
Additional material files

In this section we describe the files contained in our redbook additional material and provide more details about how to work with the included sample code. We assume that you have already obtained the sample code zip as described in Appendix B, “Additional material” on page 319, and extracted all files from the sg246324.zip file as described in “How to use the Web material” on page 320.

When all files are extracted from the sg246324.zip, the sg246324 directory structure is created. This directory contains the following files and folders:

- Ch04Workspace
  This folder contains a WebSphere Studio Application Developer Integration Edition V5.1 workspace that includes the BPEL sample described and built in 4.3, “Applying BPEL to a supply chain scenario” on page 55.

- EARFiles
  This folder contains EAR files for the redbook sample application. For a description of these EAR files see 6.1.2, “Overview of code” on page 192. The files are:
  - AggServiceMigrationV5EAR.ear
    AggServiceMigrationV5EAR.ear is the sample application after it has been migrated to WebSphere Studio Application Developer Integration Edition V5.1 as described in Chapter 7, “Migrating to WebSphere Business Integration Server Foundation” on page 239. Note that this EAR file is exported from the WebSphere Studio Application Developer Integration Edition workspace PE Sample 51 so it includes the modules from ExchangeStub2WebEAR.ear and MessageTester.ear that are used only for development testing. The AggServiceMigrationV5EAR.ear also includes source code.
  
  If you want to deploy our redbook sample application to WebSphere Business Integration Server Foundation, we recommend that you first understand the sample application by running it in WebSphere Studio Application Developer Integration Edition and then building and exporting a new EAR file that does not include the development test applications and source code.

  - ExchangeStub2WebEAR.ear
    For more details of ExchangeStub2WebEAR.ear see 7.3.5, “The exchange partner link” on page 257.

  - PEUtil.jar

  - AggServiceEAR.ear
This is the original aggregation EAR file, before migration to WebSphere Studio Application Developer Integration Edition V5.1.

- EIS1EAR.ear
- EIS2_SRVEAR.ear
- EIS3_SRVEAR.ear
- ExchangeStub2EAR.ear
- MessageTester.ear

► Jacl

This folder contains a JACL script itso_bpeconfig.jacl, which is a modified version of the bpeconfig command described in “Business Process Container configuration with bpeconfig” on page 283. You can use our modified command to create a Business Process Container configuration with the values used in the ITSO test environment or as an example of how to modify the JACL to use values specific to your environment.

► PE Sample 5

This folder contains a WebSphere Studio Application Developer Integration Edition V5 workspace that can be used as the starting point for the setup of our redbook sample application, as described in 6.1.1, “Obtaining the sample code” on page 192.

► PE Sample 51

This folder contains a WebSphere Studio Application Developer Integration Edition V51 workspace that contains the migrated code for our redbook sample application, as described in 7.4, “Running BPEL processes in WebSphere Studio V5.1” on page 263.

► Calculator.java

This Java source file is used when you create a sample BPEL process using WebSphere Studio Application Developer Integration Edition, as described in 5.6.5, “Sample BPEL process” on page 128.

► ExchangeDBFile.xml

This XML file is a sample of the file used by the exchange application to store part data. The aggregation process will fail if a file of this name is not found. The file should exist in either the WebSphere Studio Application Developer Integration Edition install directory if you are running the exchange on a WebSphere Studio test server, or in the WebSphere Application Server install directory if you are using a production server. You can either use our sample file or you can run the exchange GUI as shown in Figure 3-1 on page 33 and create the file by selecting Save Exchange DB to file.
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/SG246324

