Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server V6.2 and Best Practices

June 2009
**Second Edition (June 2009)**

This edition applies to the products listed in the following table.

<table>
<thead>
<tr>
<th>Version</th>
<th>Release</th>
<th>Modification</th>
<th>Product name</th>
<th>Product ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>XP</td>
<td></td>
<td>Service Pack 2</td>
<td>Microsoft Windows</td>
<td>5724-I75</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
<td>IBM WebSphere Business Integration Server Foundation</td>
<td>5724-H74</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>IBM WebSphere Studio Application Developer Integration Edition</td>
<td>5724-D15</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>IBM WebSphere Business Integration Modeler</td>
<td>5724-I74, 5724-I75</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td></td>
<td>IBM WebSphere Process Server</td>
<td>5724-L01, 5655-N53</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td></td>
<td>IBM WebSphere Integration Developer</td>
<td>5724-I66</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td></td>
<td>IBM WebSphere Business Modeler</td>
<td>5724-I75</td>
</tr>
</tbody>
</table>

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

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

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

Summary of changes .................................................. xvii
June 2009, Second Edition ........................................... xvii

## Part 1. Products overview and migration planning ................. 1

### Chapter 1. Introduction to the book ............................. 3
1.1 The scope of this book ........................................... 4
1.2 Intended audience ................................................. 4
1.3 The organization of this book .................................... 4
1.4 Scenarios demonstrated .......................................... 6
1.5 What is not covered in the book ................................. 6
  1.5.1 Installing and using software products .................. 7
1.6 Assumptions ....................................................... 7

### Chapter 2. Product overview ..................................... 9
2.1 WebSphere Business Integration Server Foundation overview .................................................. 10
  2.1.1 Architectural overview of WebSphere Business Integration Server Foundation .................. 10
  2.1.2 Features of WebSphere Business Integration Server Foundation .................................. 13
2.2 WebSphere Process Server overview ........................... 15
  2.2.1 Architectural overview of WebSphere Process Server .............................................. 16
  2.2.2 Features of WebSphere Process Server .......................................................... 22

### Chapter 3. Migration concepts ................................. 29
3.1 Migration concepts overview ..................................... 30
3.2 Features mapping and migration approach ..................... 31
  3.2.1 General feature comparison ................................. 31
  3.2.2 Source artifacts, bindings, and the migration approach ....................................... 35
3.3 Benefits of migrating to WebSphere Process Server ........ 37

### Chapter 4. Migration planning ................................. 39
4.1 Assessment phase ................................................ 40
4.1.1 Project planning ........................................... 42
4.1.2 Current environment assessment ......................... 45
4.1.3 Requirements gathering and impact analysis .............. 46
4.1.4 Service decomposition .................................. 56
4.1.5 Deliverables ............................................ 59
4.2 Preparation phase ......................................... 61
4.2.1 Building skills .......................................... 61
4.2.2 Using the WebSphere Process Server project experience 61
4.2.3 Refining the assessment ................................ 62
4.3 Implementation phase ..................................... 63
4.3.1 Development and migration ............................ 63
4.3.2 Testing .................................................. 64
4.3.3 Deployment .............................................. 64
4.3.4 Go live .................................................... 66
4.4 Other considerations ....................................... 66
4.4.1 Availability ............................................. 66
4.4.2 Security ................................................ 70
4.5 Product coexistence ....................................... 74

Part 2. Migration procedure ........................................... 77

Chapter 5. Best practices ........................................... 79
5.1 Planning .................................................... 80
5.2 Artifacts preparation ....................................... 80
  5.2.1 Artifact preparation for WebSphere Studio Application Developer Integration Edition ........................................ 81
  5.2.2 Preparing the modeler artifacts .......................... 81
  5.2.3 Schema XSD files ...................................... 81
  5.2.4 WSDL files and Web services ........................... 83
  5.2.5 BPEL artifacts .......................................... 84
  5.2.6 Client source .......................................... 85
  5.2.7 Modeling migration best practices ........................ 85
  5.2.8 Naming convention ...................................... 86
  5.2.9 Modeler naming convention .............................. 86

Chapter 6. Products runtime comparison ............................. 91
6.1 Products overview and comparison .......................... 92
  6.1.1 Feature comparison ..................................... 92
  6.2 Base application server level features ........................ 93
    6.2.1 Messaging in WebSphere Business Integration Server Foundation ........................................ 94
    6.2.2 Messaging in WebSphere Process Server .......................... 94
    6.2.3 Database required by WebSphere Business Integration Server Foundation ........................................ 98
    6.2.4 Database required by WebSphere Process Server .......................... 99
6.3 Business Process Choreographer Features ........................................ 100
  6.3.1 Business Process Choreographer applications .......................... 101
  6.3.2 Business Process Choreographer application’s security roles .... 102
  6.3.3 People Directory Provider configuration ................................. 103
  6.3.4 Business Rule Beans .......................................................... 105
  6.3.5 Extended messaging .......................................................... 106
  6.3.6 Common Event Infrastructure ............................................... 106
6.4 Administration in WebSphere Process Server using a console and scripts . 108

Chapter 7. Migration options ................................................................. 119
7.1 Migration paths .............................................................................. 120
  7.1.1 Migrating business processes described in BPEL ...................... 121
  7.1.2 Migrating business processes described in Flow Definition Markup
       Language .................................................................................. 122
  7.1.3 Migrating business processes mainly defined in WebSphere Business
       Integration Modeler ................................................................. 123
7.2 Migration tools ............................................................................... 124
  7.2.1 WebSphere Integration Developer ........................................... 124
  7.2.2 WebSphere Business Modeler ................................................... 142
  7.2.3 Command-line interface and scripts ....................................... 149
7.3 Migration approaches ....................................................................... 150
  7.3.1 Migrating with dependencies between applications ................ 151
  7.3.2 Considerations for long running processes ............................... 154

Chapter 8. Workspace migration .............................................................. 163
8.1 Introduction to the workspace migration ........................................ 164
8.2 General considerations ................................................................. 165
  8.2.1 General migration tasks for source artifacts ............................ 165
  8.2.2 Pre-migration considerations ................................................... 166
8.3 Preparing source artifacts for workspace migration ......................... 168
  8.3.1 Project organization changes in WebSphere Integration Developer168
       Edition workspace ..................................................................... 173
  8.3.3 Preparing the WebSphere Integration Developer workspace .... 174
8.4 Migrating workspace by using the WebSphere Integration Developer
       migration wizard ....................................................................... 175
  8.4.1 Overview of the workspace migration wizard ......................... 175
  8.4.2 Using the migration wizard ..................................................... 176
8.5 Migrating a workspace using a command-line interface and scripts .... 182
  8.5.1 Overview of the command-line script ..................................... 182
  8.5.2 Using the command-line script .............................................. 183
8.6 Additional migration considerations ................................................ 185
Chapter 12. Troubleshooting ......................................................... 359
12.1 General troubleshooting .................................................... 360
12.2 Migration troubleshooting .................................................... 361
  12.2.1 Importing into WebSphere Integration Developer ................. 361
  12.2.2 Building the project in WebSphere Integration Developer ......... 363
  12.2.3 Deploying and running the process in WebSphere Process Server .. 365
12.3 Scenario troubleshooting ...................................................... 366
12.4 Advanced troubleshooting .................................................... 369
  12.4.1 Information center and manuals ....................................... 369
  12.4.2 Product support Web site .............................................. 370
  12.4.3 IBM Support .............................................................. 371

Appendix A. Additional material ................................................. 373
Locating the Web material ......................................................... 373
Using the Web material ............................................................ 374
  How to use the Web material ................................................. 374

Abbreviations and acronyms ...................................................... 375

Glossary .................................................................................... 377

Related publications ............................................................... 383
IBM Redbooks ................................................................. 383
Online resources ............................................................. 383
How to get Redbooks ....................................................... 390
Help from IBM ............................................................. 391

Index .................................................................................... 393
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 illustrate 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.
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

- AIX®
- ClearCase®
- DB2 Universal Database™
- DB2®
- developerWorks®
- Domino®
- HACMP™
- IBM®
- Lotus Notes®
- Lotus®
- Notes®
- Parallel Sysplex®
- Rational®
- Redbooks®
- Redbooks (logo) ®
- System z®
- Tivoli®
- WebSphere®
- z/OS®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Red Hat, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, Enterprise JavaBeans, J2EE, J2SE, Java, JavaBeans, JavaServer, JDBC, JRE, JSP, JVM, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel Pentium 4, Intel Pentium, Intel, Pentium 4, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.
In this IBM® Redbooks® publication, we discuss the concepts, planning, and migration paths that you must understand before attempting to migrate source artifacts, which are created by using IBM WebSphere® Studio Application Developer Integration Edition 5.1, to IBM WebSphere Integration Developer 6.2. We also discuss how to migrate models that are developed in WebSphere Business Integration Modeler 5.1 to WebSphere Business Modeler 6.2.

In this book, we discuss the new features and differences between WebSphere Business Integration Server Foundation and WebSphere Process Server 6.2. We cover discussions around the considerations for target topologies to be migrated to, and we also cover the deprecated features of WebSphere Process Server 6.1. We also give you information that is critical to any upgrade path and help you realize the benefits of the IBM new breed of integration products.

In Part 1 of this book, we provide a product overview and help you to assess your environment and to plan a WebSphere Business Integration Server Foundation migration to WebSphere Process Server 6.2. We also discuss the new features of WebSphere Process Server that you can leverage.

Part 2 consists of best practices and detailed migration steps using the new WorkSpace migration tool to migrate the source artifacts in WebSphere Business Integration Server Foundation to WebSphere Process Server. We highlight changes and improvements between WebSphere Business Integration Server Foundation and WebSphere Process Server. We use practical migration scenarios to demonstrate the transition of WebSphere Business Integration Server Foundation business objects. Additionally, we cover client component migration, which includes how to migrate client applications. Using the information that we provide in the troubleshooting section you can resolve errors that you might encounter during the migration. We also tell you the general troubleshooting skills that are required when working with WebSphere Process Server 6.2.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Hursley Center.
Saida Davies is a Project Leader for the International Technical Support Organization (ITSO) and has published several Redbooks and Redpapers publications on WebSphere Business Integration, Web services, and WebSphere Service Oriented Middleware using multiple platforms. Saida has experience in the architecture and design of WebSphere MQ solutions, extensive knowledge of the z/OS® operating system, and a detailed working knowledge of both IBM and independent software vendors’ operating system software. Prior to joining the ITSO, she was a Senior IT Specialist who was responsible for the development of services for WebSphere MQ within the z/OS and Microsoft® Windows® platforms, which covered the architecture, scope, design, project management, and implementation of the software on stand-alone systems or on systems in a Parallel Sysplex® environment. She received Bravo awards for her project contributions. Saida has a degree in Computer Studies, and her background includes IBM System z® programming. Saida supports, contributes, and participates in Women in Technology activities and meetings.

Lei Jiang is an Advisory Software Developer for IBM Canada Toronto Software Lab. He joined IBM in 1997 and is currently working as a WebSphere Integration Developer Service and Support Team Lead. Lei was in WebSphere tooling development for five years and in WebSphere service team for six years. He helped many customers resolve design, development, and migration issues with WebSphere Integration Developer and WebSphere Studio Application Developer Integration Edition tools. He is one of the authors of the IBM Redbooks publication *Migrating IBM WebSphere Business Integration Server Foundation to WebSphere Process Server V6.1*, SG24-7673. Lei is an IBM certified Service-oriented architecture (SOA) associate and has a Master of Computer Science degree from the University of Nebraska, USA.
Simon Kofkin-Hansen is the Lead Web Middleware Consultant in IBM Application Hosting within the Integrated Technology Delivery division in Australia. He started his IT career in 1993, working with Internet and Networking-related technologies, before becoming a Software Engineer specializing in Lotus® Domino® and Notes. He joined IBM in 2007 and is currently working as a Lead Consultant in systems integration and automation specialist on numerous client accounts around the world. Simon has worked with WebSphere Products for six years, concentrating on WebSphere Business Integration Server Foundation and Process Server for the last three years. He is certified in Service-oriented architecture (SOA), WebSphere Process Server, WebSphere Network Deployment, and the Tivoli® range of Security Products.

Ji Yu is a Staff Software Engineer for the IBM China Development Lab. She joined IBM six years ago and is currently working in the WebSphere Process Server system test team. As the Team Lead, her main responsibility is to perform system and functional testing for migrating WebSphere InterChange Server to WebSphere Process Server and migrating WebSphere Business Integration Server Foundation to WebSphere Process Server. Ji Yu worked on the WebSphere Process Server product for four years. She is an IBM Certified System Administrator of WebSphere Application Server Network Deployment V6.0 and WebSphere Process Server Network Deployment V6.0. She has two developerWorks® articles and two IBM Redbooks publications; IBM WebSphere InterChange Server Migration to WebSphere Process Server, SG24-7415 and Migrating IBM WebSphere Business Integration Server Foundation to WebSphere Process Server V6.1, SG24-7673. She has a Bachelor of Software Engineering degree from Tsinghua University of China.
Peng Cheng Zhang is a Software Engineer for the IBM China Development Lab. He currently works in the WebSphere Process Server system test team. His main responsibility is to perform the system test for migrating WebSphere Server Foundation to WebSphere Process Server. Peng Cheng is an IBM Certified System Administrator of WebSphere Application Server Network Deployment V6.1. His areas of expertise include Java™, Java EE, and information security. He has a Masters degree from the Chinese Academy of Science.

Thanks to the authors of the previous editions of this book.

- The authors of the first edition, *Migrating IBM WebSphere Business Integration Server Foundation to WebSphere Process Server V6.1*, SG24-7673-00, published in October, 2008, were:

  Saida Davies
  Lei Jiang
  Simon Kofkin-Hansen
  Ji Yu
  Peng Cheng Zhang

**Become a published author**

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will 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](http://ibm.com/redbooks/residencies.html)
Comments welcome

Your comments are important to us!

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

- Use the online **Contact us** review Redbooks form found at:
  - [ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:
  - redbooks@us.ibm.com

- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400
Summary of changes

In this section, we provide the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7673-01
for Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server V6.2
as created or updated on June 15, 2009.

The previous edition is: Migrating IBM WebSphere Business Integration Server Foundation to WebSphere Process Server V6.1, SG24-7673-00 available at:

June 2009, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information.

New information

- Business Space
  Business Space is a browser-based graphical user interface that lets application users customize content from products in the WebSphere Business Process Management portfolio.

- Workspace Migration Tool
  Improves integration developer productivity by enabling the import of entire WebSphere Studio Application Developer Integration Edition workspaces into WebSphere Integration Developer and by aiding in the migration of service interfaces to custom Java code.
Changed information

► Scenarios

The scenarios are the same; however, the new workspace migration tools are demonstrated within each scenario. The TravelBooking scenarios, it demonstrates how to generate the client interface for the human task, and how to use the business space widgets to test the human task.

► Migration Options

Because there is a new migration wizard and command line parameters for the migration, this section changed significantly. In V6.1, migration tools can only migrate one service project in one time, which requires the developer to determine the project dependencies and order of the migration if multiple projects must be migrated. In the V6.2 migration tools, the migration tool can migrate the entire workspace, which contains multiple projects together. In addition, the wizard provides better migration support for other J2EE™ runtime artifacts.
Part 1 Products overview and migration planning

In this part, we provide an introduction and discussion for the scope covered in this book. We provide an overview of IBM WebSphere Business Integration Server Foundation and WebSphere Process Server. Furthermore, we discuss concepts and planning for migrating WebSphere Business Integration Server Foundation to WebSphere Process Server.

The chapters in this part are:

In Chapter 1, “Introduction to the book” on page 3, we discuss the scope and the intended audience of the book.

In Chapter 2, “Product overview” on page 9, we provide an overview of WebSphere Business Integration Server Foundation and WebSphere Process Server. In this chapter, we also focus on the new features that are available in WebSphere Process Server.
In Chapter 3, “Migration concepts” on page 29, we compare the features and functions of the two products and introduce the new features of WebSphere Process Server. We also discuss the migration approaches and the benefits of migrating to WebSphere Process Server.

In Chapter 4, “Migration planning” on page 39, we cover the recommended approach on how to plan a migration to WebSphere Process Server and the different phases during the migration process.
Introduction to the book

The topics that we discuss in this chapter are:

- 1.1, “The scope of this book” on page 4
- 1.2, “Intended audience” on page 4
- 1.3, “The organization of this book” on page 4
- 1.4, “Scenarios demonstrated” on page 6
- 1.5, “What is not covered in the book” on page 6
- 1.6, “Assumptions” on page 7
1.1 The scope of this book

In this book, we aim to provide a migration guide for migrating WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.2, planning the migration, and identifying specific migration issues.

Using WebSphere Process Server migration tools with source artifacts from WebSphere Business Integration Server Foundation V5.1, in the scenarios we demonstrate how to leverage WebSphere Business Integration Server Foundation capabilities and functionalities to the new environment and programming model of WebSphere Process Server. The migration is verified to ensure successful transition, and we document the comparison of WebSphere Business Integration Server Foundation artifacts with WebSphere Process Server artifacts.

1.2 Intended audience

In this book, we target a broad audience that begins with managers, IT architects, solutions designers, and integration developers who were involved with WebSphere Business Integration Server Foundation projects.

Chapter 3, “Migration concepts” on page 29, Chapter 7, “Migration options” on page 119, are intended for architects and solution designers but are useful to developers as well.

Chapter 4, “Migration planning” on page 39, is intended for project management.

The chapters in Part 2, “Migration procedure” on page 77 are intended for developers. The content requires an intermediate technical skill level with regard to WebSphere Adapters, WebSphere Integration Developer, and WebSphere Process Server.

1.3 The organization of this book

This book is divided into two parts.

Part 1 provides product overviews, helps you assess your environment, and helps you plan WebSphere Business Integration Server Foundation V5.1 migration to WebSphere Process Server V6.2.
The chapters in Part 1 are:

- Chapter 1, “Introduction to the book” on page 3 provides an introduction to the book. We discuss the scope and the intended audience of the book.
- Chapter 2, “Product overview” on page 9 provides an overview of WebSphere Business Integration Server Foundation V5.1 and WebSphere Process Server V6.2. In this chapter, we also focus on the new capabilities of WebSphere Process Server V6.2.
- Chapter 3, “Migration concepts” on page 29 compares the features and functions of the two products and does a functional mapping between each product feature. In this chapter, we also explain the benefits of migrating to WebSphere Process Server V6.2.
- Chapter 4, “Migration planning” on page 39 covers the recommended approach on how to plan a migration to WebSphere Process Server V6.2.

Part 2 provides the detailed steps that are required for migration, the tools that are available, the artifacts that are involved, and best practices. In this part, we also demonstrate the migration process using practical migration scenarios that cover most of WebSphere Business Integration Server Foundation V5.1 capabilities, which includes business processes, human activities and staff assignment, data connectors and data mapping, exception, and error handling.

The chapters in Part 2 are:

- Chapter 5, “Best practices” on page 79, provides the best practices to use when you migrate from WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.2.
- Chapter 6, “Products runtime comparison” on page 91, discusses the differences between runtime environments of WebSphere Business Integration Server Foundation and WebSphere Process Server.
- Chapter 7, “Migration options” on page 119, describes the common and supported paths for migrating WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.2.
- Chapter 8, “Workspace migration” on page 163, explains how to migrate artifacts that were developed in the WebSphere Application Developer Integration Edition V5.1 using the new workspace migration wizard that WebSphere Integration Developer V6.2 provides.
- Chapter 9, “Client component migration” on page 193, considers approaches for migrating the client code of the business process (the migration wizard cannot automatically migrate the client code) from WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.2.
- Chapter 10, “Post migration tasks” on page 215, provides the best practices to refactoring and tuning business process after migration. In this chapter, we
also discuss how to add enhanced features of WebSphere Process Server into the migrated content.

- Chapter 11, “Technical scenarios” on page 223, includes step-by-step instructions on migration scenarios and illustrates how to use the migration tools.
- Chapter 12, “Troubleshooting” on page 359, discuss troubleshooting for migrating WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.2.

1.4 Scenarios demonstrated

The migration scenarios that we describe and demonstrate in this book cover a broad range of WebSphere Business Integration Server Foundation V5.1 functionality to provide a practical migration path. The scenarios do not cover the complexity of the WebSphere Business Integration Server Foundation solution, which requires WebSphere Process Server education skills assessment, planning, and analysis through a normal development and deployment life cycle.

The products that we used in the migration scenarios that we performed are:

- WebSphere Business Integration Server Foundation V5.1.1
- WebSphere Integration Developer V6.2 with Integrated WebSphere Process Server development environment
- WebSphere Process Server V6.2
- WebSphere Business Integration Modeler V5.1.1
- IBM WebSphere Business Modeler Advanced Version 6.2

1.5 What is not covered in the book

In this section, we provide information about the topics that we do not cover in this book.

This book focuses on tasks that relate to the migration to WebSphere Process Server V6.2 from WebSphere Business Integration Server Foundation V5.1.

This book does not cover:

- Details about the installation of software products, which includes WebSphere Business Integration Server Foundation, WebSphere Integration Developer, and WebSphere Process Server; therefore, no step-by-step instructions are included.
1.5.1 Installing and using software products

Refer to the installation and system requirements documentation for further details about WebSphere Integration Developer V6.2 and WebSphere Process Server V6.2 products at:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp
http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp

Refer to the installation and system requirements documentation for further details about WebSphere Business Integration Server Foundation V5.1 at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp

1.6 Assumptions

In this book, we assume that:

- You are familiar with WebSphere Business Integration Server Foundation V5.1.
- You understand Eclipse and the Java/J2EE platform.
- You have working knowledge of WebSphere Application Server V6.
- You have a ready environment with both WebSphere Integration Developer V6.2 and WebSphere Process Server V6.2 installed and configured.
Product overview

In this chapter, we include the following sections:

- “WebSphere Business Integration Server Foundation overview” on page 10
- “WebSphere Process Server overview” on page 15
2.1 WebSphere Business Integration Server Foundation overview

IBM WebSphere Business Integration Server Foundation V5.1 is an evolution of the way that the WebSphere family of products implements service-oriented architecture (SOA) solutions. WebSphere Business Integration Server Foundation V5.1 extends the capabilities of the WebSphere Application Server to provide a premier Java 2 Enterprise Edition (J2EE) and Web services technology-based application platform for deploying enterprise Web service solutions for dynamic on-demand business. It represents an important step in the path that IBM proposed as an approach to build and deploy SOA-based applications.

The goal of WebSphere Business Integration Server Foundation V5.1 is to increase business flexibility by leveraging an SOA to build modular applications that are designed to adapt quickly to change. WebSphere Business Integration Server Foundation proposes to build solutions as composite applications by using a highly-integrated development environment.

The topics that we discuss in this section are:

- 2.1.1, “Architectural overview of WebSphere Business Integration Server Foundation” on page 10
- 2.1.2, “Features of WebSphere Business Integration Server Foundation” on page 13

2.1.1 Architectural overview of WebSphere Business Integration Server Foundation

WebSphere Business Integration Server Foundation uses the functionalities of the WebSphere Application Server and introduces J2EE applications to offer a complete SOA platform. WebSphere Business Integration Server Foundation provides the following key additional services:

- The Business Process Choreographer, which is a powerful container for executing business processes
- Programming model extensions, which provide a number of valuable extensions to the J2EE specification that are delivered in many different forms

The extensions include services, APIs, and tooling, for example, Business Rule Beans is one extension that is a business rule engine part of WebSphere Business Integration Server Foundation.
The Common Event Infrastructure, which was introduced as a technical preview with Version 5.1 and became part of the product in V5.1.1

Figure 2-1 shows the architecture of WebSphere Business Integration Server Foundation as an application server with extended capabilities.

**Figure 2-1  The WebSphere Business Integration Server Foundation architecture**

### Business Process Choreographer

WebSphere Business Integration Server Foundation introduces the ability to manage processes that are defined as a choreography of services. The business process choreography is about the development and execution of business process flow logic, in the way the flow is abstracted from the application.

WebSphere Business Integration Server Foundation V5.1 supports the execution of business processes that are defined in the Business Process Execution Language for Web services (BPEL4WS). BPEL4WS is a proposal for a standard for describing the definition of a business process. It controls the order, sequence, and data of service invocations, which in turn support business processes and workflows.
Business Process Choreographer, which is within WebSphere Business Integration Server Foundation, is the runtime environment for executing business processes. It provides a way to combine workflow technology with the services that J2EE offers within WebSphere Application Server. In addition to workflow enabling J2EE applications and components, the intent allows for independent Web services to be coordinated along with J2EE applications. Business Process Choreographer consists of a J2EE application, the business process engine container, a set of client interfaces, and administration and control facilities.

**Business process engine container architecture**

In this section, we discuss the architecture for the business process engine container.

- The *business process engine container* is a specialized J2EE application that executes business processes by using WebSphere Application Server runtime services and resources. It handles the life cycle of a process from the instantiation of a process template to the final deletion of the completed process. Figure 2-2 shows the components of the container.

![Business process engine container architecture](image)

The functionality of the container is published as a Session Enterprise JavaBean (EJB™) and a message-driven bean (MDB). It provides an external queue for client interaction by using Java Message Service (JMS), internal queues and a relational database, and the Business Process Choreographer database.
(BPEDB) to manage the transitions between activities of certain processes and to persist process states.

**Process Navigation**

The Process Navigation function that is within Business Process Choreographer manages the state and life cycle of each process instance. A process instance begins when the navigator receives a start request (through a business process engine Web client or one of the provided APIs). The navigator manages the data flow between activities, changes their status, manages failures, and triggers the invocation of compensation. Also the navigator performs monitoring tasks for a process instance by using its observer and audit trail components.

The navigator also includes functions to perform the following operations:

- Java snippet invocations
- External process invocations

With Process Navigation, you can invoke the external processes by using the Apache Web Services Invocation Framework (WSIF).

**People Interaction**

BPEL4WS does not include definitions for human interactions. The People Interaction component implements an extension to BPEL4WS that supports human interaction within Business Process Choreographer. Their major component is the Work Item Manager, which is responsible for resolving staff resolution queries from process participants by making use of one of the provided staff plug-in providers.

### 2.1.2 Features of WebSphere Business Integration Server Foundation

The WebSphere Business Integration Server Foundation platform offers a set of important features:

- Initial enablement for SOA
- BPEL4WS process choreography version 1.1 with extensions
- Human workflow support
- Business Rule Beans support
- Application adapters
- Programming model extensions
- CEI
- J2EE Application Server
- Integrated J2EE development environment
WebSphere Business Integration Server Foundation uses Business Process Execution Language (BPEL) version 1.1 with some IBM extensions.

WebSphere Business Integration Server Foundation V5.1 extends the WebSphere Application Server 5.1 functionalities with a set of J2EE applications. These solutions offer a whole platform called programming model extensions (PMEs). The PMEs address the following areas:

- An integrated J2EE-based workflow, the Business Process Execution Container
- A CEI
- An Advanced Transactional service
- A Web service platform
- A CORBA platform
- A set of development tools and functionalities as a scheduler service, a business rule engine, an object pools service, application profiling, and asynchronous beans

Figure 2-3 shows the elements of WebSphere Business Integration Server Foundation.

![Figure 2-3 Elements of the WebSphere Business Integration Server Foundation](image)

Because the scope of this book is oriented to the migration of processes from WebSphere Business Integration Server Foundation to IBM WebSphere Process Server, we do not cover the following original elements and functionalities in depth in this document:

- The CORBA platform

  The CORBA IDL interface API is not supported in WebSphere Process Server. The CORBA technology was a bridge for migration to a J2EE and WebSphere Application Server environment.
2.2 WebSphere Process Server overview

WebSphere Process Server is an advanced business process integration server that is built over the industry standard SOA.

WebSphere Process Server uses the SCA programming model and the Service Data Objects (SDOs) data model. Business objects are transformed, routed, and mapped by using the SCA components. WebSphere Adapters and WebSphere Business Integration Adapters provide the connectivity to the back-end Enterprise Information Systems (EIS). WebSphere Process Server uses Web services’ BPEL to define the business processes with the IBM extensions to BPEL for human tasks and business rules. WebSphere Enterprise Service Bus is built into WebSphere Process Server and can be used to provide the connectivity to Web services and the mediation flows. You can use the CEI, which is within the WebSphere Process Server, to monitor business events that can occur in the applications that are running on WebSphere Process Server.

WebSphere Process Server is complemented by WebSphere Integration Developer, which is a tool for developers to easily develop and deploy business integration solutions to WebSphere Process Server. You can use WebSphere Modeler to model the business processes for WebSphere Process Server. In addition, you can use WebSphere Monitor to monitor the business processes that are running on WebSphere Process Server.

WebSphere Process Server is based on the J2EE 1.4 infrastructure and the WebSphere Application Server platform. It is built on WebSphere Application Server Network Deployment.

In this section, we discuss:

► 2.2.1, “Architectural overview of WebSphere Process Server” on page 16
► 2.2.2, “Features of WebSphere Process Server” on page 22
2.2.1 Architectural overview of WebSphere Process Server

IBM WebSphere Process Server is a BPEL-based business process engine that is built on a service-oriented architecture integration platform with a uniform data and invocation model. WebSphere Process Server provides components that are built over the SCA to extract the implementation of business logic from the integration logic.

WebSphere Application Server Network Deployment provides the base runtime infrastructure for WebSphere Process Server. It provides such qualities as clustering, failover, scalability, and security to WebSphere Process Server.

WebSphere Process Server has three logical layers:

- The SOA core layer
  The SOA core layer consists of the SCA, the business objects, and the CEI component. The SCA and the business objects provide universal invocation and the data representation programming models. The CEI generates events for monitoring applications that are running on WebSphere Process Server.

- The supporting services layer
  The supporting services layer provides the components that are necessary for transformations of business objects and the interfaces.

- The service components layer
  The service component layer provides the functional components that are required for composite applications.

The combination of all of the components in WebSphere Process Server and the development environment that WebSphere Integration Developer provides, coupled with the usage of WebSphere Adapters and WebSphere Business Integration Adapters, facilitate quick development and deployment of integration solutions.

Figure 2-4 on page 17 shows the components of the WebSphere Process Server.
**Service-oriented architecture core**

The SOA core of the WebSphere Process Server provides the universal invocation (SCA) and the data-representation (business objects) programming models and the monitoring and management capabilities (CEI) for applications that are running on WebSphere Process Server. IBM Enterprise Service Bus is also a part of the SOA core of WebSphere Process Server, where it provides the transformation and mediation services.

Figure 2-5 shows the SOA core layer of the WebSphere Process Server.
**Service Component Architecture**

The SCA provides an abstraction layer around the service components that separate the business logic from the integration logic. In doing so, developers who are responsible for the business logic can work on the business logic, and the integration specialists can work on integrating service components.

A component within the SCA has an interface for using or invoking the service, an implementation, and a reference, if the component wants to use the services of another component. See Figure 2-6.

![Diagram](image)

Figure 2-6  Service Component Architecture

**Service Data Objects and business objects**

Data that is exchanged between the components that are running on WebSphere Process Server is represented by using business objects. A business object contains a set of attributes that represent the business data, an action on the data, such as a create or a delete action, and instructions for processing the data.

Business objects in WebSphere Process Server are based on the SDOs, which are part of the WebSphere Application Server. An SDO is a framework for the data application development. The SDO abstracts the data in a service-oriented way. It provides a technology-independent representation of the data that enables developers to work with the data without having to worry about the technology-specific API to access and use the data.
Business objects include extensions to SDOs that are important for integration solutions and are used to further describe the data that is being exchanged between the SCA components.

**Common Event Infrastructure**
The CEI that is within the WebSphere Process Server provides event management services. The CEI also provides functionality for generation, propagation, persistence, and consumption of events that represent service component processes. Events are represented by using the Common Base Event model, which is a standard, XML-based format that defines the structure of an event. The CEI provides standard interfaces and services for the applications to create events, store events, send events, and retrieve events.

**WebSphere Enterprise Service Bus**
WebSphere Enterprise Service Bus provides the connectivity and the integration needs of Web service applications and the data. WebSphere Enterprise Service Bus routes and transforms messages between the service requestor and the service provider. It also transforms the transportation protocol (for example SOAP/HTTP to SOAP/JMS) between the service requestor and service provider.

**Supporting services**
The supporting services layer in WebSphere Process Server addresses the transformation challenges for connecting components and the external artifacts. Mediation flows, business object maps, relationships, and selectors form the supporting services layer of WebSphere Process Server, which addresses the transformation challenges for integrating applications. Figure 2-7 shows the supporting services layer of WebSphere Process Server.

![Supporting services layer of WebSphere Process Server](image-url)
Mediation flows

Mediation flows are part of the WebSphere Enterprise Service Bus intercept. They modify messages that are passed between existing services and clients that use those services. A mediation flow mediates or intervenes to provide functions, such as message logging, data transformation, and routing.

Interface maps

Interface maps are a channel through which data is exchanged between components that are running on WebSphere Process Server. Interface maps resolve the differences between components that have different interfaces and enable them to communicate.

Business object maps

Business object maps support mapping between the source and the target business objects. Business object maps are supporting services components, which are within WebSphere Process Server that transform one business object to another. The mapping can be a simple copy of source attribute value to a destination attribute or involve complex transformations to assign a value to the attribute in the destination business object based on a value in the attribute of a source business object.

Relationships

Relationships correlate semantically equivalent business objects that are represented in different formats. Relationships are supporting services components, which are within WebSphere Process Server applications, that establish an association between data from two or more data types.

Selectors

Selectors are the supporting services components that provide for the flexibility in processing service components during runtime. A selector takes one invocation and allows for different targets to be called based on the selection criteria during runtime. Selectors provide the flexibility in business processes to return different results (calling different components) than as originally designed without having to change the design of the business process.

Service components

All integration artifacts that are running on WebSphere Process Server (for example, business processes, business rules, human tasks, and business state machines) are represented as components with well-defined interfaces. Within the SCA, a service component defines a service implementation.

All service components, also called SCA components, have an interface and can be wired together to form a module that is deployed to WebSphere Process Server. By using the SCA components for assembly, you can change any part of
the application without changing the other parts by simply changing the corresponding SCA component. SCA components can also interact with other applications on EIS by using WebSphere Adapters and WebSphere Business Integration Adapters.

Figure 2-8 shows the service component layer of WebSphere Process Server.

![Service component layer of WebSphere Process Server](image)

**Business processes**

Business processes are service components that provide the primary means through which enterprise services are integrated. A business process is a procedure that an organization uses to achieve a business goal. A business process consists of a series of tasks (also known as activities) and the order in which the tasks are executed.

The business process component implements a fully supported Web services BPEL engine. WebSphere Process Server includes a business process choreography engine on top of the WebSphere Application Server. The engine fully supports the BPEL specifications and IBM extensions to BPEL, such as the human tasks and timeouts for activities.

**Human tasks**

Human tasks are stand-alone service components that you can use to assign work to employees or to invoke other services. You can use human tasks to implement staff activities in business processes that require human interactions, such as approvals or manual exception handling. WebSphere Process Server supports dynamic staff assignment and multi-level escalations for human tasks.
**Business state machines**

The *business state machine* component in WebSphere Process Server provides a way to define a business process based on states and events rather than a sequential business process model.

**Business rules**

*Business rules* are service components that declare policy or conditions that must be satisfied within a business. Business rules make business processes more flexible. By using business rules within a business process, applications can respond quickly to changing business needs.

### 2.2.2 Features of WebSphere Process Server

WebSphere Process Server is a combination of the best capabilities of WebSphere InterChange Server, WebSphere MQ Workflow, and WebSphere Business Integration Server Foundation. It fully supports SOA and uses WebSphere Application Server as the underlying platform. The benefits of WebSphere Process Server include improved flexibility and scalability, a streamlined development environment, and a simplified operational environment.

WebSphere Process Server includes the following features:

- Business processes built on SOA that use the Web services BPEL, which is an industry standard specification
- SCA, which is a component-based framework that separates business logic from integration logic:
  - By using SCA, developers can concentrate on the business logic of the component without worrying about the integration logic.
  - All SCA components have an interface to interact with other SCA components.
- Modeling and monitoring of business processes by using WebSphere Modeler and WebSphere Monitor:
  - With WebSphere Modeler, business analysts can develop the process models along with business measures.
  - WebSphere Monitor monitors the events in the running processes.
- A deployment environment for applications that are developed in WebSphere Integration Developer
  - WebSphere Integration Developer is a simplified development tool with visual editors for component development, assembly, integrated testing, and deployment.
Support of human tasks, role-based task assignments, and multilevel escalations:
- You can assign tasks that are within a process to humans for manual completion.
- Task assignment to staff can be based on the role of the staff.
- You can escalate tasks that require multilevel approvals.

Business rules, business state machines, and selectors to dynamically choose interfaces based on business scenarios
- By using the embedded business rules engine, you can define the business rules in the process and design the process based on the outcome of the execution of business rules.
- By using the business rules engine, you can dynamically change the outcome business process without changing the actual business process.
- By using the business state machine, you can model the business process based on states (ad hoc) rather than sequential order.

Broad reach in integration with the support for WebSphere Adapters and WebSphere Business Integration Adapters:
- A number of WebSphere and WebSphere Business Integration Adapters are available off the shelf to connect to various EISs

Ability to manage and monitor events by using CEI

Service governance with dynamic runtime lookup and invocation of services by using WebSphere Service Registry and Repository

Ability to change business processes with minimal programming skills without redeploying the application

Dynamic endpoint administration, which allows administrators to react to changing business needs by enabling reconfiguration of service endpoints

Integrates with Tivoli monitoring functionality, including Performance Monitoring Infrastructure (PMI), Application Response Measurement (ARM), and support for the IBM Tivoli Composite Application Manager suite of products

Integrated enterprise service bus (ESB) with prebuilt mediations, which increases business flexibility and responsiveness and enhances usability, saving on time and development costs

A design that is built over WebSphere Application Server Network Deployment, leveraging the transaction, security, clustering, and workload management capabilities of WebSphere Application Server
Features introduced in WebSphere Process Server V6.2

This version includes several new features to enhance business flexibility and enable faster and more effective deployment of applications.

WebSphere Process Server, version 6.2 includes the following new features:

- Real-time access to critical process information and the ability to interact with processes to influence runtime process execution in response to changing business conditions:
  - Extends the capabilities for the business user that were introduced in WebSphere Process Server Version 6.1.2, using the common Business Space powered by WebSphere user interface, for worklist and task management with new features, such as the ability to view the related business process or task history information from the human workflow diagram and the ability for business users to create, view, modify, verify status, and cancel subtasks from within their business space.
  - Enables business users with enhanced flexibility and control over runtime processes through the ability to characterize a collection of process activities and their associated data as a defined unit, which enables them to modify the flow of steps within those in-flight process instances by skipping activities, jumping forward and backward between activities, and adding additional activities from within their Business Space.
  - Delivers new Business Calendar Manager widgets in Business Space that enable business users to add, update, and delete timetable and time interval information to reflect available time changes based on ongoing business operations.
  - Enables directed deployment from WebSphere Business Modeler into WebSphere Process Server so that you can directly deploy models to the runtime environment.
  - Simplifies the identification of failed flows by using a graphical tree view that correlates log statements and errors that appear in the system out log, enabling faster problem determination.

- Powerful enhancements that simplify the process of application deployment, grant additional control over the artifacts in the deployment environment, and improve user productivity for ongoing application operation and administration:
  - Supports direct deployment of executable process models from WebSphere Business Modeler.
  - Introduces enhancements that enable module versioning and module-aware service versioning (SCA bindings only).
– Enables configuration for role-based access to timetable information in the runtime environment with Business Space widgets.

– Supports the population of relationship tables with instance data through a SQL script or Java program, which eliminates the need for manual data entry.

– Enables easier installation and configuration of WebSphere Process Server, which is key to a successful deployment of your SOA infrastructure. New installation and configuration enhancements that increase usability and speed time-to-value include a full installation of WebSphere Process Server Version 6.2, including WebSphere Application Server Network Deployment and the Web Services Feature Pack for WebSphere Application Server, with the creation of a profile that includes the Web Services Feature Pack augmentation; improved installation error determination after an installation failure or partial success; an installation verification tool that validates that the installation produced a successful server configuration; Installation Factory Integrated Install Package (IIP) support for creating custom installation packages; and scripting capabilities for configuration of production environments.

– Improves installation experience on z/OS with enhancements that include a reduction in the number of authentication aliases generated for resources within WebSphere Process Server for z/OS, the ability for customers to use the zPMT configuration tool to create augment response files for use in augmenting their WebSphere Process Server for z/OS installation, and the generation of a more consumable Data Definition Language (DDL).

– Simplifies problem determination with consistent fault support for bindings (including tools in WebSphere Integration Developer) and unified failed event management for all components (including mediations).

– Empowers the administrator with widgets in Business Space for monitoring the health of the system. They also provide additional information from the administrative console including SCA module details, Enterprise JavaBeans™ (EJB) import binding information, and contextual links throughout the panels. They also provide more control when installing an SCA module through the administrative console or equivalent scripting and optionally deploying it to a target server or cluster.

– Simplifies the management of the Service Integration Bus with a new browser view that is tailored to the user who explores existing buses in support of ongoing operations.

– Delivers simplified user experience with the reporting capabilities of Business Process Choreographer Observer now moved to Business Process Choreographer Explorer to use these reporting capabilities when administrating business processes and human tasks. In addition, includes
new capabilities around custom view definition with time constraints relative to when the view is used and where the selected actions to be displayed match the anticipated scenarios.

- Enhancements that facilitate faster and more effective development, testing, deployment, and execution of business process solutions:
  - Delivers enhancements to more easily support additional use cases, including true support for arrays and discovery enablement for JAX-WS and JAXB2-based Java services.
  - Supports data handlers that are ready to use as-is, for all bindings, as configured in WebSphere Integration Developer.
  - Delivers capabilities to build more flexible and intelligent process solutions through the ability to access business context information and programmatically propagate that context for all binding types. A new context service is responsible for propagating the context (including the protocol headers, such as the JMS header and the user context (such as account ID), along a Service Component Architecture (SCA) invocation path. The context service offers a set of APIs and configurable settings.
  - Delivers process model extensions for Business Process Modeling Notation support, including processes defined in WebSphere Business Modeler and WebSphere Integration Developer that include generalized flows (previously called “Cyclic Flows”) in which the navigation logic was set using the visual tools and the behavior specified for incoming and outgoing links.
  - Enables the definition of an exit condition on each process step, which specifies the criteria that must be met for an activity to be automatically skipped when reached by navigation, and can be checked on entry, on exit, or on both entry and exit of process step execution.
  - Improves publish response time and messaging engine start-up time.

- Extended reach of process solutions:
  - Delivers new policy-driven connectivity for administrators to configure service mediations through policies.
  - Enhances Web Services standards support.
  - Enhances service mediation capabilities.

- Enhancements designed to ease the effort of migrating from WebSphere Business Integration heritage server solutions:
  - WebSphere Business Integration Server Foundation solutions: Improves integration developer productivity by enabling the import of entire WebSphere Studio Application Developer Integration Edition workspaces
into WebSphere Integration Developer, and by aiding in the migration of service interfaces to custom Java code.

- **WebSphere InterChange Server solutions:** Improves time to value when migrating solutions by enabling the use of migrated maps with WebSphere Version 6 Adapters on WebSphere Process Server, generating native SCA bindings (MQ, JMS, HTTP, and EJB) for use with migrated maps, supporting running text-based heritage data handlers on WebSphere Process Server, and improving the runtime performance of migrated content.

- **WebSphere Business Integration Server Express and WebSphere Business Integration Server Express Plus solutions:** Supports the migration of WBI-SX artifacts to new value-add solutions assembled with WebSphere Integration Developer for deployment on WebSphere Process Server, which provides a growth path to an enterprise Business Process Management solution that enables significant value-add capabilities, platform support, and increased workloads.

- **WebSphere MQ Workflow solutions:** Enhances qualities of service with improved human task performance for WebSphere Process Server workflow solutions and new migration utility options for generating process models optimized for visual recognition and runtime performance.

**Platform alignment and currency:**

- Supports WebSphere Application Server Version 6.1.

- Supports IBM z/OS and z/OS.e 1.9, or later, including zFS, enabling WebSphere Process Server to run on the latest release of z/OS to use native z/OS facilities.

- Supports IBM IMS Version 10, enabling WebSphere Process Server solutions to use the enhanced database and transaction processing capabilities of the latest IMS release.

- Supports Microsoft Windows Vista® as a runtime platform in nonproduction environments, enabling WebSphere Process Server to run on the latest version of the Microsoft server operating system in development and test scenarios.

- Complies with the security settings as defined by the Federal Desktop Core Configuration (FDCC) for the U.S. federal government.
Migration concepts

In this chapter, we demonstrate the architecture differences between IBM WebSphere Business Integration Server Foundation V5.1 and WebSphere Process Server V6.2. We also describe the benefits and the business motivation for the migration.

In this chapter, the topics that we cover are:

- “Migration concepts overview” on page 30
- “Features mapping and migration approach” on page 31
- “Benefits of migrating to WebSphere Process Server” on page 37
3.1 Migration concepts overview

WebSphere Business Integration Server Foundation and WebSphere Process Server are architecturally different products that address the business process management (BPM) space. Because of this fundamental difference, the migration of WebSphere Business Integration Server Foundation to WebSphere Process Server is semi-automatic. Manual intervention might be needed to migrate some features and to optimize the processes for WebSphere Process Server.

Important: Some of the features in WebSphere Business Integration Server Foundation were enhanced, changed, or moved to equivalent features in WebSphere Process Server. You must understand those changes and differences when you are considering for migration.