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, SG246324.
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>SG246324.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:** 165 MB minimum
- **Operating System:** Windows 2000 or Windows XP Professional
- **Processor:** 1 Ghz or higher
- **Memory:** 1 GB 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. Refer to Appendix A, “Working with the redbook sample code” on page 315, for more details of the contents of the SG246324.zip file and how to use the sample code.
# Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPE</td>
<td>Business Process Engine</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Execution Language</td>
</tr>
<tr>
<td>BPEL4WS</td>
<td>Business Process Execution Language for Web Services</td>
</tr>
<tr>
<td>BPMI</td>
<td>Business Process Modeling Initiative</td>
</tr>
<tr>
<td>BPML</td>
<td>Business Process Modelling Language</td>
</tr>
<tr>
<td>cXML</td>
<td>Commerce XML</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>ebXML</td>
<td>Electronic Business XML</td>
</tr>
<tr>
<td>EDI</td>
<td>Electronic Data Interchange</td>
</tr>
<tr>
<td>EIS</td>
<td>Enterprise Information Systems</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise Java Bean</td>
</tr>
<tr>
<td>FAQ</td>
<td>Frequently Asked Questions</td>
</tr>
<tr>
<td>FDML</td>
<td>Flow Definition Markup Language</td>
</tr>
<tr>
<td>JAX-RPC</td>
<td>Java API for XML-based RPC</td>
</tr>
<tr>
<td>JCA</td>
<td>J2EE Connector Architecture</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</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>MVC</td>
<td>Model-view-controller</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>PE</td>
<td>Private Exchange</td>
</tr>
<tr>
<td>SDT</td>
<td>Standalone Deployment Tool</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>SCM</td>
<td>Software configuration management</td>
</tr>
<tr>
<td>UBL</td>
<td>Universal Business Language</td>
</tr>
<tr>
<td>UBR</td>
<td>UDDI Business Registry</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description, Discovery, and Integration</td>
</tr>
<tr>
<td>URI</td>
<td>Universal Resource Identifier</td>
</tr>
<tr>
<td>WS-CDL</td>
<td>Web Services Choreography Description Language</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language</td>
</tr>
<tr>
<td>WSIF</td>
<td>Web Services Invocation Framework</td>
</tr>
<tr>
<td>xCBL</td>
<td>XML Common Business Library</td>
</tr>
<tr>
<td>XHTML</td>
<td>Extensible HyperText Markup Language</td>
</tr>
<tr>
<td>XMI</td>
<td>XML Metadata Interchange</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>XPath</td>
<td>XML Path Language</td>
</tr>
<tr>
<td><strong>XPATH</strong></td>
<td>XML Path Language</td>
</tr>
<tr>
<td><strong>XSL</strong></td>
<td>Extensible Stylesheet Language</td>
</tr>
<tr>
<td><strong>XSLT</strong></td>
<td>Extensible Stylesheet Language Transformation</td>
</tr>
</tbody>
</table>
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 325. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere Business Integration Server Foundation V5.1*, SG24-6318
- *WebSphere Application Server Enterprise V5 and Programming Model Extensions WebSphere Handbook Series*, SG24-6932
- *Exploring WebSphere Studio Application Developer Integration Edition 5.0*, SG24-6200
- *Patterns: Serial and Parallel Processes for Process Choreography and Workflow*, SG24-6306

Online resources

These Web sites and URLs are also relevant as further information sources:

- Java API for XML-Based RPC (JAX-RPC)
  
  http://java.sun.com/xml/jaxrpc
- XML Path Language
  
  http://www.w3.org/TR/xpath
- W3C Web Services Architecture
  
  http://www.w3.org/TR/ws-arch
- W3C Web Services Description Language
  
  http://www.w3.org/TR/wsd1
- Oasis UDDI
  
  http://www.uddi.org
- Business Process Modeling Language
  
  http://www.bpmi.org/bpm1.esp
- Web Services Flow Language
- XLANG Web Services for Business Process Design
  http://www.gotdotnet.com/team/xml_wsspecs/xlang-c
- W3C Web Services Choreography Description Language
  http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427
- Business Process Execution Language for Web Services Version 1.1
- XCPL.org XML Common Business Library
  http://www.xcbl.org
- RosettaNet Home
  http://rosettanet.org
- ebXML
  http://www.ebxml.org
- Commerce XML resources
  http://www.cxml.org
- Universal Business Language
  http://www.oasis-open.org/committees/ubl
- WebSphere Business Integration Server Foundation
  http://www.ibm.com/software/integration/wbisf/requirements
- WebSphere Studio Application Developer
  http://www.ibm.com/software/awdtools/studioappdev/
- WebSphere Studio Application Developer Integration Edition
  http://www.ibm.com/software/awdtools/studiointegration/
- WebSphere Studio Enterprise Developer
  http://www.ibm.com/software/awdtools/studioenterprisedev/
- WebSphere Studio Application Developer system requirements
  http://www.ibm.com/software/awdtools/studioappdev/sysreq
- WebSphere Studio Site Developer system requirements
- WebSphere Studio Application Developer Integration Edition system requirements
  http://www.ibm.com/software/integration/wsadie/requirements/
How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
Index

Numerics
5250-based applications 92

A
abstract process 47
ACID 27
activities 67, 119, 121
   BPEL editor 127
actor 59
aggregation 33–34, 194, 211, 215, 245
   process 35
AggServiceEAR 192–193, 232
AggServiceMigrationV5 259
AIX 94
ANSI 13
ANSI ASC X.12 13
Apache 87
application development 8
applications
   add to test server 171
   remove from test server 171
approval 18, 31, 193, 235, 245, 252
   invocation errors 252
   process 35, 210
assign
   create 155
atomicity 27
auction 16
authentication 205, 305
authorization 53
availability 19, 21