Because the runtime architecture for the two products is different, it is not possible to do a runtime migration of the in-flight processes. Adopt a phased migration approach. You must also migrate the source libraries from WebSphere Business Integration Server Foundation to WebSphere Integration Developer and test them before you deploy WebSphere Process Server to production.

Follow the migration process and best practices to make the migration smooth.

The migration process entails different steps:

1. Make an assessment regarding the need for the migration, risks, different options for migration, and the current environment.

2. Have the subject matter expert prepare and analyze a plan, keeping the current environment in mind. Make any changes to the WebSphere Business Integration Server Foundation libraries that are necessary to make a smooth migration.

3. Migrate the environment in a phased manner by following the migration plan that was prepared in the previous point. The migration is not fully automatic. You must make the necessary changes for the new WebSphere Process Server environment.

4. Regressively test the migrated environment to preserve all of the functionality and the business logic.

5. After the testing is complete with satisfactory results, deploy the migrated components into production.
3.2 Features mapping and migration approach

WebSphere Process Server V6.2 is a service-oriented architecture (SOA) integration platform that is built on a uniform invocation programming model and a uniform data representation model. WebSphere Process Server includes an enhanced Common Event Infrastructure (CEI) that generates events for monitoring applications. WebSphere Process Server V6.2 provides many key enhancements around service integration, quality of service, performance, and ease of use.

3.2.1 General feature comparison

The programming models of the WebSphere Business Integration Server Foundation and the WebSphere Process Server have the following major differences:

- The Web Services Invocation Framework (WSIF)-based data model in WebSphere Business Integration Server Foundation is now Service Data Objects (SDO) in WebSphere Process Server.
- The WSIF-based invocation model in WebSphere Business Integration Server Foundation is now Service Component Architecture (SCA) in
WebSphere Process Server. Web services clients are now pure Java API for XML-based Remote Procedure Call (JAX-RPC) clients rather than using WSIF.

WebSphere Process Server also contains several new features that do not correspond to WebSphere Business Integration Server Foundation features. Such features include business rules, business state machines, interface maps, relationships, and WebSphere Enterprise Service Bus mediations.

- WebSphere Business Integration Server Foundation has Business Rule Beans with limited functionality. However, WebSphere Process Server business rules support is better and more complete.

Table 3-1 shows the features comparison between WebSphere Process Server V6.2 and WebSphere Business Integration Server Foundation V5.1.

<table>
<thead>
<tr>
<th>Table 3-1 Features comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WebSphere Business Integration Server Foundation V5.1</strong></td>
</tr>
<tr>
<td>J2SE™ Version</td>
</tr>
<tr>
<td>J2EE Version</td>
</tr>
<tr>
<td>Data model</td>
</tr>
<tr>
<td>Invocation model</td>
</tr>
<tr>
<td>Business object maps</td>
</tr>
<tr>
<td>Interface maps</td>
</tr>
<tr>
<td>WebSphere Enterprise Service Bus Mediations</td>
</tr>
<tr>
<td>Selectors</td>
</tr>
<tr>
<td>Relationships</td>
</tr>
<tr>
<td>Business rules</td>
</tr>
<tr>
<td>Business processes</td>
</tr>
<tr>
<td>Business state machines</td>
</tr>
</tbody>
</table>
WebSphere Process Server V6.2 is not an extension for WebSphere Business Integration Server Foundation V5.1; instead, it depends on a new infrastructure. Therefore, direct feature mapping is not possible.

### Programming model extensions
WebSphere Business Integration Server Foundation provides several valuable extensions to the J2EE specification. These extensions are delivered in many different forms, including services, APIs, and tooling extensions.

Most of these programming model extensions (PMEs) are made available in WebSphere Application Server Version 6. Table 3-2 contains an overview of the WebSphere Business Integration Server Foundation PMEs and their availability in the version 6 product stack.

<table>
<thead>
<tr>
<th>Extension in WebSphere Business Integration Server Foundation</th>
<th>Available in V6 product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity session</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Application profiling</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Asynchronous beans</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Business Rule Beans</td>
<td>WebSphere Process Server (deprecated)</td>
</tr>
<tr>
<td>Container Managed Persistence over Anything (CMP/A)</td>
<td>WebSphere Process Server (deprecated)</td>
</tr>
<tr>
<td>Dynamic query</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Extended messaging (container-managed messaging (CMM))</td>
<td>WebSphere Process Server (deprecated)</td>
</tr>
<tr>
<td>Extension in WebSphere Business Integration Server Foundation</td>
<td>Available in V6 product</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Globalization (I18N)</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Last participant support</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Object pools</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Scheduler</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Startup beans</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Work areas</td>
<td>WebSphere Application Server</td>
</tr>
</tbody>
</table>

For more detailed information about migrating applications using PMEs that are now available in WebSphere Application Server, visit the WebSphere Application Server Information Center at the following address and access *Programming model extension migration*:


For information about migrating the PMEs, which are part of WebSphere Process Server, see the following sections in this book:

- To migrate Business Rule Beans, see 8.6, “Additional migration considerations” on page 185 section “Business Rule Beans” on page 189.
- To migrate Container Managed Persistence over Anything (CMP/A), see 8.6, “Additional migration considerations” on page 185 section “Container Manager Persistence over Anything (CMP/A) support” on page 190.
- To migrate extended messaging (CMM), see “Additional migration considerations” section “Extended messaging service” on page 190.
3.2.2 Source artifacts, bindings, and the migration approach

Use a general source artifacts migration approach when migrating between the WebSphere Business Integration Server Foundation and WebSphere Process Server. Table 3-3 and Table 3-4 on page 36 provide an overview of how we can map and migrate the WebSphere Business Integration Server Foundation bindings and artifacts into WebSphere Process Server.

Table 3-3 shows the binding mapping comparison between WebSphere Process Server V6.2 and WebSphere Business Integration Server Foundation V5.1 and the migration approach.

<table>
<thead>
<tr>
<th>WebSphere Business Integration Server Foundation V5.1 binding</th>
<th>WebSphere Process Server V6.2 construct</th>
<th>Migration approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJB</td>
<td>Export with SCA binding</td>
<td>The migration wizard creates an export with the SCA binding and wires it to the Business Process Execution Language (BPEL) component in the assembly editor.</td>
</tr>
<tr>
<td>JMS</td>
<td>Export with JMS binding</td>
<td>The migration wizard creates an export with the JMS binding and wires it to the BPEL component in the assembly editor.</td>
</tr>
<tr>
<td>IBM Web service with SOAP/HTTP</td>
<td>Export with Web service binding (SOAP/HTTP)</td>
<td>The migration wizard creates an export with the Web service binding that is wired to the BPEL component.</td>
</tr>
<tr>
<td>IBM Web service with SOAP/JMS</td>
<td>Export with Web service binding (SOAP/JMS)</td>
<td>The migration wizard creates an export with Web service binding that is wired to the BPEL component.</td>
</tr>
<tr>
<td>Apache Web service with SOAP/HTTP</td>
<td>Export with Web service binding (SOAP/HTTP)</td>
<td>The migration wizard creates an export with Web service binding that is wired to the BPEL component.</td>
</tr>
</tbody>
</table>

Table 3-4 on page 36 shows the artifacts mapping between WebSphere Process Server V6.2 and WebSphere Business Integration Server Foundation V5.1 and the migration approach.
**Table 3-4  Artifacts mapping**

<table>
<thead>
<tr>
<th>WebSphere Business Integration Server Foundation V5.1 artifact</th>
<th>WebSphere Process Server V6.2 construct</th>
<th>Migration approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL service</td>
<td>Process (BPEL) component.</td>
<td>Automatically handled by the migration wizard.</td>
</tr>
<tr>
<td>BPEL-to-BPEL invocation</td>
<td>Wire the connection in the assembly editor if the BPEL exists in the same project. Use import with SCA binding if the BPEL is in another project.</td>
<td>Automatically handled by the migration wizard.</td>
</tr>
<tr>
<td>EJB service</td>
<td>Import with the EJB binding.</td>
<td>Automatically handled by the migration wizard. The migration wizard creates an import with a Stateless EJB binding. A Java component maps the EJB to a WSDL interface and wires the two together. The wizard also wires the Java component to the BPEL component.</td>
</tr>
<tr>
<td>JMS</td>
<td>Import with the JMS binding.</td>
<td>Automatically handled by the migration wizard. The migration wizard creates an import with JMS binding.</td>
</tr>
<tr>
<td>Web service (SOAP/HTTP or SOAP/JMS)</td>
<td>Import with Web service binding</td>
<td>Automatically handled by the migration wizard. The migration wizard creates an import with the Web service binding and wires the import to the BPEL component.</td>
</tr>
<tr>
<td>Java service</td>
<td>Java component</td>
<td>Automatically handled by the migration wizard. The migration wizard creates a Java component and wires it to the BPEL component.</td>
</tr>
<tr>
<td>Transformer service</td>
<td>Data maps and interface maps</td>
<td>Manual through the Assembly Diagram. Use either the Process Server map functionality or create a Java component that calls the proxy that was generated in Server Foundation.</td>
</tr>
</tbody>
</table>

Or use the proxy that was generated in WebSphere Business Integration Server Foundation in a Java component in WebSphere Process Server.
3.3 Benefits of migrating to WebSphere Process Server

Migrating to WebSphere Process Server has the following benefits:

- Superior architecture that is built over SOA
  
  SOA is an IT architectural style that supports the transformation of business into a set of linked services, or repeatable business tasks, that can be accessed when needed over a network. These services can coalesce to accomplish a specific business task, enabling your business to quickly adapt to changing conditions and requirements.

- Powered by IBM SOA foundation product set
  
  IBM SOA Foundation is an integrated, open-standards-based set of software, best practices, and patterns that are designed to provide what you need to get started with your SOA. Developing reusable business processes on an SOA foundation has the following benefits:
  
  – Business and IT flexibility
  – Streamlined and efficient operations
  – Consistent workflows, business rules enforcement, and automation for regulatory compliance (Sarbanes Oxley, HIPAA)
  – Rapid response to emerging market opportunities, customers, and partners
  – Measurement of business processes that enable continuous improvement and optimization

  IBM built a comprehensive suite of products in the IBM SOA Foundation product set. These products span the entire BPM life cycle, enabling you to model, assemble, deploy, and manage your application and business processes.

- Converged SOA platform for process and application integration
  
  This single-integrated platform supports a vast array of features that are needed for business integration. The features include business process choreography, business state machines, business rules, enterprise service bus (ESB), and CEI.

- Support for open standards for defining business processes (BPEL), business objects (SDO), and the invocation model (SCA).

- Enhanced support for Web services.

- Enhanced support for messaging with the help of bindings support for WebSphere MQ and JMS.
- More efficient WebSphere Adapters, such as better memory management and manageability, compared to WebSphere Business Integration Adapters.
- Support for human intervention in the business process.
- Better platform for business integration.
- Simplified development environment by using WebSphere Integration Developer.
- Robust server platform that is based on WebSphere Application Server Network Deployment.
- Simplified administration using the administrative console.
- Supports dynamic runtime change of service endpoints.

For more benefits about WebSphere Process Server, see 2.2.2, “Features of WebSphere Process Server” on page 22.
Migration planning

In this chapter, we help the project management team to plan the migratory path. We begin by discussing what considerations must be taken into account. The output is a recommended approach that can be driven through three main project phases:

► **Assessment**
  
  This phase leads to high-level planning in accordance with the requirements for an assessment of the existing environment and requirements for gathering and analysis.

► **Preparation**
  
  This phase ensures that the design of the new environment is suitable and that the necessary level of skills and experience is reached to secure the project.

► **Implementation**
  
  This phase regroups and executes all of the necessary tasks that are needed to deliver the migration.

We provide relevant information in the following sections:

► “Assessment phase” on page 40
► “Preparation phase” on page 61
► “Implementation phase” on page 63
4.1 Assessment phase

In this section, we explain why the migration must be driven through a dedicated project and consider the key items to take into account. The migration project must begin with an assessment phase. Generally speaking, because the intent is to move from a known source environment (WebSphere Business Integration Server Foundation) to a new target environment (WebSphere Process Server), we recommend the following path:

1. Plan the project, including the phases, staffing, dependencies, and budget. See 4.1.1, “Project planning” on page 42.

2. Assess the current environment of the existing WebSphere Business Integration Server Foundation environment, that is, define where you stand. See 4.1.2, “Current environment assessment” on page 45.

3. Gather the requirements and analyze the impact. Define where you are going. See 4.1.3, “Requirements gathering and impact analysis” on page 46.

Figure 4-1 on page 41 helps you to visualize this approach.
Figure 4-1  Migration approach
4.1.1 Project planning

WebSphere Business Integration Server Foundation projects deal with an inherent complexity that is related to service-oriented architecture (SOA) solutions.

While there are several different approaches to migration, we recommend the best practice approach that we outline in this section. These guidelines lead to a project approach that is decomposed into three main phases as described in 4.1, “Assessment phase” on page 40, 4.2, “Preparation phase” on page 61, and 4.3, “Implementation phase” on page 63.

Project planning is where the project manager outlines the overall project plan. This person breaks down the different parts of the project into the necessary phases and ensures that the necessary human resources are engaged for the different phases. Budgeting the project and outlining the specific business requirements also fall into this phase.

Because the migration involves more effort than a simple software upgrade, the effort can be analyzed as:

- Migrating existing business processes from WebSphere Business Integration Server Foundation to a new platform staying current with ongoing support
- Using the migration as an opportunity to add value to the current business by using the functionalities and features brought about by WebSphere Process Server

For these reasons, we recommend that you adopt a project approach for the migration. With such an approach, you can assess, optimize, budget, prepare, and deliver the migration properly and get as much return on investment (ROI) as possible.

Because the intent is to move from a known source environment (WebSphere Business Integration Server Foundation) to a new target environment (WebSphere Process Server), we recommend that you know the target before you prepare for the migration implementation. A project approach is recommended for this purpose, as explained in the following points:

- Start using WebSphere Process Server for new developments as soon as possible. Because you need comprehensive experience with the new platform, from the development system to the production system, execute a project for this purpose.
- Choose a new project to begin acquiring and building your skills on the SOA or business process management (BPM) platform. A pilot project is generally the best compromise to reach this goal with reduced risks.
Start building WebSphere Process Server skills now, along with the WebSphere Application Server, if needed. By running a pilot project, with a well defined and tailored technical scope, you can select points of interest and build relevant key skills.

Avoid starting the WebSphere Process Server experience with a pure migration project. Use a pilot project to identify new and interesting capabilities that will bring value to the solution. A first pilot project also helps you to get used to the new platform before you focus on the migration implementation.

Go through a full life cycle from development to go live with WebSphere Process Server. By doing so, you get a proven production infrastructure before migrating. Such an approach smooths the learning curve and optimizes the migration effort.

In summary, we strongly advise that you invest in the assessment and preparation phases because they help secure the migration implementation phase.

**Optimizing your migration**

Use the migration as an opportunity and not as a constraint:

- Identify your business objective and value of the modified solution:
  - Enhance the solution with capabilities that were previously not possible.
  - Leverage the new functions.

- Generally, avoid a “big bang” approach:
  - Unless your migration project is small, it is generally too risky to migrate everything at one time.
  - Migrate by viable technical stages, for example, migrate the server first and then business objects using migration tooling between WebSphere Business Integration Server Foundation and WebSphere Process Server.
  - Migrate by functional stages, for example, first do a pilot project, and then start migrating non-critical business objects. Next, progressively migrate more critical business objects.
  - A parallel infrastructure cost can be compensated by new functionalities that bring added value.

It is generally cost effective to combine migration with the next application or infrastructure change when such a change is required.

**Migration project roles and responsibilities**

In Table 4-1, we describe the recommended roles for the migration project.
Table 4-1  Project roles in detail

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project manager</td>
<td>Oversees project management, budget, and staffing coordination.</td>
</tr>
<tr>
<td>Analyst</td>
<td>Gathers and analyzes requirements on the new environment, helps to design test scenarios, and verifies functional test results. Preferably has process modeling experience.</td>
</tr>
<tr>
<td>WebSphere Business Integration Server Foundation product specialists (team)</td>
<td>Has hands-on experience with WebSphere Business Integration Server Foundation, WebSphere MQ (if applicable), and databases.</td>
</tr>
<tr>
<td>WebSphere Process Server product specialists (team)</td>
<td>Has hands-on experience with WebSphere Process Server, service integration bus, and databases.</td>
</tr>
<tr>
<td>Development team</td>
<td>Performs the migration of the applications, which includes running migration tools and developing customizations, fixes, and others.</td>
</tr>
<tr>
<td>Testing team</td>
<td>Designs, prepares, and performs tests.</td>
</tr>
<tr>
<td>Maintenance team</td>
<td>Supports the new environment and quickly develops and applies fixes. The team is a combination of developers, testers, product specialists, and administrators.</td>
</tr>
</tbody>
</table>

**Project manager:** The project manager is used during the entire project life cycle. We recommend that you staff the project with experienced WebSphere Process Server specialists. Developers and specialists must be experienced with WebSphere Process Server projects already. Do not perform a migration project as the first project on WebSphere Process Server because changes to tooling and the runtime environment require additional skills.

Table 4-2 on page 44 describes the roles that are required in each project phase and the expected percent of utilization of the role during the phase.

Table 4-2  Project phases mapped to project roles

<table>
<thead>
<tr>
<th>Phase</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment</td>
<td></td>
</tr>
<tr>
<td>Migration project planning</td>
<td>Project manager 60%, and architect 40%</td>
</tr>
</tbody>
</table>
### 4.1.2 Current environment assessment

In this section, we describe the input that is related to the current environment.

#### Current IT environment context

Before covering the WebSphere Business Integration Server Foundation project scope, collect input that is related to the global IT environment. Here are some examples of questions that are related to this task:

- What are the identified dependencies between the WebSphere Business Integration Server Foundation project and other domains?
- What is the project roadmap for the connected applications?
- What is the SOA strategy and maturity?
- Is there an ongoing task about providing reusable services?
- What governance model is currently in place?
- How are the projects and the common assets funded?
Current WebSphere Business Integration Server Foundation project context
To assess the current WebSphere Business Integration Server Foundation solution, use the following examples of the input to collect:

- Business or functional input
  Functional description and business relevance of interfaces

- Architectural input
  Architectural pattern description of interfaces

- Technical input:
  - Service level agreement (SLA), which is likely to be the same for the new environment
  - Availability and scalability of the current solution
  - Performance of throughput, utilization, response times, and so on
  - Security related to security zones, users, roles and groups, and access rights

4.1.3 Requirements gathering and impact analysis

Gather requirements from IT and business departments and consolidate them with current environment assessment results. The requirements include changes in SLA, functionality changes that can be connected with the migration, technological changes, and others.

It is important to assess whether it is intended to migrate “as is” or to add value to the current solution. It is critical to determine if the migration is driven by constraint, opportunity, or both because they influence the ROI.

Impact analysis
Each project has its own set of unique requirements and constraints. Therefore, perform an analysis of the impact that this project will have on the business, infrastructure, and technical departments.

In this section, we provide a list to use as input to build up your own impact assessment. We discuss the following concepts:

- Organizational concepts and changes
- Project delivery
- Methodology
- Architecture and design
- Development
Testing
Deployment
Run time and security

**SOA and IT governance**

SOA governance is an extension of IT governance that focuses on the life cycle of services and composite applications in an organization's SOA. For an introduction to SOA governance, see *IBM SOA Governance Life Cycle* at the following Web address:


WebSphere Process Server in conjunction with other products, such as WebSphere Service Registry and Repository, is an enabler to implement such organizational concepts.

**BPM enabled by SOA, business, and IT alignment**

To understand how SOA and BPM can work together to facilitate business and IT alignment, see the *BPM and SOA: Better Together* paper at the following Web address:


WebSphere Process Server, in conjunction with other products, such as Business Modeler and Business Monitor, is also an enabler to implement such organizational changes.

**Center of Excellence**

For a high scale SOA, it is common to have a Center of Excellence (CoE) in place. The migration project can either leverage or be an opportunity to initiate such an organizational structure.

A CoE is a proven organizational model for driving architecture and design capabilities, which in turn, enables successful delivery of business strategy. A CoE has the following goals:

- Socialize architecture by communicating framework, best practices, assets, patterns, and methods.
- Conduct independent architecture reviews.
- Provide architecture vitality by continuously assessing and refining architecture framework, and support assets based on internal and external influences.
- Provide project support with direct project assistance to drive architecture.
- Provide skills transfer and early proof of concepts to identify skills gaps, create development roadmap, and drive the use of new technologies.
Promote asset adoption to manage component, pattern, and data re-use processes to reduce project risk and to accelerate delivery.

Provide a best practice policy and procedures to enable expert resources to accelerate the delivery of critical architecture practices.

Support production by enabling infrastructure teams to execute on build, deploy, tuning, and metrics reporting.

Project delivery
In the following sections, we present information that is related to project delivery and that might be impacted by the migration.

Project costs and funding
As WebSphere Process Server brings updated SOA concepts, its deployment during the migration project can impact the usual project costs and funding model:

- Hardware
  WebSphere Process Server is supported on new platforms; therefore, it can use the new hardware. The migration project generally implies a transition phase that can raise license and hardware costs due to a parallel run.

- Data management
  Because the internal data model is richer in WebSphere Process Server than in WebSphere Business Integration Server Foundation, data management can be impacted by archiving and warehousing costs to maintain the long running instances or business processes for an instance.

- License procurement and maintenance
  We recommend that you investigate what you can do with Extended Entitlement, which is described at the following Web address:

- Funding
  Because WebSphere Process Server participates in the SOA vision, it brings new concepts around project funding. A typical example is the way that SOA intends to identify and reference services to ensure that they are shared by different entities or projects in an optimal way:
  - Capacity to reuse an existing service
  - Capacity for the service to scale up properly with growing consumers
This act of sharing can impact the current funding model by raising the following types of questions:

– Who pays to develop and maintain the common services?
– How does the consumer pay to use them?

The migration project

You can raise the ROI of the project as follows, for example:

– Mitigate the parallel infrastructure cost through the enablement of interoperability means between WebSphere Business Integration Server Foundation and WebSphere Process Server, for example, WebSphere Process Server can reuse existing WebSphere Business Integration Server Foundation business objects after they are migrated, which spares the corresponding new development costs. Another example is that WebSphere Process Server and WebSphere Business Integration Server Foundation can use the same existing relationships, which spares some database-related costs.

– Leverage extended license entitlement for migration. Installed base licensees with an active subscription can activate entitlement to the WebSphere Process Server and receive support for both old and new products during migration.

– Identify new functions that add value to the migrated solution.

Project timeline and product roadmap

Among the factors that impact the project timeline is the product roadmap. During the transition phase, going from WebSphere Business Integration Server Foundation to WebSphere Process Server, the migration project needs to manage both product roadmaps.

For the WebSphere Business Integration Server Foundation, the main factor is the date of end of support:

– The end of support has not yet been announced for WebSphere Business Integration Server Foundation. IBM intends to support these products until at least April 2010. To track this end-of-support date, check the following Web address periodically for updates:


– To the extent that these products include third-party prerequisite software, such as WebSphere Business Integration Adapters for enterprise applications, continued service for such software during this period can be determined on a case-by-case basis and in the sole discretion of IBM.

For WebSphere Process Server, the main factor is the way that the new releases of the product fill in possible functional gaps, easing technical
migration and adding new functionalities, providing business differentiators, for example.

**Service level agreement**

An SLA is a formally negotiated agreement between two parties. It is a contract that exists between customers and their service provider or between service providers. It records the common understanding about services, priorities, responsibilities, guarantee, and collectively the level of service, for example, it may specify the levels of availability, serviceability, performance, operation, or other attributes of the service, such as billing and penalties in the case of violation of the SLA.

An SLA might be made more complex with your new SOA solution because WebSphere Process Server offers a richer set of integration patterns. Therefore the way that SLAs are defined might be impacted by the migration.

**Methodology**

The word *methodology* can have different meanings depending on the context. Here we use it in a generic manner. Therefore, the examples should be self explanatory.

In this section, we discuss how various methodologies can be impacted by or related to the migration project. There are reasons that can lead to define new methodologies and that can lead to the modification and enrichment of current methodologies.

The following examples of methodologies might be needed to support the migration implementation:

- A proven methodology can be used to structure the detailed migration study. Typically based on best practices and experience collected from previous projects, this kind of methodology generally comes from a consulting entity. Sometimes it is required because WebSphere Process Server is new to the project and external help is required to complete such items as a gap analysis of the new product functionalities.

- A methodology for regression testing might have to be defined to secure this phase.

- From a system life cycle perspective, a *Transition for Operations* methodology might have to be defined to secure skills building for this team.
The following considerations, among others, are related to project life cycle management:

- From the general project life cycle perspective, the main project phases are the same as when implementing an SOA or any IT infrastructure or development project.

- If you look for well-proven methodologies and tools related to SOA project life cycle management, you might investigate service-oriented modeling and architecture (SOMA) and Rational® Unified Process (RUP) for SOA. The IBM SOMA methodology helps address SOA implementation through dedicated phases, such as services identification, specification, realization, and so on. It uses a holistic approach, combining top-down and bottom-up analysis.

**Architecture and design**

Design patterns provide a way to describe the current WebSphere Business Integration Server Foundation solution and the target WebSphere Process Server solution from the architecture design perspective. Indeed, such a common “architecture language” is useful to categorize the differences in both SOA solutions and to define a migration path between the product specific implementations.

**Development**

In this section, we discuss points that are related to the development phase of the usual project life cycle. The WebSphere Business Integration Server Foundation development team is impacted by the migration.

The WebSphere Process Server development tooling (WebSphere Integration Developer) is new. It is completely independent from the WebSphere Business Integration Server Foundation development tooling (WebSphere Studio Application Developer Integration Edition V5.1 toolkit).

WebSphere Integration Developer is the development environment for building integrated business applications targeted for WebSphere Process Server. SOA support in WebSphere Process Server is based on a new programming model referred to as the Service Component Architecture (SCA). One of the primary purposes of WebSphere Integration Developer is to provide the appropriate tools to easily build and test SCA-based applications targeted for WebSphere Process Server. Both SCA and the tools support for SCA are aimed at helping developers decouple business logic from implementation details.

WebSphere Integration Developer is built on the Rational Software Development Platform. Rational Software Development Platform is itself based on Eclipse 3 technology. Each IBM product that is built on this platform can coexist and share plug-ins with other Rational Software Development Platform-based products.
Supported platforms for build time
Investigate precisely the supported platforms for the build time (WebSphere Integration Developer) in terms of hardware and operating system. You can find such information in the information center and support pages of the product at the following address:

http://www-1.ibm.com/support/docview.wss?rs=2308&uid=swg27010621

Coexistence of development tools: The main part of the standard migration tooling is in WebSphere Integration Developer. During the transition phase, you might have to use the WebSphere Business Integration Server Foundation and WebSphere Process Server development tools in parallel. Therefore, the feasibility of their coexistence on a common platform must be also verified, which relates to the compatibility of products to run on a single machine, the required memory, and so on.

Software upgrade process for build time
Fix packs are provided regularly to fix bugs and provide enhancements through minor product upgrades. Because WebSphere Integration Developer is different from the WebSphere Studio Application Developer Integration Edition, we recommend that you investigate the supported software upgrade process for WebSphere Integration Developer. You can find such information on the support pages of the product at the following address:


New development concepts and skills building
WebSphere Process Server brings a new programming model based on such concepts as SCA, Business Process Execution Language (BPEL), mediation flow component, and business rules. Developing for WebSphere Process Server in the most efficient way also implies that you must investigate more advanced concepts, such as the following examples:

- Versioning and dynamicity
- Error handling
- Quality of service

Therefore, the development team must build a new skills set to be operational on the WebSphere Process Server.

Team development
With WebSphere Integration Developer, you can develop applications in a team environment by sharing resources with a central repository. Through Eclipse, WebSphere Integration Developer provides a client for the Concurrent Versions...
You can also share a project by using Rational ClearCase® or other repositories.

Verify the supported repositories and their versions to anticipate any impact if a software configuration management (SCM) system is already used with the WebSphere Business Integration Server Foundation toolset.

**Further information**

For further information about the development tasks, see the product documentation and the WebSphere Integration Developer Information Center at the following Web address:


**Testing**

In this section, we discuss points that are related to the test phase of a typical project life cycle.

**Unit testing**

Unit testing does not raise any particular remark except the fact that developers must get used to the new unit test environment that is available with WebSphere Integration Developer.

**Regression testing**

Regression testing generally focuses on functional tests. However, in the case of migration, it is important to ensure that the nonfunctional requirements are met. We recommend that you verify nonfunctional items, such as:

- Quality of service (global resilience of the solution, flow control and throttling capabilities, and so on)
- The operations team's ability to monitor and administer the new platform

**Performance testing and benchmarking**

Because the WebSphere Business Integration Server Foundation and WebSphere Process Server runtimes and deployment configurations are different, run dedicated performance tests as early as possible to assess the capacity of the migrated solution.

**Deployment**

In this section, we discuss points that are related to the deployment phase of a typical project life cycle.
**Packaging**
Investigate the capabilities of the WebSphere Process Server platform regarding
the packaging process (using a GUI and automation tools). WebSphere
Business Integration Server Foundation and WebSphere Process Server are
different from this perspective and impact two areas:

- Generation of the software packages to be deployed
- Inter-environment adaptation procedures

**Component architecture of the solution**
WebSphere Process Server and WebSphere Business Integration Server
Foundation are similar with regard to the component architecture of the solution.
WebSphere Process Server, like WebSphere Business Integration Server
Foundation, is based on the WebSphere Application Server architecture. The
solution can also interact with other third-party products, such as the following
examples:

- A Lightweight Directory Access Protocol (LDAP) directory, such as Tivoli
  Directory Server
- A security system, such as Tivoli Access Manager
- Some monitoring tools from the Tivoli brand such as Tivoli Performance
  Viewer

Obviously, the migration can impact the third-party products that are used with
the solution. See the information related to third-party products in the
WebSphere Process Server Information Center and support pages of the
product at the following address:


**Topology for high availability and scalability**
The deployment topology of WebSphere Business Integration Server Foundation
is slightly different than WebSphere Process Server. Therefore, you must assess
your current environment from an availability and scalability point-of-view, and
select the corresponding deployment topology of WebSphere Process Server to
assure the desired level of business continuity. Overall availability of your
environment is closely related to costs, and your decision shall trade off between
costs and availability factors. See 4.4.1, “Availability” on page 66.

**Capacity planning**
Capacity planning is obviously dependent on the chosen deployment topology.
Because WebSphere Business Integration Server Foundation and WebSphere
Process Server run times are similar, it should be relatively straightforward to
determine capacity for existing processes that are being migrated to the new
environment. IBM can provide estimates based on Techline tools and benchmark
studies; however, you must run performance tests on site as early as possible to get realistic data.

**Product coexistence**
WebSphere Process Server and WebSphere Business Integration Server Foundation environments must coexist under the following circumstances:

- Both environments are up and running during piloting phase.
- Applications are migrated in several stages.
- A period of coexistence must be achieved to finish long running processes. Alternatively resubmit unresolved business processes on WebSphere Business Integration Server Foundation while starting and working on migrated processes in the new WebSphere Process Server environment.

Product coexistence has a potential effect on cost because it implies parallel infrastructure, which affects hardware consumption and license consumption. However, parallel infrastructure enables risk reduction and there are ways to reduce the costs. The extended entitlement can be a means to reduce costs that are related to licenses.

Therefore, we recommend that you investigate carefully these possibilities to optimize the transition phase.

**Run time and security**
In this section, we discuss notions related to run time, quality of service, and security.

**Installed run time**
The installed run time of WebSphere Business Integration Server Foundation does not migrate directly to a WebSphere Process Server run time:

- No standard migration path exists for instance data, such as unresolved business process events and long running business processes. The only supported way is to let them terminate in the WebSphere Business Integration Server Foundation environment.

- No standard migration path exists for system configuration. The only supported way is to redo the configuration steps for WebSphere Process Server, which is related, for example, to server tuning parameters, server security settings, and system monitoring.

- You must use the usual project life cycle, from development to production, with installation, setting, and tuning of the new product to set up your new production system with migrated applications.
Supported platforms for run time
Investigate the supported platforms for the WebSphere Process Server run time in terms of hardware and operating system. You can find that information in the information center and the support pages of the product at the following address:

Third-party products: The same concern applies to third-party products, such as databases and LDAP, that you can also find in the support pages. During the transition phase, the run times of WebSphere Business Integration Server Foundation and WebSphere Process Server might have to be used in parallel. If they do, verify the feasibility of their coexistence on a common platform (the compatibility to run on a single machine, required memory, and so on).

Software upgrade process for run time
Fix packs are provided regularly to fix bugs and provide enhancements through minor product upgrades. Because WebSphere Process Server is different from WebSphere Business Integration Server Foundation, investigate the supported software upgrade process for WebSphere Process Server. You can find this information in the WebSphere Process Server support pages at the following address:

Security
The security of WebSphere Business Integration Server Foundation and WebSphere Process Server is based on the security mechanisms for the base security of the application server. You can divide the security tasks into security of the server environment and security of the applications that are running on the server. For more details about security, see 4.4.2, “Security” on page 70.

4.1.4 Service decomposition
Service decomposition is critical to the success of migration. The goal of service decomposition is to break down the entire migration project by service isolations. An initial set of relatively separated services is created, which we can migrate and implement in separate stages.

These identified services are required by, and meaningful to, the business. To realize this business alignment, business process analysts (additional role) participate in service decomposition with software architects (main role). A one-to-one mapping from business to software services might be ideal, but it is
not always possible. Software engineering principles and other requirements can lead to other combinations.

After the business service is identified, perform service dependency analysis to re-organize the service into the groups, by functional or business requirement. By doing this, you can create separate deployable services and deploy them into the new environment in different stages.

Figure 4-2 illustrates the service decomposition process.

In Figure 4-2, you see four individual business services that handle the exemplary travel booking process:

- Payment processing
- Airline booking service
- Hotel booking service
- Car booking service
Each of these services is further categorized, per SOA standards, into different layers:

- Business logic layer
- Service components layer
- Implementation layer

Figure 4-2 on page 57 shows a high-level representation of the decomposed services into these layers. These layers help isolate the components so that they can be migrated in controlled stages.

**Service decomposition approaches**

Several approaches to service decomposition are possible, including top-down, meet-in-the-middle, and bottom up, as shown in Figure 4-4 on page 67. These approaches are guidelines. Therefore, choosing an appropriate service decomposition approach for your organization is entirely dependent on your existing business infrastructure.

**Domain decomposition**

Domain decomposition is the top-down approach. It starts with the business logic and corresponding service components that are associated with that logic. These
components are then grouped together in the implementation stage where they are packaged and deployed.

**Goal-service modeling**
Goal-service modeling is the meet-in-the-middle approach. It starts with mapping the business logic and the existing service implementation to the relevant service components.

**Existing service asset analysis**
Existing service asset analysis is the bottom-up approach. It involves examining deployed service implementations, finding dependencies and relationships with the service components, and mapping these back to the associated business logic.

### 4.1.5 Deliverables

In this section, we provide a list of possible outputs and deliverables to expect from the assessment phase. They are neither exhaustive nor mandatory.

**Requirements list**
Add business and IT requirements to the list that summarizes the requirements. A key point that is related to ROI is to define whether the requirements are driven by constraints, such as an end-of-support date, or opportunity, such as adding business value through new product capabilities.

**Main options to deliver the requirements**
At this stage, the main options to fulfill the requirements should be defined. Also, some decisions cannot be made yet because of a lack of information or experience.

**Return on investment study**
The assessment phase can include an ROI study.

**Risk assessment and mitigation plan**
The risk assessment should provide a mitigation plan for each identified risk. The preparation phase can be used to mitigate or even eliminate some risks. Table 4-3 on page 60 outlines the basic considerations by defining the following criteria:

- A question or a concern in relation to the migration project
- The corresponding risk area
- A proposed mitigation plan
### Table 4-3  Basic migration project considerations

<table>
<thead>
<tr>
<th>Question or concern</th>
<th>Risk area</th>
<th>Mitigation plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the capacity of the development team to deliver migrated components optimized for WebSphere Process Server run time?</td>
<td>Development skills</td>
<td>▶ Build WebSphere Process Server and WebSphere Integration Developer skills with training.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Get help from the CoE or consulting for best practices.</td>
</tr>
<tr>
<td>Is the operations team ready to administer the new WebSphere Process Server solution?</td>
<td>Operations team skills and tools</td>
<td>▶ Build WebSphere Process Server skills with training.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Run a pilot project before the migration project to create and test new administrative tools and procedures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Get help from the CoE or consulting for best practices.</td>
</tr>
<tr>
<td>Who is in charge to decide which services can be reused and the funding model to use them?</td>
<td>SOA Governance enablement and maturity</td>
<td>▶ Get help from the SOA CoE or the IT governance structure already in place.</td>
</tr>
<tr>
<td>From a technical perspective, you might be ready to migrate, but how do we get interest and funding from the line of business?</td>
<td>Business and IT alignment</td>
<td>▶ Get help from the SOA CoE for sponsorship.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Add business value to the project by using new product capabilities around BPM and human tasks for example.</td>
</tr>
<tr>
<td>It looks like the project is too costly because we need to run a parallel infrastructure during an extended period of time.</td>
<td>Project cost</td>
<td>▶ Investigate the possibilities around the interoperability of WebSphere Process Server and WebSphere Business Integration Server Foundation to reduce development costs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Investigate the extended entitlement possibilities.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Add business value to get more funding.</td>
</tr>
</tbody>
</table>

#### High-level planning
At this stage, it is possible to provide high-level planning with rough estimates for effort and costs.
4.2 Preparation phase

The preparation phase is intended to make sure that the project team is ready to implement the migration.

4.2.1 Building skills

Ensure that the skill set is properly updated to fulfill the migration implementation needs. The purpose is to have a sufficient knowledge of WebSphere Process Server. Education and training is a prerequisite, but it is generally not sufficient. Take time to gain practice from real experience to build the skills.

4.2.2 Using the WebSphere Process Server project experience

Ideally you should run a WebSphere Process Server project before doing the actual migration implementation. Indeed, if the project team has already gone through a full life cycle with WebSphere Process Server and has a working WebSphere Process Server infrastructure, then all of the related risks are already eliminated, and you can concentrate on the pure migration tasks.

Real experience can be obtained by doing a pilot project. A pilot has two advantages:

- The scope of the pilot can be adapted to fit in a restricted budget.
- The pilot does not put the business at risk.

A pilot project should have the following objectives:

- Build overall skills around WebSphere Process Server. Consider the following examples:
  - Development skills
    Set up best practices, methodology and so on.
  - Operations skills
    Set up new procedures and tools to administer the solution properly.
  - Project management skills
    Update or set up new project activities and refine estimates to get them delivered (workload assessment).

- Enable the new infrastructure for the WebSphere Process Server solution.
  A typical example is the effort around clustering and security. WebSphere Process Server provides native clustering for high availability (HA) management, load balancing, and integration with security managers to
define roles. These concepts require practical experience to be implemented efficiently, as in the following examples:

- Best practices to set up the cluster topology and the security roles through an LDAP directory or custom registry
- Best practices to use and administer the cluster, particularly monitoring a distributed platform, logs collection, and analysis, required infrastructure for shared resources highly availability, and so on

➤ Deliver new functionalities that add business and IT value, compared to the previous WebSphere Business Integration Server Foundation solution, as in the following examples:

- Use human tasks to create an application that is dedicated to functional error and used by business users, which in consequence spares IT resources that are usually dedicated to support users.
- Enable business rules to leave some control and dynamic decisions to the business users.
- Set up WebSphere Process Server clustering to raise high availability and load balancing capabilities.
- Enable technical event tracking or business monitoring through key performance indicators (KPIs), based on Common Event Infrastructure (CEI) capabilities.

➤ Implement integration patterns from scratch, gain an independent point of view that enables the team to optimize the code produced, and compare this code with the previous server version, WebSphere Business Integration Server Foundation code.

### 4.2.3 Refining the assessment

If the assessment cannot be sufficiently detailed during the first phase, you must refine it. You can conduct a detailed study to complete the missing points.

**Detailed study**

A detailed study might be required to specify the technical migration path to migrate the WebSphere Business Integration Server Foundation artifacts into WebSphere Process Server artifacts, for example, a gap analysis might be required for a precise status on the functional parity between WebSphere Business Integration Server Foundation and WebSphere Process Server products.
A gap analysis covers the following tasks:

- Identifies new features in WebSphere Process Server
- Identifies features that disappeared or are not equivalent in WebSphere Process Server
- Provides solutions to fill in the gaps

Chapter 6, “Products runtime comparison” on page 91, and Chapter 8, “Workspace migration” on page 163, are dedicated to product implementation details and can be used as input for the gap analysis.

**Decisions and precise estimates for effort and budget**

At this stage, it is possible to choose between the remaining options that come from the assessment phase and to assess precisely the corresponding resources, effort, and budget.

### 4.3 Implementation phase

Before starting the implementation phase, complete the following prerequisite tasks:

- The assessment and preparation phases must be done. By completing them, you can ensure that the right resources and skills are ready and setup to deliver the migration project.

- The overall strategy for migration must be precisely defined, which includes making decisions about how to migrate the business objects and knowing in which stages the migration should occur.

In the following sections, we describe the steps to execute the implementation phase. From this point, the migration project phases are close to the usual project life cycle. For a review of the project considerations, see 4.1.3, “Requirements gathering and impact analysis” on page 46. The emphasis is now on specific points of those considerations.

### 4.3.1 Development and migration

The development team must define the technical migration path, for example, the team must decide which artifacts can be migrated with the standard migration tooling and what needs to be customized and newly developed. The team can also consider completely new modules. The team must complete all the necessary development and configuration work of the new process and infrastructure at this time.
Perform the following tasks for development and migration in the implementation phase:

1. Prepare the environment.
   This task applies to preparation of the hardware infrastructure, data, middleware, user accounts, and so on. You can do this task in parallel with the next task. However, you must complete it prior to the new business objects that are being deployed onto the new environment.

2. Migrate the application.
   This task entails the following actions:
   - Perform process modeling, which is where business objects are migrated from the WebSphere Business Integration Server Foundation server to the new WebSphere Integration Developer tooling.
   - Migrate applications from WebSphere Business Integration Server Foundation to WebSphere Process Server.
   - Perform configuration work.
   - Perform customization and new development.
   - Prepare additional migration scripts.

4.3.2 Testing

Perform testing in iterations together with migration until acceptance criteria are met. Testing ensures that the migration approach is correct and that the applications work correctly in the new environment. Testing should include the following tests among others:

- Functional and integration (for new developments)
- Regression (for migrated developments)
- Performance
- Infrastructure

Perform the following tasks for testing in the implementation phase:

1. Design test scenarios.
2. Design and implement automatic tests.
3. Perform tests.

4.3.3 Deployment

Deployment is probably the most complex and risky part of migration. It is necessary to ensure business continuity and no loss of data, while providing the possibility to roll back if significant problems are revealed. For deployment, the
WebSphere Process Server environment must be set up (installation and configuration). Then the migrated components can be deployed into it. As a reminder, it is possible that, during the transition period, a parallel infrastructure is necessary.

Perform the following tasks for deployment in the implementation phase:

1. Back up the environment.
   Make a full backup of the production environment, which allows rollback when application migration or piloting harms production.

2. Migrate the application.
   Migrate production applications and deploy them to the new (preproduction) environment. Both the migration and the deployment are properly tested up front and automated to a maximum degree. This automatization helps to minimize the time that is needed for migration and, thus, reduces the difference between environment backup and pilot start.

3. Perform smoke testing.
   Quickly retest that the applications were migrated correctly.

4. Conduct a pilot run.
   In this phase, migrated applications are running in production in parallel with the applications that are in the old environment. Most of the users are still working with applications in the old environment. Selected users are using the new environment. This approach reduces risk in the following ways:
   - A non-functional pilot environment does not stop production. You still can use your old environment.
   - Problems in the pilot environment (bug fixes, outages) impact only a small number of users.
   There is enough time to find most of the problems before going to full production.

5. Ensure coexistence.
   Unless your migration project is small, perform the migration, development, and deployment in different phases. It is important to have coexistence, where both old and new systems are up and running so that the server can achieve high availability during the migration process. After the newly migrated application is fully tested, you can start introducing the new services in phases, redirecting the new client requests to the newly migrated components. Coexistence also ensures that you can roll back to the running production environment should any problems occur in the new environment.

6. Shut down the old environment and full production operation of the new environment.
When you are sure that the migrated applications are working properly, stop your old environment and redirect all users to the new environment.

### 4.3.4 Go live

As a reminder, the operations team must be able to administer the migrated solution properly.

**Important:** Keep in mind that there might be cases where a longer period of coexistence is needed, for example, to finish long running processes on the WebSphere Business Integration Server Foundation environment. We do not cover this case here; however, you can see 7.3.2, “Considerations for long running processes” on page 154, for more information about this approach.

### 4.4 Other considerations

In this section, we look at other factors that you might consider in your planning.

#### 4.4.1 Availability

The deployment topology of WebSphere Business Integration Server Foundation is slightly different than WebSphere Process Server. That is, you must assess your current environment from an availability and scalability point of view. Then you must select the corresponding deployment topology of WebSphere Process Server to ensure the desired level of business continuity. Overall availability of your environment is closely related to costs, and your decision will trade off between costs and availability factors. Figure 4-5 on page 68 shows the relationship between the level of availability and costs.
There are five standard levels of availability:

- **Basic systems**
  
  This level offers no protection of data and services. It only provides backups.