B
backup 299
binding 44, 107
BPE 36, 199, 244, 272
bpecheck 297
bpeconfig 283
bpecontainer.jar 273
BPEL xi, 1–2, 4–6, 8, 37, 39, 43–44, 54, 63, 81, 95, 239, 247, 252, 263, 272, 301
   activities 67, 119, 121
   compensation 50, 120
   correlation set 49, 65, 120
   debugger 54
   deployment 159
   editor 6, 56, 64, 95, 105–106, 119–120
      activities 127
      business process 124
      partner link 124
      tasks 119, 124
      variables 126
   elements 49, 120
   empty 50
   event handling 50
   fault handling 50, 120
   flow 50
   invoke 50, 67
   link 50
   partner link 64
   partner link type 49
   partners 119
   pick 50
   process 83
   receive 50, 67
   reply 50
   sequence 50
   staff assignment 120
   supply chain 55
   switch 50
   variable 49, 65
   variables 120
   wait 50
   while 50
BPML 48
business event 47
business integration 2, 8, 11
business integration perspective 101
business partners 51
business problem 16
business process 30
   BPEL editor 124
   create 136
   example 128
Business Process Choreographer xi, 1–8, 221
Business Process Container 223, 272

© Copyright IBM Corp. 2004. All rights reserved.
EIS 30, 90
EIS1 18–19, 30, 193–194, 231, 234
  item information 17
testing 235
EIS1EAR 192, 232
EIS2 18, 30–31, 194
  item pricing 17
EIS2_SRVEAR 192, 194, 232
EIS3 18, 30–32
  item availability 17
EIS3_SRVEAR 192, 232
EJB 32, 44, 54, 84, 88, 207, 257, 273
Electronic Business XML 52
electronic commerce 11, 13–14
Electronic Data Interchange See EDI
elements 106, 120
  elements pane 109
empty 50
Enterprise Archive See EAR
Enterprise Generation Language (EGL) 90
Enterprise information systems See EIS
Enterprise JavaBeans See EJB
environment space 274
environmental 19, 25
errors 247
event handling 50
event
  business process 128
  WSDL 115
exchange 14, 16, 33, 35, 194
  auction 16
  buyer aggregation 15
closed tender 16
  horizontal 16–17
large buyer 15
large seller 15, 17
negotiated price 16
neutral 15
private 15, 17
public 15
reverse auction 16
seller aggregation 15
set price 16–17
vertical 15
exchange partner link 257
ExchangeStub2EAR 192, 194, 232
executable process 47
export 184, 269
  EAR 188
Extended Messaging 31
Extensible Stylesheet Language Transformations
See XSLT
external interfaces 47
F
  fault handling 50, 120
  FDML 1–3, 8, 37, 48, 55, 95, 239, 248
    editor 106
File Transfer Protocol 14
flow 50
Flow Definition Markup Language, See FDML
FTP 14
functional requirements 18
G
generate deploy code 159, 263
H
  horizontal exchange 16–17
  Host-on-Demand 89
HTML 54
HTTP 32, 40, 44
HTTP server 53
I
  IBM Agent Controller 92
  IBM Distributed Debugger 91, 93
  IBM Portal Toolkit 88
  IBM System House 16, 30
  IBM WebFacing Tool 91–92
  ILE C 91
  ILE C++ 91
  ILE COBOL 91
  ILE RPG 91
import
  sample code 197
IMS 89–90, 94
installation
  Business Process Container 221, 271, 273, 275
  DB2 272
  WebSphere Studio Application Developer Integration Edition 96
integration
  supply chain 5
  supply chain 5
inventory 12, 31
invocation errors 252
invoke 50, 67
create 151
iSeries 91
isolation 27
item description 16
items
  approval 33

J
J2EE 41, 53, 84, 88, 273
J2EE hierarchy view 104
J9 55
JAAS 204
JAACL 283
JAR 242
Java 40, 44, 84, 88, 102, 273
Java 2 Platform, Enterprise Edition See J2EE
Java Build Path 178
Java Community process 41
Java Message Service See JMS
Java perspective 197
Java Servlets 84
JavaServer Faces 88
JavaServer Pages See JSP
JAX-RPC 41, 95
JCA 31, 89
JCL 90
JDBC 224, 275, 305
JDK 86, 93
JMS 35, 37, 44, 53, 95, 172, 253, 267, 273, 277
JMS connection factory 202, 204
JMS destination 202, 206, 267
JMS Provider 194, 197
JMS provider 224, 276
JNDI 207, 267
JRE 94
JSP 31, 84, 88
JSR 88
JSR 168 88
JSR-101 41