- **Redundant data**
  
  Disk redundancy or disk mirroring is used to protect against data loss.

- **Component failover**
  
  Hardware and software components are doubled, which means that at least two instances of the same component are in the system. In case of the failure of one instance, the second instance takes over the load. It differentiates between vertical clustering (for example, two or more instances of WebSphere Process Server on one hardware node) and horizontal clustering (for example, two or more hardware nodes that are running WebSphere Process Server).

- **System failover**
  
  A standby or backup system takes over for the primary system if the primary system fails. Usually, a clustering software monitors the health of the system and automatically fails over the load to a secondary system.

- **Disaster recovery**
  
  This level is similar to system failover. However, the primary system and the secondary system are at different geographical locations (primary site and
backup site). The secondary system takes over the complete load in case of the primary site’s loss.

The overall availability of the system depends on the availability of each of its components and how these components are connected to each other. The system is only truly highly available if each of the components has a failover partner. A system cannot be considered highly available if there is an instance of a single component, somewhere in the architecture, for example, if there is a single instance of a back-end database and no High-Availability Cluster Multi-Processing (HACMP™), on that database, then the database becomes the single point of failure, making the entire system not highly available. This state is still the case if you clustered the application servers and have high availability across the directory servers. High availability is only as strong as the weakest link.

**High availability in WebSphere Business Integration server Foundation**

A WebSphere Business Integration Server Foundation system, in general, is composed of the server itself, the Business Process Choreographer relational database, and a messaging infrastructure that is usually WebSphere MQ. A high availability solution is normally reached by clustering WebSphere, clustering WebSphere MQ, and using a high availability solution for the database.

**High availability in WebSphere Process Server**

At a high level, every WebSphere Process Server environment involves three fundamental layers: WebSphere Process Server applications, a messaging infrastructure, and one or more relational databases, as shown in Figure 4-5.

![Figure 4-5 Components that can be clustered in WebSphere Process Server](image)
The three layers are:

- **WebSphere Process Server applications**
  
  Clustering WebSphere Process Server applications is not significantly different from clustering a plain J2EE application in a WebSphere Application Server V6.1 environment.

- **Relational databases**
  
  WebSphere Process Server requires a certain application configuration and runtime configuration to be stored in relational database tables. The messaging infrastructure also uses relational database tables for persistence. A high availability clustering mechanism, such as HACMP, HADR or TSA for DB2®, are some mechanisms that can be considered to achieve this availability.

- **Messaging infrastructure**
  
  WebSphere Process Server requires the use of a messaging infrastructure. Some of that messaging infrastructure must use WebSphere Application Server service integration buses and the WebSphere Default Messaging Java Message Service (JMS) Provider.

  In addition, you have the option to choose WebSphere MQ as a messaging provider for the Business Process Choreographer component. Clustering the messaging infrastructure is perhaps the most complex aspect of the overall clustering. However, because the service integration bus is used, which requires WebSphere Application Server to run it, the messaging infrastructure can also be clustered by using the WebSphere clustering techniques. However, there are several aspects to consider when you select the topology to adopt. For the Business Process Choreographer, if you choose the messaging infrastructure WebSphere MQ, you have additional considerations too.

Consult the following references for more details and recommendations about a WebSphere Process Server environment for your specific requirements:

- **Basic steps for clustering WebSphere Process Server**:
  
  - *IBM WebSphere Developer Technical Journal: Basic steps for clustering WebSphere Process Server*
    

  - IBM WebSphere Process Server documentation
    
4.4.2 Security

The security of WebSphere Business Integration Server Foundation and WebSphere Process Server is based on the security mechanisms for the base Application Server security. You can divide the security tasks into security of the server environment and security of the applications that are running on the server. In WebSphere Application Server V6.1, several enhancements and features were added about how security is handled. Development resources must be aware of these changes.

Securing the server environment

Securing the environment involves enabling global security, creating profiles with security, and restricting access to critical functions to selected users. In WebSphere Application Server V6.1, the following features were added regarding security:

- Three administrative roles with finer levels of authorization
- Bootstrap secure messaging
- Enhanced LDAP support

For a more comprehensive list of these features, see “What is new for security specialists” in the WebSphere Application Server Information Center:

The following security components make up the WebSphere Application Server security mechanism:

- User registry
  
  WebSphere Process Server supports various user registries (local operating system, LDAP, custom registry, and federated repositories) that stay the same compared to WebSphere Business Integration Server Foundation.

- Transport layer security (Secure Sockets Layer (SSL))

- Service integration bus security:
  - JMS or WebSphere MQ security
  - Web services security

**Securing applications**

Securing an application involves the following aspects:

- Authentication
  
  A user or a process that invokes an application must be authenticated.

- Access control
  
  Access control determines whether the authenticated user has permission to perform the operation.

- Integrity and privacy of the data that an application accesses

- Identity propagation with single sign-on
  
  Identity propagation with SSO permits a user to provide authentication data one time and then passes this authentication information to downstream components.

The components of Business Process Choreographer are implemented as WebSphere J2EE applications and thus adopt the security mechanisms of WebSphere Application Server. WebSphere Application Server implements and extends Java and J2EE security standards. Consider and configure the security of the following artifacts:

- Application security, which includes securing application resources, mapping security roles to users and groups and others, and securing both Web and EJB applications.

- Client security

See the WebSphere Process Server Information Center for more comprehensive information about creating a secure WebSphere Process Server environment:

Performance considerations
You can assess and prepare the WebSphere Process Server environment to run applications with the desired throughput and response times. Conceptually, the principles of performance tuning and scalability are similar in WebSphere Business Integration Server Foundation and WebSphere Process Server.

Short running BPEL processes are synchronous and transactional. Long running processes and asynchronous SCA invocations work by using the messaging infrastructure and the underlying relational databases to ensure persistence of a state.

The performance considerations to keep in mind are:

► The safest approach to size your WebSphere Process Server environment is to prepare and run performance tests on a proof of concept and derive your sizing based on its result. Your proof of concept must have the following characteristics:
  – Cover the typical transactions that are provided by your application.
  – Ensure that the testing environment and transaction distribution are as close to the production environment as possible.
  – Allow for the sizing of your testing environment to be smaller than your production environment. You can extrapolate the target throughput from the scalability curve. We provide one at the end of this section.

► Sometimes it is better to modify or redesign the migrated business processes for better performance, if it is in accordance with business requirements.

► Use the following hints when you migrate:
  – Avoid SOAP because it is a heavy processing protocol.
  – Do not carry too much data through the flow.
  – Avoid subprocesses, if possible.
  – Do business-level modeling. Do not get down to too low-level programming in BPEL.

In addition, your test cannot simulate production 100% percent because you cannot simulate all of the possible user interactions.

WebSphere Process Server provides configuration parameters that have a significant impact on performance. Therefore, set your environment according to the available performance tuning guidelines and configuration, which are optimized during performance tests.
Tuning recommendations

In this section, we discuss performance tuning tips for WebSphere Process Server. Performance tuning is always left as a step at the end of the migration, and in many cases, this step is left out. However this step of the migration process can give you biggest benefit, when done correctly. See the following Web addresses for information and tips about performance tuning for WebSphere Application Server V6.1:

- “WebSphere tuning for the impatient: How to get 80% of the performance improvement with 20% of the effort”
  
  [Link to article](http://www.ibm.com/developerworks/websphere/library/techarticles/0602_lurie/0602_lurie.html)

- “Tuning performance” in the WebSphere Application Server Information Center
  

Keep in mind the following basic principles and guidelines to follow when performance tuning your WebSphere Process Server environment:

- Run WebSphere Process Server in production mode in production environments. Do not run it in development mode in production environments.

- Disable Tracing and Monitoring when possible.

- Move databases off of Cloudscape (the default).

- Locate messaging (service integration bus) data stores on high-performance database management systems (DBMSs).

- Configure threads for messaging and work managers.

- Tune thread pools on your JDBC™ providers.

- Configure the statement cache for long running processes.

- Use the appropriate Java heap size in all environments (collect verbose gc and graph it). Unless specified specifically by the IBM labs through a problem management record (PMR) process, never set your Java virtual machine (JVM™) heap size to larger than 1500 MB.

- Regarding JVM, keep in mind the following guidance:
  - Set the heap and nursery size to handle garbage collection efficiently.
  - For optimal performance, run the heap with less than 60%, possibly even 50%, occupancy.
  - Set AIX® threading parameters.
  - Use HotSpot server instead of client.
– Set the thread stack size if you are using many threads.
– Unless specified specifically by the IBM labs through a PMR process, never set your JVM heap size to larger than 1500 MB.
– Adjust the heap size, if java.lang.OutOfMemory occurs.

Performance tuning is an iterative process. It must be given time and focus as you move from WebSphere Business Integration Server Foundation to WebSphere Process Server, enabling new features and enhancements, which can all impact performance on the environment. See “MustGather: Performance problems with IBM WebSphere Process Server” at the following Web address to assist you in troubleshooting performance problems in WebSphere Process Server:

http://www-1.ibm.com/support/docview.wss?uid=swg21269374

4.5 Product coexistence

WebSphere Process Server and WebSphere Business Integration Server Foundation environments must coexist under the following circumstances:

► Have both environments up and running during the pilot phase.
► Migrate your applications in several stages.

To achieve a staged and controlled migration, isolate the business objects by two distinct groupings, functional and components.

First perform a proof of concept and choose one of your least critical business applications. When proven, conduct a staged approach to migrate your most difficult application last for the following purposes:

– Prove that the infrastructure works with your newly migrated application.
– Build skills, experience, and confidence within the team that is tasked with the implementation of the WebSphere Process Server environment.

► Achieve a period of coexistence to finish long running processes on WebSphere Business Integration Server Foundation while you start working on migrated processes in the new WebSphere Process Server environment.

A WebSphere Business Integration Server Foundation and WebSphere Process Server installation can run on the same machine. Therefore, ensure that they are not running in port conflicts. See “Coexisting” in the WebSphere Process Server Information Center at the following Web address for further detail:

You might need to call services or even processes on the WebSphere Business Integration Server Foundation environment from the new WebSphere Process Server environment, which is possible using typical methods, such as Remote Method Invocation (RMI) calls or Web services calls. However, make these scenarios part of your test plan to ensure the interoperability between both systems.
Migration procedure

In this part, we describe the migration procedure from WebSphere Business Integration Server Foundation to WebSphere Process Server. We also discuss the best practices to follow during the migration and the options and tools to use during this process.

This part includes the following chapters:

In Chapter 5, “Best practices” on page 79, we discuss best practices for the migration procedure, which includes the planning phase and artifacts preparations phase.

In Chapter 6, “Products runtime comparison” on page 91, we explain the difference between WebSphere Business Integration Server Foundation and WebSphere Process Server runtime environment and discuss the necessary considerations regarding the runtime migration.

In Chapter 7, “Migration options” on page 119, we describe the common and support migration path for migrating WebSphere Business Integration Server Foundation to WebSphere Process Server. We also introduce tools and options that are used during migration, such as WebSphere Integration Developer, WebSphere Business Modeler, and command-line interface (CLI) and scripts. In this chapter, we focus on the tools’ capabilities with regard to migration.
In Chapter 8, “Workspace migration” on page 163, we introduce the new workspace migration features that are available in WebSphere Integration Developer V6.2 and discuss the necessary preparations before the workspace migration and general migration task.

In Chapter 9, “Client component migration” on page 193, we discuss the different options and approaches for migrating different client applications from WebSphere Business Integration Server Foundation to WebSphere Process Server.

In Chapter 10, “Post migration tasks” on page 215, we discuss the considerations and best practices for tuning your application after migration.

In Chapter 11, “Technical scenarios” on page 223, we include step-by-step instructions for two migration scenarios. We illustrate the end-to-end scenarios for migrating an existing running application in a WebSphere Business Integration Server Foundation environment to WebSphere Process Server and for adding new functions to further extend the application.

In Chapter 12, “Troubleshooting” on page 359, we cover hints and tips for resolving possible errors or problems during the migration process.
Best practices

In this chapter, we discuss best practices for the migration process of WebSphere Business Integration Server Foundation to WebSphere Process Server.

This chapter includes:

- “Planning” on page 80
- “Artifacts preparation” on page 80
## 5.1 Planning

The best practices for planning are:

- Document the system and process design both before and after migration. Be sure to capture the integration architecture and design, business processes, design decisions made during migration, and quality of service requirements because it minimizes any rework efforts.

- Implement the WebSphere Process Server production environment and migrate processes to this environment.

- Run the production pilot in parallel with your WebSphere Business Integration Server Foundation environment.

  A pilot impacts only a limited number of users and helps to fine-tune your application in the production environment. A pilot also allows for seamless rollback.

- Perform exhaustive and functional integration and performance tests. Using this approach helps you to avoid many production problems.

- Using the source control system for managing your source code, cut new release/baseline for the migration project.

- Back up your environment and be prepared to do a rollback. If serious problems occur, you can roll back to your original environment.

- Assess non-functional metrics of your old environment, and define a new service level agreement (SLA).

  The SLA defines the service level that is provided by your environment (or application) to other environments or organizational units. Plan your environment to meet the SLA. Migration is a good time to reconsider the non-functional requirements and topology of your environments.

- Prepare a testing (staging) environment that is as close to production as possible.

  Different testing and production environments usually result in discovering many problems when you get into production.

## 5.2 Artifacts preparation

In this section, we discuss best practices for preparing artifacts for a migration project.
5.2.1 Artifact preparation for WebSphere Studio Application Developer Integration Edition

The following list summarizes the best practices for preparing the artifacts for WebSphere Studio Application Developer Integration Edition:

- Follow the recommendations for Web Services Description Language (WSDL) files in 5.2.4, “WSDL files and Web services” on page 83.
- Follow the naming conventions that are defined for the project. We provide some recommendations in 5.2.8, “Naming convention” on page 86.
- Organize the projects in the workspace and prepare them with a rebuild and cleaning process. Leave only the artifacts that you really must migrate, and follow the recommendations in 8.2.2, “Pre-migration considerations” on page 166.
- Change the namespaces to avoid name conflicts. Follow the recommendations for XML Schema Definition (XSD) files in 5.2.3, “Schema XSD files” on page 81.
- Assure that the reference of all artifacts is correct. Define the shared components.
- Follow the recommendations for Business Process Execution Language (BPEL) files in 5.2.5, “BPEL artifacts” on page 84.
- Do not migrate the generated code.

5.2.2 Preparing the modeler artifacts

The following list summarizes the best practices for WebSphere Business Modeler:

1. Prepare the workspace. Delete the projects that are not necessary to migrate.
2. Review the name convention, and change the names before you migrate.
3. Review and correct all of the errors in the model.
4. Change the catalog names, and follow the name conventions.

5.2.3 Schema XSD files

The following list summarizes the best practices for schemas:

- In WebSphere Process Server V6, do not use two different WSDL or XSD definitions that have the same name and target namespace because they are not allowed. Even in WebSphere Business Integration Server Foundation, this situation is possible.
When the XSDs with the same name and namespaces refer to the same entity, leave only one definition, and delete the copies. Clean and rebuild the project. Solve the reference problems by pointing to the remaining type.

If the XSD refers to different types:
- When there are few conflicts, change the name of the definitions.
- When many name definitions are in conflict, change the namespace.
- When some imports in a WSDL conflict in the namespace, introduce the imports in import chaining. Only one import is allowed for each namespace per WSDL file.

Be specific. Where possible, avoid XSD complex types that have references to the `xsd:anyType` type.

Ensure that all complex types are given a name and that each complex type can be uniquely identified by its target namespace and name, for example, avoid anonymous references by using a complete definition.

Table 5-1 shows an example of an anonymous reference.

<table>
<thead>
<tr>
<th>Anonymous reference to avoid</th>
<th>Complete definition with name and namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;schema attributeFormDefault=&quot;qualified&quot;\n        elementFormDefault=&quot;unqualified&quot;\n        targetNamespace=&quot;<a href="http://booking.bpe.samples.webSphere.ibm.com%22%5Cn">http://booking.bpe.samples.webSphere.ibm.com&quot;\n</a>        xmlns=&quot;<a href="http://www.w3.org/2001/XMLSchema%22%5Cn">http://www.w3.org/2001/XMLSchema&quot;\n</a>        xmlns:tns=&quot;<a href="http://booking.itso.ibm.com%22%3E%5Cn">http://booking.itso.ibm.com&quot;&gt;\n</a>    &lt;element name=&quot;CustomerElement&quot;&gt;\n        &lt;complexType&gt;\n            &lt;all&gt;\n                &lt;element name=&quot;id&quot; type=&quot;string&quot;/&gt;\n                &lt;element name=&quot;name&quot; type=&quot;string&quot;/&gt;\n                &lt;element name=&quot;address&quot; type=&quot;string&quot;/&gt;\n            &lt;/all&gt;\n        &lt;/complexType&gt;\n    &lt;/element&gt;\n&lt;/schema&gt;</td>
<td>&lt;schema attributeFormDefault=&quot;qualified&quot;\n        elementFormDefault=&quot;unqualified&quot;\n        targetNamespace=&quot;<a href="http://booking.bpe.samples.webSphere.ibm.com%22%5Cn">http://booking.bpe.samples.webSphere.ibm.com&quot;\n</a>        xmlns=&quot;<a href="http://www.w3.org/2001/XMLSchema%22%5Cn">http://www.w3.org/2001/XMLSchema&quot;\n</a>        xmlns:tns=&quot;<a href="http://booking.itso.ibm.com%22%3E%5Cn">http://booking.itso.ibm.com&quot;&gt;\n</a>    &lt;complexType name=&quot;Customer&quot;&gt;\n        &lt;all&gt;\n            &lt;element name=&quot;id&quot; type=&quot;string&quot;/&gt;\n            &lt;element name=&quot;name&quot; type=&quot;string&quot;/&gt;\n            &lt;element name=&quot;address&quot; type=&quot;string&quot;/&gt;\n        &lt;/all&gt;\n    &lt;/complexType&gt;\n    &lt;element name=&quot;CustomerElement&quot; type=&quot;tns:Customer&quot;/&gt;\n&lt;/schema&gt;</td>
</tr>
</tbody>
</table>
In WebSphere Process Server V6.2, two different WSDL or XSD definitions that have the same name and target namespace are not allowed. Another possibility is in the generated artifacts:

- For any WSDL or XSD that is generated from an Enterprise JavaBean (EJB) or JavaBean, ensure that the target namespace is unique. The Java class name and package name are represented by the target namespace.
- When you use the Web service wizard without specifying the target namespace explicitly. Define a custom package to namespace mapping in the wizard to ensure that the namespace of the generated files is unique for the given class.
- When you use the Java2WSDL command without specifying the target namespace, define the -namespace Java2WSDL command line option to ensure that the namespace of the generated files is unique for the given class.

### 5.2.4 WSDL files and Web services

The following list summarizes the best practices for the WSDL files:

- Follow the Web Services Interoperability (WS-I) specification.
- Define only one part per WSDL message. This decision follows the WS-I specification, which is the WebSphere Process Server V6.2 preferred style.
- When possible, use the document or literal style for the WSDL file, which is the preferred style in WebSphere Process Server V6.2.
- Ensure that all complex types are given a name and that each complex type can be uniquely identified by its target namespace and name.
- Generate service deploy code by using IBM Web services (as opposed to Apache SOAP/HTTP). IBM Web services are directly supported in WebSphere Process Server V6.2, and Apache Web services are not supported.
- In WebSphere Studio Application Developer Integration Edition V5.1, you have two ways in which you can organize WSDL and XSD files to minimize the amount of reorganizing during migration. In WebSphere Process Server V6.2, shared artifacts, such as WSDL and XSD files, must be located in WebSphere Business Integration projects (WebSphere Business Integration modules and libraries) to be referenced by a WebSphere Business Integration service.
- When more than one service project shares WSDL files, use a Java project that the service projects can reference. In the migration, create a new WebSphere Business Integration Library with the same name as the WebSphere Studio Application Developer Integration Edition V5.1 shared
Java project. Before any other migration, copy the artifacts in the Java project to the library. The migration wizard can resolve the artifacts when it migrates the services projects that use those artifacts.

- Keep a local copy of all WSDL or XSD files that a service project references in the service project itself. The service project IDs of WebSphere Studio Application Developer Integration Edition are migrated to a WebSphere Business Integration module in WebSphere Integration Developer. A module cannot have dependencies on other modules, for example, a service project with dependencies on another service project, for the sake of sharing WSDL or XSD files, is not migrated cleanly.

- Define the WSDL interfaces precisely. Avoid XSD complex types that have references to the `xsd:anyType` type, where possible.

- Use a bottom-up approach to write the Java or EJB services. First, write the Java interface, and then generate the service. Avoid creating top-down EJBs or Java services where possible because the Java or EJB skeleton that gets generated from the WSDL PortTypes or messages depend on Web Services Invocation Framework (WSIF) classes (WSIFFormatPartImpl).

- Avoid creating or using WSDL interfaces that reference the `soapenc:Array` type because this type of interface is not natively supported in the Service Component Architecture (SCA) platform. Also avoid creating message types whose high-level element is an array type, for example, the maxOccurs attribute is greater than one. This type of interface is not natively supported in the SCA programming model.

### 5.2.5 BPEL artifacts

The following list summarizes the best practices for BPEL files:

- Use the assign activity wherever possible as opposed to the transformer service, which is only needed when an advanced transformation is needed. Use this practice because intermediate components must be constructed in order for an SCA module to invoke a transformer service. Additionally, there is no special tooling support in WebSphere Integration Developer for the transformer services that are created in WebSphere Studio Application Developer Integration Edition V5.1. You must use the WSDL or XML editor to modify the Extensible Stylesheet Language Transformation (XSLT) that is embedded in the WSDL file if it is needed to change the behavior of the transformer service.

- In the Java snippets, avoid sending WSIFMessage parameters to any custom Java classes. Also, do not use the WSIF metadata APIs, if possible.
5.2.6 Client source

The following list summarizes the best practices for the client source migration:

- Do not develop any new clients that adopt the CORBA IDL interface APIs. This is not supported in WebSphere Process Server.
- Do not develop any new clients that adopt the JMS Business process APIs, which is not supported in WebSphere Process Server. Avoid using the Business Process Choreographer Generic Messaging API (generic message-driven beans (MDBs)) because it is not provided in WebSphere Process Server V6. An MDB interface that offers late binding is not available in WebSphere Process Server.
- Invoke the generic Choreographer API, which is preferred to invoking the generated session beans that are specific to a particular version of a process (including the valid-from string).
- If you have a business process with multiple replies for the same operation, ensure that, if any of them has client settings, all replies for that operation have the same client setting. In WebSphere Process Server, each operation replay has only one client setting.

5.2.7 Modeling migration best practices

The best practices for the modeling migration are:

- Use the latest fix pack that is available for each modeler tool. In some unfixed versions, the error view can show errors that are not synchronized with the model.
- Migrate the whole workspace, which is faster than migrating each project.
- Before you migrate a workspace, delete the unused projects.
- Do not use special characters.
- Open, check, and clean each model before the migration process.
- Use the source control system, and cut separated release/baseline, or always keep a functional copy of the original model and a copy of the WebSphere Business Integration Modeler in a different machine until you finish the migration.
- Ensure that there is enough free disk space before you start a workspace migration. WebSphere Business Modeler makes an additional backup of the workspace.
5.2.8 Naming convention

Use the following recommendations for naming conventions in your project:

- Try to install the products in a folder with a short path, and do not use names that are too long. There is a restriction in the length of the names and paths to the artifacts in Java. In some operating systems, the length is 259 characters.

- Do not use acronyms and abbreviations unless the abbreviation is widely used.

- Follow the Java naming convention in which the first letter of a word and the first letter of each internal word are capitalized.

- For the artifacts, use English characters. These names generate code artifacts and sometimes they are mapped, removing spaces and changing capitalization.

- Be consistent in naming the artifacts. It can be helpful to use a suffix in each artifact type, as in the following examples:
  - Libraries: Name + “Lib”
  - Interfaces: Name + “Interface”
  - SCA components:
    - Human task: + “HT”
    - State machine: Name + “SM”.
    - Rule group: Name + “RG”.
    - Rules - Rule set: Name + “RS”.
    - Rules - Decision table: Name + “DT”.
    - Selector: Name + “SL”.
    - Java object: Name + “POJO” or “JV”

- Define a consistent pattern for the namespaces, such as:
  http://modulename/interfacename/project

5.2.9 Modeler naming convention

Follow these recommendations for modeler naming:

- Be consistent in naming conventions. A model project can use many activities. Define the convention for elements, and follow this convention in all activities, business items, input, and outputs. Also, use a convention for the metrics and KPIs (not covered in this book).

- Avoid non-English characters. Place the first letter of each word in uppercase. Do not use long activity names.

- Consider the possibility of changing each element name and defining a more appropriate name. Some default value names of modeler are not the best
name for some entities, for example, the default “Input Criteria” and “Output Criteria” name that is used in each activity names the operations and the methods in Java implementation.

- Because the names of the generated artifacts are a translation of the names in modeler, try the translated names, and follow the naming convention. Table 5-2 shows some translations that are performed in the code generation.

<table>
<thead>
<tr>
<th>Artifacts in WebSphere Business Modeler</th>
<th>Artifacts in WebSphere Studio Application Developer Integration Edition</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog name</td>
<td>Business items/data translated to package name</td>
<td>Do not use several levels, for example, com.ibm.itsso.project. It works for WebSphere Business Integration Server Foundation, but not to migrate to WebSphere Process Server.</td>
</tr>
<tr>
<td>Business item name</td>
<td>Booking order</td>
<td>The translator suppresses the spaces. Use uppercase on the first letter of each word.</td>
</tr>
<tr>
<td>Process catalog name</td>
<td>Data</td>
<td>Starts with a lowercase letter.</td>
</tr>
<tr>
<td>Process name</td>
<td>Booking order</td>
<td>Use uppercase on the first letter of each word.</td>
</tr>
<tr>
<td>Activity not human task</td>
<td>Reservation System</td>
<td>Use uppercase on the first letter of each word.</td>
</tr>
<tr>
<td>Input and output logic</td>
<td>Input criteria</td>
<td>Consider defining a verb as a name.</td>
</tr>
</tbody>
</table>
The mapping to WebSphere Integration Developer from modeler artifacts is different, as shown in Table 5-3.

**Table 5-3  Example name mapping to WebSphere Integration Developer**

<table>
<thead>
<tr>
<th>Artifacts in WebSphere Business Modeler</th>
<th>Artifacts in WebSphere Studio Application Developer Edition</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td><strong>Example</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Input and Output logic</td>
<td>Input criteria Output criteria</td>
<td>Method name in Java service implementation</td>
</tr>
<tr>
<td>Catalog name</td>
<td>Business items/data</td>
<td>A folder in the Library project</td>
</tr>
<tr>
<td>Catalog name</td>
<td>Business items/data</td>
<td>The XSD file</td>
</tr>
<tr>
<td>Business item name</td>
<td>Customer</td>
<td>Elements in the XSD file</td>
</tr>
<tr>
<td>Process catalog name</td>
<td>data</td>
<td>Folders</td>
</tr>
<tr>
<td>Process name</td>
<td>Booking order and BPEL artifacts</td>
<td>Folder name and BPEL artifacts</td>
</tr>
<tr>
<td>Artifacts in WebSphere Business Modeler</td>
<td>Artifact in WebSphere Studio Application Developer Integration Edition</td>
<td>Recommendation</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>Example</strong></td>
<td></td>
</tr>
<tr>
<td>Human Task Activity</td>
<td>Booking order form</td>
<td>Starts with lowercase. Use uppercase on the first letter of each word.</td>
</tr>
<tr>
<td></td>
<td>Inline tel file (itel file)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BookingOrderForm_1872960898.itel</td>
<td></td>
</tr>
<tr>
<td>Java Service Activity</td>
<td>Reservation system</td>
<td>Starts with lowercase. Use uppercase on the first letter of each word.</td>
</tr>
<tr>
<td></td>
<td>Java file</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ReservationSystem_01098838552Impl.java</td>
<td></td>
</tr>
<tr>
<td>Input and Output logic</td>
<td>Input criteria, Output criteria</td>
<td>Consider defining a verb as the name.</td>
</tr>
<tr>
<td></td>
<td>Operation name in BPEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;bpws:invoke name=&quot;BookingOrderForm_InputCriteria&quot; operation=&quot;InputCriteria&quot; ...&gt;</td>
<td></td>
</tr>
<tr>
<td>Input and Output logic</td>
<td>Input criteria, Output criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WSDL message name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>in BookingOrderInterface.wsdl &lt;wsdl:portType name=&quot;BookingOrder&quot;&gt; &lt;wsdl:operation name=&quot;InputCriteria&quot;&gt; &lt;wsdl:input message=&quot;tns:InputCriteria&quot; name=&quot;InputCriteriaRequest&quot;/&gt; &lt;wsdl:output message=&quot;tns:OutputCriteria&quot; name=&quot;InputCriteriaResponse&quot;/&gt; &lt;/wsdl:operation&gt; &lt;/wsdl:portType&gt;</td>
<td></td>
</tr>
<tr>
<td>Input and Output logic</td>
<td>Input criteria, Output criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Method in Java service</td>
<td></td>
</tr>
<tr>
<td></td>
<td>public DataObject InputCriteria(DataObject reservationInput)</td>
<td></td>
</tr>
</tbody>
</table>
Products runtime comparison

In this chapter, we discuss and highlight the changes and improvements between the runtimes of the WebSphere Business Integration Server Foundation and WebSphere Process server V6.2. We provide this information to give you an understanding of the necessary migration tasks that must be performed. We provide a summary of the changes in a table that gives you the old WebSphere Business Integration Server Foundation components and their new relative components within WebSphere Process Server.

The topics that in this chapter are:

- “Products overview and comparison” on page 92
- “Base application server level features” on page 93
- “Business Process Choreographer Features” on page 100
- “Administration in WebSphere Process Server using a console and scripts” on page 108
6.1 Products overview and comparison

WebSphere Business Integration Server Foundation and WebSphere Process Server are based on WebSphere Application Server. In this section, we provide comparative information about the relationship of WebSphere Business Integration Server Foundation and WebSphere Process Server.

Although both products are based on similar technology, migration of the runtime environment is not supported.

For a detailed product overview of both WebSphere Business Integration Server Foundation and WebSphere Process Server please refer to Chapter 2, “Product overview” on page 9.

6.1.1 Feature comparison

In Table 6-1, we compare the features of the old WebSphere Business Integration Server Foundation and WebSphere Process Server runtimes. Each feature is explained in detail later in this chapter.

<table>
<thead>
<tr>
<th>WebSphere Business Integration Server Foundation</th>
<th>WebSphere Process Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes. See 6.2, “Base application server level features” on page 93</td>
<td>Profiles. See 6.2, “Base application server level features” on page 93</td>
</tr>
</tbody>
</table>
6.2 Base application server level features

Currently, there is no way to preserve the configuration information of a WebSphere Business Integration Server Foundation runtime environment for the new WebSphere Process Server environment. It is not possible to run a network deployment environment with a Deployment Manager that manages both WebSphere Business Integration Server Foundation and WebSphere Process Server nodes, servers, and clusters, which is due to the way nodes are different between the two products. In WebSphere Application Server V6 and later, the concept of profiles was introduced, which replaced the way nodes in WebSphere Business Integration Server Foundation are formed.

For details about profiles in WebSphere Application Server V6, see “Profile concepts” in the WebSphere Application Server Information Center at the following address:

6.2.1 Messaging in WebSphere Business Integration Server Foundation

Because WebSphere Process Server is based on a newer version of WebSphere Application Server, one important change in the WebSphere Application Server that is relevant to WebSphere Process Server is the introduction of the service integration bus.

In WebSphere Business Integration Server Foundation communication occurred through asynchronous messaging using the Java Message Service (JMS) interface. There was the choice of either base JMS or alternatively extended messaging utilizing the Enterprise JavaBean (EJB) container, typically WebSphere MQ was used as a JMS provider.

For further details pertaining to WebSphere Business Integration Server Foundation messaging see messaging within WebSphere Business Integration Server Foundation at the following address:


6.2.2 Messaging in WebSphere Process Server

Because WebSphere Process Server is based on a newer version of WebSphere Application Server, one important change in the WebSphere Application Server that is relevant to WebSphere Process Server is the introduction of the service integration bus.

Service integration bus

The service integration bus is a Java Message Service (JMS) 1.1 compliant JMS provider for reliable message transport. It has the capability of intermediary logic to adapt message flow intelligently in the network.

The capabilities of service integration bus are fully integrated within WebSphere Application Server, which enables it to take advantage of WebSphere security, administration, performance monitoring, trace capabilities, and problem determination tools. The service integration bus consists of the following parts:

- Bus members
  Bus members are application servers or clusters that are added to the bus.

- Messaging engines
  These engines are the application servers or cluster components that manages bus resources. When a bus member is defined, a messaging
engine is automatically created on the application server or cluster. The messaging engine provides a connection point for clients to produce or from where to consume messages.

An application server has one messaging engine per bus of which it is a member. A cluster has at least one messaging engine per bus and can have more. In this case, the cluster owns the messaging engines and determines on which cluster members the messaging engines run.

- **Destinations**
  Destinations refer to the place within the bus to which applications attach to exchange messages. Destinations can represent Web services endpoints, messaging point-to-point queues, or messaging publish and subscribe topics. Destinations are created on a bus and hosted on a messaging engine.

- **Message store**
  Each messaging engine uses a set of tables in a supported data store, such as a Java Database Connectivity (JDBC) database, to hold information, such as messages, subscription information, and transaction states. Messaging engines, each using its own set of tables, can share a database.

**Service integration bus messaging in clusters**
In a distributed server environment, you can cluster for high availability and scalability. You can add a cluster as a bus member and achieve goals:

- **High availability**: One messaging engine is active in the cluster. In the event that the messaging engine or server fails, the messaging engine on a standby server is activated.

- **Scalability**: A single messaging destination is partitioned across multiple active messaging engines in the cluster. The messaging order is not preserved.

**Service integration buses in WebSphere Process Server**
WebSphere Process Server has at least two, and at most four, buses. The Service Component Architecture (SCA) requires the following two buses with a messaging engine to be present in order to run:

- **SCA.SYSTEM bus**: This bus accomplishes asynchronous messaging between SCA components and modules. It is required to run any module.

- **SCA.APPLICATION bus**: This bus interacts between a module and an external resource, such as Adapters or JMS resources.

The Business Process Choreographer component of WebSphere Process Server, which consists of the Business Flow Manager and the Human Task Manager, requires one additional bus to execute business process and human task components within SCA modules. This bus is the *Business Process*
Choreographer bus. This bus is used for internal message processing while processing long running business processes and human tasks. To use the Common Event Infrastructure (CEI), another bus is required, which is the CommonEventInfrastructure bus. This bus is used to transmit Common Base Events asynchronously to the CEI server.

Figure 6-1 shows the Integrated Solutions Console of a stand-alone WebSphere Process Server, where all four buses are configured.

In WebSphere Business Integration Server Foundation, the following queues were used by Business Process Choreographer:

- BPEIntQueue
- BPEApiQueue
- BPERetQueue
- BPEHldQueue

In WebSphere Process Server, the BPEApiQueue is no longer used because the JMS API that is provided with WebSphere Business Integration Server Foundation is not supported in WebSphere Process Server.
All other queues are also required in Business Process Choreographer within WebSphere Process Server. Also, there are additional queues for the new human task container. The following queues are needed by Business Process Choreographer within WebSphere Process Server:

- BPEIntQueue
- BPERetQueue
- BPEHldQueue
- HTMIntQueue
- HTMHldQueue

In WebSphere Process Server, two options are available for the messaging system where the queues must reside: the JMS provider and the WebSphere MQ JMS provider.

**Default JMS provider**
The default JMS provider is the service integration bus that we mentioned in “Service integration buses in WebSphere Process Server” on page 95. A Business Process Choreographer bus must be present when you use this option. When you use the installation wizards for Business Flow Manager and Human Task Manager, this bus, which includes a messaging engine and the queues listed in the previous section, is automatically created and configured.

**WebSphere MQ JMS provider**
WebSphere MQ can also be used as a JMS provider. Here, the queue manager and the queues must be created on your own. See “Creating the queue manager and queues for Business Process Choreographer” for documentation and scripts that support these steps for a queue manager or a WebSphere MQ Cluster at the following Web address:


Avoid using the WebSphere MQ messaging provider and instead use the service integration bus for Business Process Choreographer in WebSphere Process Server because:

- Although WebSphere MQ is supported for the Business Process Choreographer in WebSphere Process Server, the wizards to set up the business process container and the human task container create service integration bus resources. Configuring WebSphere MQ is a manual and relatively complex process.

WebSphere Process Server uses the service integration bus also for the SCA run time. For SCA only, the service integration bus can be used as a messaging provider. If you choose to preserve WebSphere MQ as a messaging provider for
the Business Process Choreographer, the configuration must include both the service integration bus and WebSphere MQ as messaging providers, which is an unnecessary complication.

6.2.3 Database required by WebSphere Business Integration Server Foundation


However, the database schema and the items that are stored in the database changed between these products. Therefore, it is not possible to reuse the Business Process Choreographer database from the WebSphere Business Integration Server Foundation environment on the new WebSphere Process Server environment. It is also not possible to move any kind of template or instance data from a WebSphere Business Integration Server Foundation Business Process Choreographer database to a WebSphere Process Server Business Process Choreographer database.

In regard to both products, in a clustered Business Process Choreographer setup, one database serves all of the business process containers in the WebSphere cluster. In a non-clustered setup, the database is dedicated to the business process container on one server, which means that each deployment target (server or cluster) requires its own, isolated database for Business Process Choreographer.

For more information about creating the Business Process Choreographer database, see “Business Process Choreographer database configurations” in the WebSphere Process Server Information Center at:


When you use Business Process Choreographer, WebSphere Business Integration Server Foundation requires one database to store business process templates and instance data.

Table 6-2 on page 99 lists the components of WebSphere Business Integration Server Foundation that require database tables, the default database names, and where the tables that are associated with these components are stored. The database names are configurable.
6.2.4 Database required by WebSphere Process Server

WebSphere Process Server has new features, such as relationships, business rules, and selectors. Some of these features also have the requirement to store data.

Table 6-2 lists the components of WebSphere Process Server that require database tables, the default database names, and where the tables that are associated with these components are stored. The database names are configurable.

Table 6-2 Database within WebSphere Business Integration Server Foundation

<table>
<thead>
<tr>
<th>Database (default name)</th>
<th>Description</th>
<th>Components using the database</th>
</tr>
</thead>
</table>
| BPEDB                  | - Business Process Choreographer database that stores template and instance data of business processes and staff activities.  
- Is required for executing business processes and staff activities. | - Business Processes  
- Staff Activities |

Table 6-3 Databases within the WebSphere Process Server

<table>
<thead>
<tr>
<th>Database (default name)</th>
<th>Description</th>
<th>Components using the database</th>
</tr>
</thead>
</table>
| BPEDB                  | - Business Process Choreographer database that stores template and instance data of business processes and human tasks.  
- Is required for executing business processes and human tasks. | - Business Processes  
- Human Tasks  
- Business State Machines |
| WPRCSDB                | - Common database that acts as a repository by various components.  
- Is required. | - Relationship service  
- Recovery  
- Mediation  
- Application Scheduler  
- Customization (selectors and rule group)  
- Event sequencing  
- ESB logger mediation primitive |
### Database (default name) Description Components using the database

<table>
<thead>
<tr>
<th>Database (default name)</th>
<th>Description</th>
<th>Components using the database</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEIDB</td>
<td>- CEI database that stores Common Base Events.</td>
<td>- Common Event Infrastructure</td>
</tr>
<tr>
<td>EsbLogMedDB</td>
<td>- The enterprise service bus (ESB) logger mediation database is used by the MessageLogger mediation primitive. &lt;br&gt; - The Message Logger primitives store message information in the Common database. &lt;br&gt; - The Common database is the default for the ESB logger mediation database, but you can use an external database for it.</td>
<td>- Enterprise Service Bus logger mediation primitive</td>
</tr>
<tr>
<td>Messaging engines data store</td>
<td>- The messaging engines data store is used by the message engines that are required for running WebSphere Process Server. &lt;br&gt; - Every messaging engine has its own schema.</td>
<td>- Service integration buses messages are stored here.</td>
</tr>
<tr>
<td>Selector and Business rules group database</td>
<td>- Stores the selector or business rule artifacts that are in the application.</td>
<td>- Selector &lt;br&gt; - Business rule</td>
</tr>
</tbody>
</table>

### 6.3 Business Process Choreographer Features

Business Process Choreographer gives you the ability to execute business processes within WebSphere Business Integration Server Foundation and WebSphere Process Server. In WebSphere Business Integration Server Foundation, one component within Business Process Choreographer handled the business process navigation and supporting staff activities.

In WebSphere Process Server, Business Process Choreographer was divided into two components, the Business Flow Manager and the Human Task Manager, which are enriched with new functionality. Basically, the focus on People Interaction in WebSphere Process Server was largely enhanced within
the new Human Task Manager. Figure 6-2 shows how the Business Flow Manager component handles the navigation of business processes.

![Diagram](image)

**Figure 6-2 Componentized Business Process Choreographer in WebSphere Process Server**

For detailed information about Business Process Choreographer, see the Web page at the following address:

http://www-128.ibm.com/developerworks/websphere/zones/was/wpc.html

### 6.3.1 Business Process Choreographer applications

The division of Business Process Choreographer into two subcomponents is obvious in the runtime environment. There is one component that handles the business process navigation and the staff processing in WebSphere Business Integration Server Foundation. In WebSphere Process Server, one component functions as a container for executing business processes, and another
component functions as a container for processing different kinds of human interaction. The following business applications are responsible for this division:

- **BPEContainer_nodename_servername**, which represents the Business Flow Manager in WebSphere Process Server
- **BPCEXplorer_nodename_servername**, which replaces the **BPEWebClient_nodename_servername** application in WebSphere Business Integration Server Foundation

The division into two subcomponents started in WebSphere Business Integration Server Foundation with the release of service pack 1 of WebSphere Business Integration Server Foundation V5.1.1. This release included the Human Task Manager. In WebSphere Process Server V6.2, **TaskContainer_nodename_servername** replaces the **HumanTaskManager_nodename_servername** application in WebSphere Business Integration Server Foundation.

In WebSphere Process Server V6.2, the following additional applications are worth mentioning:

- **Business Rules Manager** replaces the old deprecated Business Rules Beans of WebSphere Business Integration Server Foundation. Business Rules Manager is the main tool that business analysts use in browsing and modifying business rule values.
- **Business Process Choreographer Event Collector** is a J2EE application that must be configured before you can use the Business Process Choreographer Observer. This application is responsible for collecting all the Common Event Infrastructure events and storing them in a database.
- **Business Process Choreographer Explorer** is a J2EE application that is based on JavaServer™ Faces (JSF) technology. This application provides a user interface for administering business processes and handling human tasks.
- **Business Space** is a browser based graphical user interface that lets users customize content, through predefined widgets, from products within the WebSphere Business Process Management portfolio.

### 6.3.2 Business Process Choreographer application’s security roles

In WebSphere Business Integration Server Foundation and in WebSphere Process Server, **security roles** and **runAs roles** are associated with the Business Process Choreographer applications. The same roles apply in both systems, but WebSphere Process Server introduces additional and new roles.
Table 6-4 provides a general overview of the roles that are available in Business Process Choreographer with WebSphere Business Integration Server Foundation and WebSphere Process Server.

Table 6-4  Security roles in Business Process Choreographer in both products

<table>
<thead>
<tr>
<th>BPC application</th>
<th>Role</th>
<th>In WebSphere Business Integration Server Foundation</th>
<th>In WebSphere Process Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEContainer</td>
<td>▶ BPESystemAdministrator</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ BPESystemMonitor</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ WebClientUser</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ JMSAPIUser</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ BPEAPIUser</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BPEWebClient / BPCEXplorer</td>
<td>▶ WebClientUser</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TaskContainer</td>
<td>▶ TaskSystemAdministrator</td>
<td>X (starting with V5.1.1.1)</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ TaskSystemMonitor</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ EscalationUser</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>▶ TaskAPIUser</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

All of the roles that were available in WebSphere Business Integration Server Foundation are also available in WebSphere Process Server. When you migrate to WebSphere Process Server, the security must take the new roles into account to assign the users to these roles. For more details about the roles that are available in WebSphere Process Server V6.2, see “Instance-based authorization roles for business processes and activities” in the WebSphere Process Server Information Center at the following address:


In addition, you must define an authorization alias for Business Process Choreographer to enable the business process and human task containers to access the Business Processor Choreographer bus. These applications must have access to the bus to run properly.

6.3.3 People Directory Provider configuration

With WebSphere Process Server V6.1, the Staff Support Service and staff plug-in were re-named to the People Directory Provider, which was carried
through to WebSphere Process Server V6.2. The People Directory Provider introduced such features as the Virtual Member Manager and substitution.

With Business Process Choreographer, you can separate the logic of your business processes and human tasks from the staff resolution. You can resolve staff queries by using a staff plug-in that is specific to the directory service, as shown in Figure 6-3

![Figure 6-3 Staff resolution and People Directory provider plug-ins](image)

In general, the staff resolution architecture did not change between WebSphere Business Integration Server Foundation and WebSphere Process Server. Each of the staff plug-ins is associated with at least one configuration. In particular, a configuration specifies an Extensible Stylesheet Language Transformation (XSLT) file that performs the mapping between staff verbs and staff queries that are specific to the staff repository.