L
large buyer 15
large seller 15, 17
LDAP 53
link 50
links 242

listener ports 175, 202, 207, 231
location 13
logging 279
logistics 12
long running process 74, 248

M
maintainability 19, 22
manageability 19, 24
market 14
mass update 251
  process 36
  testing 215
MDB 32, 252
message 71, 107
message driven bean See MDB
MessageTester 192, 232
Migrate command 245
migration 6, 8, 37, 239
  aggregation process 245
  approval process 245
  errors 247
  workspace 242
missing output variables 248
model-view-controller 90
model-view-controller See MVC
MVC 86
MVS 90

N
negotiated price 16
neutral exchange 15
nodes 255
non-functional requirements 18

O
OASIS 46, 52
Object REXX for Windows Interpreter Edition 93
operation 44, 71, 107
optimization 13
orchestration 47
OSI 40

P
package explorer view 103
panes 108
part 71, 107
partner link 64, 75
  BPEL editor 124
  create 139
partner link type 49
partners 119
PEAGGIN 193–194
performance 19–20
perspective 101
  Java 197
PEUtil 192, 195
pick 50
plug-ins 84
port 44, 107
port type 44, 49, 62, 71, 107
  approval 253
portal xi
portlet 88
pricing 16
private exchange 5, 11, 15, 17, 30, 301, 306
  business problem 16
  requirements 18
  scenario 16
private executable process 70
process 1–2, 8, 33, 47, 83
  abstract 47, 56
  aggregation 34–35
  approval 35, 210
  descriptions 34
  executable 47, 70
  integration 4
  long running 74, 248
  mass update 36, 215
process abstract 60
process debugger 95
process flow 56
process interface
  create 133
  WSDL 133
production 12
project 102
  build path 176
prototype 16
public abstract process 60
public exchange 15
publish 208
purchasing scenario 56
Q
queue 206, 222
queue connection factory 228
queue destination 174, 229
queue manager 222
R
Rational ClearCase LT 92
receive 50, 67
Redbooks Web site 325
  Contact us xiv
Remote System Explorer (RSE) 92
reply 50
requirements 11
  availability 21
  availability 19
  data integrity 20, 27
  environmental 19, 25
  functional 18
  maintainability 19, 22
  manageability 19, 24
  non-functional 18
  performance 19–20
  private exchange 18
  scalability 19, 21
  security 19, 23
  usability 20, 26
response 248
response time 20
reverse auction 16
REXX 90
roles 49, 71
RosettaNet 48, 51
S
sample code 300
  deployment 220
  EAR 192
  import 197
  scalability 19, 21
scenario 2
schedule 215
SCM 86
scope 50
security 19, 23, 169, 203, 224, 266, 305
seller 56
seller aggregation 15
sellers 14
sequence 50
sequence diagram 57
servers 275
service 44, 107
  create 129
service project
  create 166
service projects 102
services view 102
set price 16–17
Simple Mail Transfer Protocol 14
single sign-on 53
SMTP 14
SOA 53
SOAP 14, 32–33, 41, 43–44
software configuration management See SCM
SQL 90
staff 36, 203, 247, 311
staff activity 212
staff assignment 120
Standalone Deployment Tool 312
standards 39–40, 46, 84
static volumetrics 20
Struts 86, 90
supplier 16, 30, 33, 35
supply chain 2, 4–5, 7, 11, 46
  concepts 12
  inventory 12
  location 13
  ntegration 5
  optimization 13
  production 12
  transportation 13
  using BPEL 55
supply change
  exchange 14
switch 50
  create 144
System House 1–2
TCP/IP 40
telecommunication 13
test server
  configuration 202
  create 167, 198, 265
  JMS properties 172
  publish 209
  security 169
  start 208, 267
testing 235, 268
  approval process 210
  EIS1 235
throughput 20
Tomcat 87
toolbar 102, 105
transactions 14, 27
transportation 13
TSO 90

U
UBL 52
UBR 46
UDDI 6, 43, 45, 53
UDDI Business Registry 46
UML 57, 84, 88
  sequence diagram 57
UN/CEFACT 13
UN/EDIFACT 13
Unified Modeling Language See UML
Universal Business Language 52
Universal Description, Discovery, and Integration
See UDDI
URL 44
usability 20, 26
use case 16
  create new item entry 19
  update item entry 19
users 311
utilization 20

V
Value Added Networks 13
VANs 13–14
variable 49, 65, 75, 248
variables 120
  BPEL editor 126
vertical exchange 15
views 102, 108
VisualAge for Java 91
VisualAge Generator Developer 93
VisualAge Java, Enterprise Edition 93
VisualAge RPG 91, 93
VisualAge Smalltalk Enterprise 93

W
W3C 43, 45, 48, 108
wait 50
WDSL editor 105
Web 88
Web server 53
Web service 237, 257
Web services 7–8, 14, 16, 42, 51, 53, 84, 88
Web Services Architecture 43
Web Services Choreography Description Language 43
Web Services Description Language See WSDL
Web Services Flow Language 48
Web Services Gateway xi, 6, 32, 53
Web Services Invocation Framework See WSIF
WebSphere Administrative Console 273, 275
logon 275
WebSphere Application Server 5, 8, 31, 89
WebSphere Application Server - Express 88
WebSphere Application Server Enterprise 91, 191, 272
  deployment 191, 220
WebSphere Business Integration Server Foundation xi, 1–5, 7–8, 52–53, 89–90, 99, 239, 272, 282, 301
  deployment 271
WebSphere Commerce xi, 35
WebSphere Development Studio Client for iSeries 91–92
  Overview 91
WebSphere Development Studio for iSeries 91
  Overview 91
WebSphere Development Tools for iSeries 91
  Overview 91
Websphere JMS Provider 204
WebSphere MQ 31, 36, 220, 222, 272, 281, 301
WebSphere Studio 83–86
  export 184
  Platform support 93
WebSphere Studio Application Developer 88, 93
  Overview 88
WebSphere Studio Application Developer Integration Edition xi, 1, 5, 8, 39, 44, 52, 54, 83, 89, 94, 165, 197, 239
  common tasks 165
  installation 96
  Overview 89
  sample deployment 191
WebSphere Studio Asset Analyzer 91
WebSphere Studio Enterprise Developer 90
  Overview 90
WebSphere Studio Site Developer 87–88, 93
  Overview 87
WebSphere Studio Workbench 84–85, 87
while 50
wizard 95, 273, 275
workload 20
workspace 197, 241, 245
  migration 242
World Wide Web Consortium 43
WSA 43
WSAA 91
WS-Addressing 43, 49
wsadmin 273
WS-BusinessActivity 43
WS-CDL 43, 48
WS-Coordination 43
WSDL 14, 41, 43, 45, 49, 54, 59, 62, 81, 89, 102, 106, 252
  best practices 119
  binding 44, 107
  data types 108
  editor 44, 95, 106, 108
    bindings 113
    messages 112
    panes 108
    services 114
    tasks 110
    views 108
  elements 106
  example 115
  files 119
  message 43, 107
  operation 44, 107
  part 107
  parts 43
  port 44, 107
  port type 44, 107
  process interface 133
  service 44, 107
  using the editor 106
Using BPEL Processes in WebSphere Business Integration Server Foundation - Business Process Integration and Supply Chain Solutions

WS-Eventing 43
WS-Federation 43
WSFL 48
WSIF 41, 89, 251, 257
WS-Policy 43
WS-ReliableMessaging 43
WS-Routing 43
WS-SecureConversation 43
WS-Security 43
WS-Transaction 43
WS-Trust 43
WYSIWYG 88

X
X.400 messaging 13
xCBL 51
XLANG 48
XML 14, 31, 35, 40, 44, 51, 84, 88, 108
XML Common Business Library 51
XML Path Language See XPath
XML Pointer Language See XPointer 41
XML Schema 49, 108
XPath 41, 49
XPointer 42
XSLT 41, 89

Z
z/OS 90
This IBM Redbook examines business process integration using Business Process Choreographer to build solutions that help to streamline the supply chain process. We provide a sample scenario that integrates business processes to streamline a supply chain showing how suppliers and customers can use global repositories to synchronize data.

The sample scenario discussed in this publication was initially implemented using WebSphere Application Server Enterprise Process Choreographer to aggregate and transform supplier product data. Data is then published to a global repository using the WebSphere Web Services Gateway, and also loaded to a catalog that uses a WebSphere Commerce portal.

We take this existing scenario and detail how to migrate the business processes to run on WebSphere Business Integration Server Foundation V5.1. As part of this migration effort we describe how to use WebSphere Studio Application Developer Integration Edition V5.1 as the development environment for process migration. The migrated sample uses Business Process Execution Language (BPEL) processes.