The Staff Support Service provides a default set of staff query verbs that you can use for standard staff queries. The list of these default staff query verbs is
extended. WebSphere Process Server includes the following additional predefined staff verbs:

- *Users by user ID without named users*, which assigns users given their user ID, except the ones that are explicitly excluded.
- *Group members without named users*, which assigns all users of a group, except the ones that are explicitly excluded.
- *Group members without filtered users*, which assigns all users of a group, except the ones that are found by using a search filter.
- *Group*, which assigns group members, but as a group work item, and supports WebSphere security dynamic groups.

Already in WebSphere Business Integration Server Foundation, you can customize verb sets and staff queries to better match your needs or your staff directory schema. To customize the verb set, create the new verb in the XML verb set and its corresponding mapping template in the XSLT file. The stylesheet requires a template to take the new incoming parameterized XML verb and transform it into the language that is supported by the specialized staff resolution plug-in. This new transformation file is used at deployment time to transform the query.

Usually, customized transformation files are used to support new verb sets or to suit the company’s LDAP schema. These customized files can be moved from the WebSphere Business Integration Server Foundation environment to the new WebSphere Process Server environment. However, you must extensively test that they are functioning properly in the new environment.

The basic aspects of using the staff service are described in the WebSphere Process Server Information Center:

```
```

### 6.3.4 Business Rule Beans

Business Rule Beans are a deprecated feature in WebSphere Process Server. To understand how to migrate Business Rule Beans, see 8.6.3, “Unsupported artifacts” on page 189.
6.3.5 Extended messaging

The extended messaging (CMM) feature of WebSphere Business Integration Server Foundation is deprecated in WebSphere Process Server. To understand how this feature should be migrated, see 8.6.3, “Unsupported artifacts” on page 189.

6.3.6 Common Event Infrastructure

The Common Event Infrastructure was introduced as a technical preview in WebSphere Business Integration Server Foundation V5.1 and received a supported component with V5.1.1. This framework provides capabilities to publish Common Base Events from different event sources and provides them to different event consumer applications.

Figure 6-4 illustrates the capabilities of Common Event Infrastructure.

![Figure 6-4 Capabilities of the Common Event Infrastructure](image)

Part of the Common Event Infrastructure is also a relational database, where emitted events are stored. In WebSphere Business Integration Server Foundation, you can use the Common Event Infrastructure API and the ECSEmitter API to emit custom events. Further, built-in events for business process related activities can be enabled for emission by the Business Process Choreographer.

The Common Event Infrastructure supports custom event consumer applications. Events can be consumed synchronously by using an EJB interface.
that the CEI provides or asynchronously using JMS messaging or publish/subscribe.

Figure 6-5 illustrates the architecture of the Common Event Infrastructure.

![Figure 6-5  Architecture of the Common Event Infrastructure](image)

The Common Event Infrastructure is also a part of WebSphere Process Server. The APIs of the Common Event Infrastructure remain the same. Therefore, applications that used these APIs in WebSphere Business Integration Server Foundation do not need to change during a migration to WebSphere Process Server.

It is not possible to preserve instance data from WebSphere Business Integration Server Foundation. The content of the WebSphere Business Integration Server Foundation Common Event Infrastructure database cannot be moved to a WebSphere Process Server Common Event Infrastructure database. It is also not possible to preserve the configuration data for the Common Event Infrastructure setup.
6.4 Administration in WebSphere Process Server using a console and scripts

In this section, we help to provide a basic understanding of the administrative features within WebSphere Application Server V6.2. All WebSphere Process Server administrative and management functionality is built on top of WebSphere Application Server.

The capabilities of managing the Process Choreographer Containers in WebSphere Process Server are enhanced. You can perform these administrative tasks by using either the Integrated Solutions Console or the administrative Jython scripts. Using the Jython scripts within the administration of the WebSphere Process Server offers some advantages:

▶ It is an easily repeatable and auditable process.
▶ Scripts are not prone to human error.
▶ Scripts perform the tasks quicker than using the Integrated Solutions Console.

There is greater control over the security and running of scripts, as opposed to using the Integrated Solutions Console.

Within the WebSphere Process Server administration console (Figure 6-6 on page 109), under Application servers → server1 → Business Integration, you see the administrative functions to manage the runtime and configuration environment for the task container and the business process container. For more details about the administration of WebSphere Process Server, see “Administering WebSphere Process Server” in the WebSphere Process Server Information Center at the following address:

When you make changes in the runtime configuration, you do not have to restart a server to make the changes take effect. After restarting the server, the runtime changes are discarded, unless they are also saved to a configuration.

**Servers and clusters**

Administering servers and clusters within WebSphere Process Server is the same as administering these components in WebSphere Application Server. You can perform this functionality by using either the Integrated Solutions Console or Jython scripts. The preferred approach is to use scripts to do the server management because it is quicker and removes the human error component. See the following topics for further information:

- “Administering servers and clusters” in the WebSphere Process Server Information Center


- “Scripting the application serving environment (wsadmin)” in the WebSphere Application Server Information Center

**Deployment environments**

The majority of the administrative functionality in WebSphere Process Server for deployment environments is the same as for administering servers and clusters. However, there is an exception in regard to the export of the deployment environment definitions. The administrative tasks are as follow:

- Stopping and restarting the Deployment manager
- Stopping and restarting cluster members
- Exporting deployment environment definitions using the administrative console
  

- Importing deployment environment definitions using the administrative console
  

- Exporting deployment environment definitions using the command line
  

- Importing deployment environment definitions using the command line
  

For more information, see “Administering deployment environments” in the WebSphere Process Server Information Center at the following address:


**Business process container and Human Task Manager**

WebSphere Process Server offers the following capabilities for administering the business process container and Human Task Manager:

- Business process container installation wizard
- Human Task Manager installation wizard
- Query and replay of failed messages that are in the retention or hold queue
- Change of the retry and retention queue message limits

Business process container messages are stored in the retention queue if a temporary error condition, such as a database deadlock or a connection failure, is detected. Messages from the retention queue are automatically fed...
back into normal processing. If the messages count, on the retention queue, reaches the retention queue message limit, the processing continues in quiesce mode. For more information, see “Querying and replaying failed messages, using the administrative console” in the WebSphere Process Server Information Center at the following address:


After the maximum number of retries, failed messages are put into the hold queue where an administrator can replay them.

Enablement or disablement of event emission to the Common Event Infrastructure or the AuditLog for all business processes that are running in the container.

Scripts are provided for the following administrative tasks. The install_root that is specified in the scripts path refers to the WebSphere Process Server home directory:

- The bpeconfig.jacl script, which can be used to configure both the business process container and the Human Task Manager components of the Business Process Choreographer.

  install_root/ProcessChoreographer/config/bpeconfig.jacl

For a complete list of the options and switches that are available for the bpeconfig.jacl configuration script, see “Using the bpeconfig.jacl script to configure Business Process Choreographer” in the WebSphere Process Server Information Center at the following address:


- Deleting audit log entries within the Business Flow Manager.

  install_root/bin/wsadmin -f
  install_root/ProcessChoreographer/admin/deleteAuditLog.py -server server_name [-profileName profileName] [options]

For a complete list of the options and switches that are available for this script, see “Deleting audit log entries, using administrative scripts” in the WebSphere Process Server Information Center at the following address:


- Deleting process templates that are no longer used.

  This action is performed when templates and all objects that belong to them that are no longer referenced within any application in the WebSphere configuration repository must be removed.
**install_root\bin\wsadmin -f**

**install_root\ProcessChoreographer\admin\deleteInvalidProcessTemplate.py** -server server_name -templateName templateName -validFrom validFromString [-profileName profileName]

For a complete list of the options and switches that are available for this script, see “Deleting process templates that are no longer valid” in the WebSphere Process Server Information Center at the following address:


**Deleting human task templates that are no longer used from the Business Process Choreographer database**

**install_root\bin\wsadmin -f**

**install_root\ProcessChoreographer\admin\deleteInvalidTaskTemplate.py**
-server server_name -templateName templateName -validFrom validFromString -nameSpace nameSpace [-profileName profileName]

For a complete list of the options and switches that are available for this script, see “Deleting human task templates that are no longer valid” in the WebSphere Process Server Information Center at the following address:


**Deleting completed process instances that have reached an end state of finished, terminated, or failed**

**install_root\bin\wsadmin -f**

**install_root\ProcessChoreographer\admin\deleteCompletedProcessInstances.py**

([-node nodeName] -server server_name) | (-cluster cluster_name)
(-all | -finished | -terminated | -failed)
([-templateName templateName [-validFrom timestamp]]
[-startedBy userID]
[-completedBefore timestamp]
[-profileName profileName]

For a complete list of the options and switches that are available for this script, see “Deleting completed process instances” in the WebSphere Process Server Information Center at the following address:

Deleting data from the observer database for process instances that match specified conditions
Deleting this data can improve report generation performance.

```
install_root\bin\wsadmin -f
install_root\ProcessChoreographer\admin\observerDeleteProcessInstanceData.py
    ( [-node node_name] -server server_name) | (-cluster cluster_name )
    [ -profileName profile_name ][ -dataSource dataSource_JNDI_name ]
    [ -dbSchemaName dbSchemaName] (-deletedBefore timestamp|(-templateName template_name -validFrom timestamp )|(-force
    [-templateName template_name -validFrom timestamp]
        -state state -reachedBefore timestamp )
)
```

For a complete list of the options and switches that are available for this script, see “Deleting data from the reporting database” in the WebSphere Process Server Information Center at the following address:


Querying and replaying failed messages script
This script helps to determine whether any failed messages for business processes or human tasks exist, and if so, it tries to reprocess them.

```
install_root\bin\wsadmin -f
install_root\ProcessChoreographer\admin\queryNumberOfFailedMessages.py -cluster cluster_name [ -bfm | -htm ] [-profileName profileName]
```

For a complete list of the options and switches that are available for this script, see “Querying and replaying failed messages, using administrative scripts” in the WebSphere Process Server Information Center at the following address:

Refreshing people query results, because the results that are returned are generally static

```
install_root\bin\wsadmin -f
install_root\ProcessChoreographer\admin\refreshStaffQuery.py -server server_name [-processTemplate templateName | (-taskTemplate templateName [-nameSpace nameSpace]) | -userlist username{,username}...][-profileName profileName]
```

For a complete list of the options and switches that are available for this script, see “Refreshing people query results, using administrative scripts” in the WebSphere Process Server Information Center at the following address:


Removing unused people query results

```
install_root\bin\wsadmin -f
install_root\ProcessChoreographer\admin\cleanupUnusedStaffQueryInstances.py -server server_name [-profileName profileName]
```

Further, on an application basis, you can start and stop process templates.

For more detailed requirements about using this script to manage the business process container in the WebSphere Process Server environment, see “Using scripts to administer Business Process Choreographer” in the WebSphere Process Server Information Center at the following address:


### Common Event Infrastructure

In this section, we present tasks that explain the mechanisms that are used when administering the Common Event Infrastructure at run time. Again, these tasks can be performed either through the administrative console or the `wsadmin` scripting interface. You can perform the following tasks:

- Enabling and disabling the event service through the console
- Creating an event emitter factory by using the console
- Creating an event group by using the console
- Creating an event filter by using the console

When enabling the Common Event Infrastructure components by using scripting, you must initialize the `wsadmin` scripting environment so that you can either type the commands interactively into the console or enter the commands into a script.
All of the following examples are for scripts in the Jython language and are for the interactive mode.

- **Enabling the event service through scripting:**
  - To start the wsadmin tools, enter the following script:
    
    ```bash
    install_root\bin\wsadmin
    ```
  - To enable the event service at the server scope, enter the following scripts:
    
    ```bash
    AdminTask.enableEventService('[ -nodeName node1 -serverName server1 ]')
    ```

  Alternatively, for the cluster environment:
  - To start the wsadmin tools, enter the following scripts:
    
    ```bash
    install_root\bin\wsadmin
    ```
  - To enable the event service at the cluster scope, enter the following scripts:
    
    ```bash
    AdminTask.enableEventService('[ -clusterName cluster1 ]')
    ```

For a complete list of the options and switches that are available for this script, see “Enabling the event service using scripting” in the WebSphere Process Server Information Center at the following address:


- **Disabling the event service through scripting:**
  - To start the wsadmin tools, enter the following script:
    
    ```bash
    install_root\bin\wsadmin
    ```
  - To disable the event service at the server scope, enter the following scripts:
    
    ```bash
    AdminTask.disableEventService('[ -nodeName node1 -serverName server1 ]')
    ```

  Alternatively, for the cluster environment:
  - To start the wsadmin tools, enter the following script:
    
    ```bash
    install_root\bin\wsadmin
    ```
  - To disable the event service at the cluster scope, enter the following scripts:
    
    ```bash
    AdminTask.disableEventService('[ -clusterName cluster1 ]')
    ```
For a complete list of the options and switches that are available for this script, see “Disabling the event service using scripting” in the WebSphere Process Server Information Center at the following address:


- Creating an emitter factory with scripting
  You can only do this task by using a script. Use Jython because JACL is deprecated. For an example of the emitter factory creation script, see “Creating an emitter factory with scripting” in the WebSphere Process Server Information Center at the following address:


- Creating an event group using scripting
  You can only do this task by using a script. For an example of the event group creation script, see “Creating an event group with scripting” in the WebSphere Process Server Information Center at the following address:


- Creating an event filter with scripting
  You can only do this task by using a script. For an example of the event filter script, see “Creating an event filter with scripting” in the WebSphere Process Server Information Center at the following address:


- Purging events from the event database
  You perform this task to provide maintenance to the events database to prevent a degrade in performance with redundant data. For more information, see “Purging events from the event database” in the WebSphere Process Server Information Center at the following address:

For additional information regarding administering the CEI, see “Administering Common Event Infrastructure” in the WebSphere Process Server Information Center at the following address:


**Service components**

Service components can be broken into three distinct sections:

- Business processes and human tasks
- Business state machines
- Business rules and selectors

Each of these sections has their own administrative functions, which we explain in the following sections.

**Business processes and human tasks**

The administration of business processes and human tasks entails the following functions:

- Administering process templates and process instances, which can be done either through the console or by using scripts
- Administering task templates and task instances, which again can be done by using scripts or the Integrated Solutions Console
- Reporting on business processes and activities

For more information about administering business processes and human tasks, see “Administering business processes and human tasks” in the WebSphere Process Server Information Center at the following address:


**Business state machines**

Administering business state machines falls into two different categories, of which the administrative tasks can only be performed through the console:

- Finding business state machines, views correlations, and set properties, which enables you to determine the business state machine instance
- Viewing display states of business state machines, with which administrators can administer or debug business state machine instances

For more information, see “Administering business state machines” in the WebSphere Process Server Information Center at the following address:
Business rules and selectors

Business rules control the behavior of a business practice. Selectors provide an interface that can change business process results based on criteria.

Business rules and selectors can only be administered by using the Integrated Solutions Console. For more information, see “Administering business rules and selectors” in the WebSphere Process Server Information Center at the following address:

Migration options

In this chapter, we include the following sections:

- “Migration paths” on page 120
- “Migration tools” on page 124
- “Migration approaches” on page 150
7.1 Migration paths

For all the similarities in goals and functionality between WebSphere Business Integration Server Foundation V5.1 and WebSphere Process Server 6.2, the product architecture and programming model differences between both products are substantial. The runtime migration (upgrade path) is not provided in WebSphere Process Server 6.x; therefore, the source artifact migration path is the only option for migrating WebSphere Business Integration Server Foundation V5.1 applications.

More information: Many features that are available in WebSphere Business Integration Server Foundation 5.1 moved to the base version of WebSphere Application Server 6.x. For details about migrating those features, see “Programming model extension migration” in the WebSphere Application Server Information Center at:


Source artifacts can be migrated from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer using the workspace migration wizard, which involves migrating the application to the new WebSphere Integration Developer programming model so that new functionality and features can be used. The application can then be redeployed and installed to the WebSphere Process Server.

Many different source component artifacts exist in the WebSphere Business Integration Server Foundation application. Business process choreography is the core component. Because of the architectural changes, some of the artifacts are replaced by newer Service Component Architecture artifacts, and others might not be supported any more, for example, the Business Rule Beans are replaced by Business Rule component. For more detailed migration consideration of other WebSphere Business Integration Server Foundation components, refer 8.6, “Additional migration considerations” on page 185.

In this section, we focus on migration paths for business processes.

Based on the type of business process that is used and the way business artifacts were initially developed, there are the following migration paths for migrating business processes and staff artifacts to WebSphere Process Server:

- Migrating business processes that are described in Business Process Execution Language (BPEL)
7. Migrating business processes described in BPEL

For the business processes described in BPEL, WebSphere Business Integration Server Foundation V5.1 and WebSphere Process Server V6.1 have the following major differences:

- **Programming model**
  In WebSphere Business Integration Server Foundation, the programming model is Web Services Invocation Framework, which is a simple Java API for invoking Web services.

  In WebSphere Process Server, the programming model is a Service Component Architecture (SCA), which is a set of specifications that describe a model for building applications and systems using a service-oriented architecture (SOA).

- **Level of support for Business Process Execution Language for Web Services (BPEL4WS)1.1**
  In WebSphere Process Server, we extended the support of BPEL4WS 1.1 with major capabilities for the upcoming Web Services Business Process Execution Language (WS-BPEL) version 2.0 specification.

The WebSphere Integration Developer migration wizard enables the migration of service projects that are developed in WebSphere Studio Application Developer Integration Edition to the WebSphere Integration Developer programming model automatically.

Figure 7-1 on page 122 shows the migration path for the BPEL processes.
7.1.2 Migrating business processes described in Flow Definition Markup Language

In WebSphere Studio Application Developer Integration Edition, processes are described by using either BPEL or a proprietary format known as *Flow Definition Markup Language* (FDML), which is deprecated but is still supported in WebSphere Business Integration Server Foundation 5.1.1. It is no longer supported in WebSphere Process Server.

When you attempt to migrate FDML processes to the WebSphere Process Server, an additional migration step is necessary. The WebSphere Integration Developer migration wizard only accepts the BPEL process. Therefore, you must use the following two-step approach:

1. Migrate the FDML process to a WebSphere Business Integration Server Foundation-compliant BPEL process.
2. Migrate the WebSphere Business Integration Server Foundation-compliant BPEL process to the WebSphere Process Server.
WebSphere Studio Application Developer Integration Edition V5.1 provides the automatic migration tool to migrate the FDML process to the BPEL process. Figure 7-2 illustrates the two-step approach for migrating the FDML process.

![Diagram](image)

Figure 7-2  Two-step approach for migrating the FDML process

### 7.1.3 Migrating business processes mainly defined in WebSphere Business Integration Modeler

If your project uses WebSphere Business Integration Modeler extensively, the preference is to migrate from WebSphere Business Integration Modeler 5.1 to WebSphere Business Modeler 6.2. WebSphere Business Modeler, which can automatically migrate the old version of a model during the model import. In this case, migrate to WebSphere Business Modeler Version 6.2. You just import the old version of a model or workspace into WebSphere Business Modeler V6.2, and from this model, export it to WebSphere Integration Developer. After importing the model into WebSphere Integration Developer, manually integrate the services and bindings in the new SCA platform.

This path is useful in the following scenarios:

- The project must use the modeling features of WebSphere Business Modeler, for example, when it is used to improve the model, to run simulations, or to define matrix.
- Many activities of the project are human-task activities.
- The project wants to explore the monitoring functionality of WebSphere Business Monitor V6.2.
Figure 7-3 illustrates the migration paths from a WebSphere Business Modeler V5 project.

![Diagram of migration paths](image)

**Figure 7-3   Migration path from a WebSphere Business Integration Modeler V5 project**

WebSphere Business Modeler V6.2 provides several options to migrate WebSphere Business Modeler V5 projects. For more information, see 7.2.2, “WebSphere Business Modeler” on page 142.

### 7.2 Migration tools

In this section, we introduce tools for migrating from WebSphere Business Integration Server Foundation to WebSphere Process Server. We also include a general overview of WebSphere Integration Developer.

#### 7.2.1 WebSphere Integration Developer

IBM WebSphere Integration Developer is a visual software development environment for creating and assembling SOA business applications. It enables WebSphere Integration Developer to quickly discover and assemble composite applications, maximizing the reuse of all types of business services. It also helps to ensure that the processes integrate Web services, enterprise applications, human tasks, and other service components into effective SOA-based business solutions.

In this section, we introduce the general capabilities of WebSphere Integration Developer and include general information about the migration wizard.
For detailed information about how to use the WebSphere Integration Developer
migration wizard, see 8.4.2, “Using the migration wizard” on page 176.

WebSphere Integration Developer is built on the Rational Software Development
Platform, which is based on Eclipse 3.2 technology. It provides full integration
with other IBM tools and products. WebSphere Integration Developer is primary
SOA integration tool for IBM.

**Eclipse:** For more information about Eclipse and tutorials, visit the following
Web site, and explore the Getting Started pages:

[http://www.eclipse.org](http://www.eclipse.org)

In this section, we introduce the basic terms and concepts of WebSphere
Integration Developer that are used in migrating the WebSphere Interchange
Server V4.3 source artifacts.

We discuss:

- Business Integration perspective and views
- Module and library
- Editors
- Integration test client
- Integration debugger
- Migration wizard

**Business Integration perspective and views**

The Business Integration perspective provides simple, uncluttered views of
essential resources, so that you can model and build business solutions. It filters
out unnecessary details and unused tools. By default, when you launch the
product, the Business Integration perspective is opened. It is the primary
perspective in WebSphere Integration Developer and is where the majority of
business integration work is done. Figure 7-4 on page 126 shows the Business
Integration perspective.
This perspective has the following views, as numbered in Figure 7-4:

1. The *Business Integration view* provides a logical view of the key resources in each module, mediation module, and library.

2. The *Physical Resources view* shows all of the file-level resources from the modules and libraries in their natural form.

3. The *Editor pane* shows the resource editor when you open the resource from the Business Integration view.

4. The *References view* shows all of the artifacts that reference the selected artifact and all the artifacts that the selected artifact references.

5. The *Outline view* works with the resource that is opened in the editor pane and shows the navigation tree view or graphic overview.

6. The *Visual Snippets view* is used to compose customized behavior in the form of Java code.

7. The *Build Activities view* helps you manage builds. You can select build activities for both automatic and manual builds and invoke immediate manual builds that are independent of the build activity selections.

8. The *Properties view* shows the property information that is associated with selected artifacts from Business Integration view or Physical Resource view.
When an artifact is opened with an editor, you can modify properties of selected elements in the editor.

9. The Problems view helps you to debug errors that have error and warning messages.

10. The Server Logs view shows all the logs and messages for the server.

11. The Servers view can be used to create servers, start or stop a server, deploy an application, and test an application.

Module
When building integrated business solutions with WebSphere Integration Developer, you create modules, mediation modules, or libraries to contain the resources and code. A module represents a basic unit of deployment and encapsulates SCA resources (SCA modules), J2EE projects, Java projects, and dependent libraries. It is packaged in an EAR file and contains the following artifacts:

- SCA resources and module assembly
- J2EE projects
- Java projects
- Dependent libraries

By using the dependency editor, you can add dependent libraries, Java projects, and J2EE projects to a module and choose to deploy them with the module.

Library
A library project is another type of business integration project. Unlike the module project, a library project is not a deployable unit. The library project holds artifacts that can be shared between multiple modules. A library contains the following artifacts:

- Interfaces
- Business objects
- Business object maps
- Relationships

Before a module can use the resources from a library, you must add the library to the module dependency list by using the dependency editor. Figure 7-5 on page 128 shows the OrderProcessing module, which requires three libraries: ChargingLibrary, OrderLibrary, and ShippingLibrary.
A library cannot be deployed by itself. However, you can add a library to the module and select it to be deployed with the module, as Figure 7-5 shows. At runtime, libraries are not shared, but deployed, with the module that depends on it. Also, you can add library dependencies to another library. Figure 7-6 on page 129 shows how the module and library are deployed on the server.
During deployment, the shared library is copied to the server with the module. Each application has its own copy of the library. In this case, if the shared library is updated, all the modules that have a dependency with the library must be redeployed to get the updated library in place.

**Editors**

An editor is a tool to create and modify resources or artifacts. Depending on the type of artifacts that you are editing, the appropriate editor opens in the center or main pane of the workbench, for example, a Java editor opens when you double-click a Java source file, and business objects open in the Business Object editor.

**Assembly editor**

With the assembly editor of WebSphere Integration Developer, you can build applications by assembling the SCA components. When you open a module assembly with the assembly editor, you can visually compose the integrated...
application by adding components and connecting them with wires in the Editor pane. Figure 7-7 shows a module assembly that is opened in the Editor pane. The modeled application is also referred to as the *assembly diagram*. The Editor pane has a canvas where you create the assembly diagram.

![Figure 7-7  Assembly editor](image-url)
**Business object editor**

By using the business object editor, you can build and edit business objects, their attributes, and business graphs through a graphical interface, as shown in Figure 7-8. Use this editor to add, delete, reorder, and change the type of an attribute.

![Figure 7-8 Business object editor](image)
**Business process editor**

The business process editor is a graphical programming environment in which you can visually create and manipulate business processes. The business process editor is designed for ease of use and to facilitate your creativity. While designing your process, think of using this editor as an exercise in painting. The white space is called the *canvas*. The canvas where you create your process from the objects that you pull from the palette on the left and with reference to the objects in the tray on the right. Below all of this is an interactive properties area that changes to display pertinent details about the object that you currently have selected on the canvas or the tray.

Figure 7-9 illustrates a TravelBooking business process opened in the business process editor.
**Interface editor**

An interface consists of one or more operations and a binding style.

An operation is a description of an action implemented by the component. An operation can be a request-response type or a one-way type. The binding style specifies the protocol and data format of the operation.

WebSphere Integration Developer is a flexible development environment in which you can determine the best approach for application development. You can build interfaces before or after component implementation. When you take a top-down development approach, which begins with creating the interface and then adding the implementation, you launch the interface editor, provide a name for the interface, and add one or more operations to it. Inputs, outputs, and faults are added to each operation.

Figure 7-10 shows an interface that is created by the interface editor.

![Operations](image)

<table>
<thead>
<tr>
<th>Operations and their parameters</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getBalance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="1 Input(s)" /></td>
<td><img src="image" alt="" /></td>
<td>balance</td>
</tr>
<tr>
<td><img src="image" alt="1 Output(s)" /></td>
<td></td>
<td>approval</td>
</tr>
<tr>
<td><img src="image" alt="Fault(s)" /></td>
<td></td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>systemFailure</td>
</tr>
<tr>
<td><strong>getHistory</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="1 Input(s)" /></td>
<td></td>
<td>customerName</td>
</tr>
<tr>
<td><img src="image" alt="1 Output(s)" /></td>
<td></td>
<td>customerPastHistory</td>
</tr>
<tr>
<td><img src="image" alt="Fault(s)" /></td>
<td></td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>systemFailure</td>
</tr>
<tr>
<td><strong>updateCreditRating</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="1 Input(s)" /></td>
<td></td>
<td>currentRating</td>
</tr>
</tbody>
</table>

Figure 7-10   Interface editor

**Interface mapping editor**

An interface provides the means to interact with a component. It is independent of the internal implementation of the component. In WebSphere Integration Developer, you can use the interface mapping editor to resolve and reconcile the differences between the interfaces of interacting components. With the interface mapping editor, you can build and edit interface maps in a graphical mode.

Figure 7-11 on page 134 shows an interface map that is opened in the interface mapping editor.
Business object mapping editor

A business object map assigns values to target business objects based on values in source business objects. The business object mapping editor, as shown in Figure 7-12 on page 135, is a graphical interface with which you can build and edit business object maps and their attributes. By using this editor, you can define mapping of data between business objects.
Relationship editor

A relationship correlates at least two or more semantically equivalent business objects, which are represented in different physical formats. By using the relationship editor (Figure 7-13) in WebSphere Integration Developer, you can build and edit relationships and their attributes through a graphical interface.
**Visual snippet editor**

The visual snippet editor is a diagrammatic programming environment that you use to graphically create and manipulate Java code. This editor was designed for ease of use and to facilitate your creativity. While designing your visual snippet, think of using this editor as an exercise in painting. The white space is called the *canvas*, which is where you create your snippet from the objects that you pull from the palette to the left. Below the canvas is an interactive properties area that changes to display pertinent details about the object that you currently have selected on the canvas.

As shown in Figure 7-14, the visual snippet editor is divided into several distinct areas, each with its own individual use.

![Figure 7-14  Visual snippet editor](image)

This visual snippet editor has the following areas, as numbered in Figure 7-14:

1. The *Palette* is the shaded area to the left of the canvas that houses the objects that you drag to the canvas to build your snippet.
2. The *canvas* is the white empty area in the middle of the editor that you use it to assemble the activities to compose your visual snippet.

3. The *expression builder* is a miniature dialog that opens when you click an expression. It prompts you to graphically compose your expression.

4. The *tray* displays the inputs, outputs, and exceptions that are associated with your snippet.

5. The *properties area* displays properties that are relevant to the object that is currently selected on the canvas.

6. The *visual snippets view* shows a categorized view of all of the existing snippets that are available to you to use.

**Test suite editor**

The test suite editor is the designated tool for editing test suites. It features a rich user interface that enables you to easily create and manage your test suites, including test cases. The test suite editor closely resembles the test client, which helps you easily transition from using one tool to the other. Figure 7-15 on page 138 shows an example of the test suite editor.
Figure 7-15   Test suite editor
Integration test client
The integration test client is the designated tool for testing modules and components. The test client features a sophisticated user interface that enables you to easily manage and precisely control your tests. Figure 7-16 shows the integration test client.

![Integration test client](image)

*Figure 7-16  Integration test client*
Integration debugger

The integration debugger is the designated tool for debugging business integration components. The integration debugger features a sophisticated user interface with which you can easily and precisely perform component debugging. The integration debugger is primarily controlled from the debug perspective, which is shown in Figure 7-17.

This debug perspective has the following views, as numbered in Figure 7-17:

1. In the Debug view, you can control the execution of component instances and alter their state at run time through start, stop, and stepping control buttons.

2. By using the Integration Test Client, you can test your modules and components. You can also use the test client to automatically start the integration debugger and create a running instance of your component that you can debug.
3. You use *component editors* to set breakpoints on component elements or source code. Then you use the integration debugger to graphically step through instances of the components and work with their variables or messages.

4. The *Breakpoints view* displays the breakpoints that are set on component elements and the source code in all components.

5. The *Variables view* displays all variables, messages, and associated values for the stack frame that is selected in the Debug view.

### Migration wizard

Use the migration wizard tool to migrate WebSphere Business Integration Server Foundation service project source artifacts to WebSphere Integration Developer. Figure 7-18 shows the most important steps in the execution of this migration wizard.

---

**Figure 7-18  Artifact migration wizard**

---

The following numbered list corresponds with the numbers in Figure 7-18 to provide further explanation about what the migration wizard does:

1. The wizard creates a new business integration module and defines the class path according to the original class path.
2. It copies the source artifacts to the new module.

3. It migrates the business processes (BPEL files) from BPEL4WS to the new level that WebSphere Process Server supports, which is built on BPEL4WS Version 1.1 with major capabilities of the WS-BPEL Version 2.0 specification.

4. It defines the human task activities by using the new definition of .itel files and cleans the BPEL file of the <wpc:staff> tags.

5. It generates a monitoring .mon file for each BPEL process to preserve the default monitoring behavior from WebSphere Studio Application Developer Integration Edition.

6. It creates an SCA component for each BPEL process.

7. It creates imports and exports depending on the deployment options that are chosen in WebSphere Studio Application Developer Integration Edition.

8. It wires the BPEL component to its partner links (imports, exports, and Java components).

For more information about how to use the migration wizard, see 8.4.2, “Using the migration wizard” on page 176.

### 7.2.2 WebSphere Business Modeler

WebSphere Business Modeler streamlines the process of creating realistic business models. By using it, you can integrate the different aspects of a complex business process in one model. WebSphere Business Modeler provides sophisticated analysis tools with which you can evaluate both current and potential business processes. It also provides a direct link from business process modeling to the implementation of software services. WebSphere Business Modeler offers the following functions among others:

- Reuse any existing business analysis by importing models or definitions from other sources, which includes importing Microsoft Visio® data.

- Associate business processes with models of resources, data and information, organizations, and other processes to obtain a detailed, complete understanding of your process flow.

- Publish process models so that reviewers can view and comment on the models by using WebSphere Business Modeler Publishing Server.

- Communicate your analysis by predefined or custom reports that you can generate in Microsoft Word or PDF files.

- Update process models with team members anywhere in the world by using the versioning capabilities of WebSphere Business Modeler.
Simulate the behavior and performance of your business processes, analyze the simulations, and generate statistics to determine potential areas of process improvement.

Include, in the business process, information about business measures (key performance indicators and metrics).

Import information into your business models, including WSDL and XSD files.

Import statistics of the real behavior of the process by using WebSphere Business Monitor monitoring results.

Export your process models in formats, such as BPEL, WSDL, XSD, FDL, and UML, that other applications can use.

You can export the models from WebSphere Business Modeler as BPEL definitions to be imported in the WebSphere Integration Developer tool.

**Migrating modeler projects**

You can migrate the modeler projects from WebSphere Business Integration Modeler V5 by using the migration tool that is included in WebSphere Business Modeler V6.2. Because both modelers use an XML Metadata Interchange (XMI) repository, the migration is a mapping from the version 5.1.1 XMI repository to the version 6.2.

To perform the migration:

1. Make a backup copy of all projects in WebSphere Business Integration Modeler.

2. Install the latest fix pack that is available for WebSphere Business Integration Modeler 5.1.

3. Ensure that the model in WebSphere Business Integration Modeler is complete and error free. Validate each process model to migrate.

   Remember that some warning messages are normal in BPEL mode, but you must solve any errors in the error view. The error list is an XMI file that is in the project directory as `messages.xmi`.

4. Define how to migrate:
   - A complete workspace of V5
   - A set of modeler exported projects
   - Projects in a Concurrent Versions System (CVS) or a team repository

   The advantage of the workspace migration is that you can perform it in only one step for all of the projects.

5. (Optional) Export the WebSphere Business Modeler V6.2 project to WebSphere Integration Developer for completing the implementation.
Migrating a WebSphere Business Integration Modeler V5 workspace

To migrate a WebSphere Business Integration Modeler V5 workspace:

1. Start the WebSphere Business Modeler V6.2 product.
   a. Select the location of the WebSphere Business Integration Modeler V5 workspace as the workspace.

   **Tip:** If you select the **Use this as the default and do not ask again** option in WebSphere Business Modeler with an old workspace, change the default value after you start the product. To change the value, select **Window → Preferences**. In the Preferences window, select **Workbench → Startup and Shutdown**, and select the **Prompt for workspace on startup** option.

2. In the Migrate Workspace window (Figure 7-19), define a location path for the workspace backup. Optionally, you can specify to make the backup a compressed file. In this case, ensure that all of the files and directories use only English characters. Click **OK**.

![Figure 7-19   Defining the backup location in the Migrate Workspace window](image)

3. In the WebSphere Business Modeler – Workspace migration completed window (Figure 7-20 on page 145), click **OK**. This window shows a summary with a list of the projects that migrated successfully and those that migrated unsuccessfully.
4. Review and correct eventual problems in the projects. You can manually adjust some configuration artifacts to prepare them to export to WebSphere Integration Developer.

**Migrating WebSphere Business Integration Modeler V5 exported projects**

To migrate WebSphere Business Integration Modeler V5 exported projects:

1. Export the model by using WebSphere Business Integration Modeler as a WebSphere Business Integration Modeler V5 Project. It generates a compressed file.

2. Open the importing wizard by selecting **File → Import → WebSphere Business Modeler Import**.
3. In the Import – Select type window (Figure 7-21), select the format **WebSphere Business Modeler Project**. Click **Next**.

![Select format in the WebSphere Business Modeler Import window](image-url)
4. In the Import – Source and destination window (Figure 7-22), for Source directory, click **Browse** to browse for the folder that contains the compressed file. Select the project file. For Target project, choose an empty target project, or click **New** to create a new project. Click **Finish**.

![Image of WebSphere Business Modeler Import window](image)

**Figure 7-22  Select project in the WebSphere Business Modeler Import window**

The Import wizard recognizes that there is a migration from an earlier version of WebSphere Business Modeler and shows a version detection message.

5. When the importing process is completed, the wizard’s window shows the finished message. If there are any warning or error messages, click **Details**.
Migrating WebSphere Business Integration Modeler V5 projects by using CVS

To migrate WebSphere Business Integration Modeler V5 projects by using CVS:

1. Right-click the project tree view, and select **Team → Check Out Project**.
2. Browse the project tree, and select the WebSphere Business Integration Modeler V5 project. Click **Finish**.

   WebSphere Business Modeler V6 makes a copy of the original project from the CVS repository and migrates that copy.

3. Review the generated project, and make corrections based on any error messages that are displayed.

Exporting WebSphere Business Modeler V6.1 projects to WebSphere Integration Developer

Export the model in WebSphere Business Modeler V6.1 by using the project interchange file, and import the exported file in to WebSphere Integration Developer:

1. Open the WebSphere Integration Developer import wizard.
2. In the Import Projects window (Figure 7-23 on page 149), select the model project, and click **Finish**.
Figure 7-23  Importing the project from WebSphere Business Modeler V6.2

WebSphere Integration Developer shows the project with error and warning messages because the model does not include complete definitions and implementation of the activities.

3. Implement the activities, define the bindings, and finish the project to deploy it in WebSphere Process Server.

### 7.2.3 Command-line interface and scripts

When migrating the WebSphere Business Integration Server Foundation source artifacts to WebSphere Process Server, you must complete three fundamental tasks:

1. Prepare source artifacts for migration, which you might need to perform in WebSphere Studio Application Developer Integration Edition.
2. Migrate the WebSphere Studio Application Developer Integration Edition V5.1 service project and related artifacts to the WebSphere Integration Developer Business Integration Module project.

3. Use WebSphere Integration Developer to manually complete the migration.

   This step involves fixing any Java code that cannot be automatically migrated and verifying the wiring of the migrated artifacts.

Steps 1 and 3 require a manual process that cannot be automated. However, for step 2, WebSphere Integration Developer provides a command script utility, called WSADIEWorkspaceMigration. While the WSADIEWorkspaceMigration command script accomplishes the same tasks as the migration wizard (outlined in Figure 7-18 on page 141), it has solid benefits over the GUI. The migration wizard is using the interactive graphic user interface, and you must set the options in the wizard manually in order for the task to complete. With command-line scripts, you can create batch jobs to migrate multiple workspace at once. Because all the response options are in the scripts, you can run the migration batch job on the weekend or at night when the system resource is not busy.

For more details about how to use the WSADIEWorkspaceMigration command script, and to see a sample script for migrating the service project, see 8.5, “Migrating a workspace using a command-line interface and scripts” on page 182.

### 7.3 Migration approaches

A business solution that is running on WebSphere Business Integration Server Foundation usually consists of multiple applications. Before you migrate an application, consider that there might be dependencies between these applications. In 7.3.1, “Migrating with dependencies between applications” on page 151, we discuss details about evaluating different migration approaches for scenarios with different dependencies between applications.

If long running processes are used in applications of WebSphere Business Integration Server Foundation, you must apply additional considerations. There are three approaches for handling migration of WebSphere Business Integration Server Foundation projects with long running processes:

- Coexistence approach
- Move-at-once approach
- Advanced move-at-once approach
These approaches might be mixed based on the project size and complexity. To choose the right approach, you must thoroughly assess the existing long running process instance environment. Based on that assessment, as described in the next section, consider one approach or a mix of several approaches.

### 7.3.1 Migrating with dependencies between applications

Depending on the complexity of the application to be migrated, additional considerations might be needed. For the large complex applications, usually we separate the whole application into different groups and migrate them in separated stages over a period of time. In this case, determining the dependencies between the applications is very important and is the key factor for the success of the whole migration project. The general rules-of-thumb for the multi-stage migration are:

- **Applications are independent of each other.**
  
  This is the easiest case for a migration. No special considerations apply here.

- **Applications do not interact with each other but interact with common custom data.**
  
  When several applications share information in a common database, they can be moved to the new environment one at a time. The migrated applications use the same data store like the remaining applications on the WebSphere Business Integration Server Foundation system.

- **Applications invoke one another, with no circular dependencies, as shown in Figure 7-24.**

![Figure 7-24](image-url) Applications invoking one another, with no circular dependencies

In this case, the common approach is to migrate one application after the other. The WebSphere Business Integration Server Foundation and the WebSphere Process Server environment can coexist until all of the applications are migrated.
There are two possibilities for starting the migration of that system:

– Start migrating application A, as illustrated in Figure 7-25. Notice that the numbers correspond to the numbers in the illustration:

  i. Leave B and C on the old system until they are ready for migration (1).

  ii. When application B is ready to run on the new environment, you can make a small change to application A’ to direct the invocation to the migrated application B’ (2), which allows application B to be phased out on WebSphere Business Integration Server Foundation.

  iii. Change B’ from call C (3) to C’ (4).

– Start migrating application C, as illustrated in Figure 7-26 on page 153. When ready, change B to call C’ (1), followed by B. Change the invocation in A to B’ (2), and change the client to call A’ (3).

  Compared with the previous approach, this approach requires changes to the WebSphere Business Integration Server Foundation applications to direct their calls to the migrated applications in the new environment.
Chapter 7. Migration options

Figure 7-26  Migration starting last first

We recommend the first approach, as illustrated in Figure 7-25 on page 152, because WebSphere Process Server does the migration work by directing the calls to the other environment. Because this is the target platform and it is used in future development, it is better to spend time and resources by using this product whenever possible, instead of developing on the old WebSphere Business Integration Server Foundation platform. This development is necessary in the second approach.

If long running processes are calling one another, we recommend that you use a migration approach that starts at the last called process through the first process in the call chain. This recommendation is opposite the same case for other applications, which is why coexistence is not an issue in the other case. Starting with the last processes in the call chain minimizes the period of coexistence that is needed for WebSphere Business Integration Server Foundation and WebSphere Process Server. However, it requires changes to the applications on WebSphere Business Integration Server Foundation so that the newly migrated processes are invoked.

- Circular dependencies exist between applications.

Having circular dependencies between applications is the most complex migration scenario. Depending on the complexity of the dependencies, evaluate which of the following options fits best as a migration approach:

- Migrating individual applications and modifying applications that are not yet migrated to invoke the newly migrated application
This approach combines the two approaches that we described earlier. You must make additional changes to the migrated applications as you keep moving applications to the WebSphere Process Server. This option implies a large maintenance and planning effort. Also, development on both platforms is required.

– Migrating all applications at one time

The advantage here is that you can do the whole project cycle, in terms of development and test, separately and completely on the new platform. However, this option might be risky because you move all functions of the business application at one time.

### 7.3.2 Considerations for long running processes

Long running processes are interruptible processes that can include human interaction and execution in parallel threads. They persist their state in the Process Choreographer database (BPEDB) and the Process Choreographer messaging resources. Long running processes can last from seconds to years, depending on the business model and requirements.

You cannot migrate the Process Choreographer database content to the WebSphere Process Server because the database schema changed between the WebSphere Business Integration Server Foundation and the WebSphere Process Server. Also, you cannot move the internal messages that are used for navigation.

Therefore, at the time of migration, you cannot automatically transition unfinished process instances in WebSphere Business Integration Server Foundation to WebSphere Process Server. You must take special actions for these processes. You must consider these specialities, with regard to long running processes, during migration planning.

**Assessing process instances in WebSphere Business Integration Server Foundation environment**

To choose the best migration approach for long running process instances in the WebSphere Business Integration Server Foundation environment, determine their average and maximum duration. By doing this, you can gain an overview of the environment to be migrated.

If available, capture the following values for every process that needs to be migrated:

- Number of completed process instances
- Average duration of a completed process instance
- Maximum duration of a completed process instance
You can use either of the following ways to determine the duration of process instances:

- The business operations know the duration to expect for a process instance of a certain type. Also custom logging of values might be helpful to determine the duration of a process in the application.

- A technical evaluation of the data is found in the Process Choreographer database. Use the BPEDB to query data that is related to process instances and obtain their duration. There are multiple ways to obtain data from there.

If you do not select the “Automatically delete process after completion” option, data can be obtained for finished processes from the BPEDB.

If the data is not obtained for finished processes, then it can still be observed, when running instances are started. If the number of started process instances per a certain time is constant, the start time of running instances in the BPEDB can provide good guidance. Repeat that query for the start time of running instances in reasonable time frames for the following reasons:

- Get an impression on the number of process instances that are completing in these time frames.

- Get confidence that the obtained numbers are applicable.

The AuditLog view within the BPEDB is also a good source of information about the process duration. Look for processes that started and finished events. You can also calculate the average and maximum duration based on these results. You must enable auditing for the process to use this data. For more information about using the AuditLog, see “Enhanced audit log data analysis and query for BPEL processes with Process Choreographer 5.1” on the Web at the following address:


We recommend that you use both the business and technical approaches to help you obtain a complete picture of the old environment, especially if discrepancies are observed between the results of the two approaches.

By having the necessary values, you know the following information:

- How long the process instances typically take until they are completed

You can determine how much time is needed to finish the running process instances on the WebSphere Business Integration Server Foundation environment.
A reasonable time after which you can terminate a process instance

As might be the case, some processes take longer than the expected maximum time. Create a plan for what to do in this case, for example, consider a bank that is running a process for loan approvals. There is a maximum time for that process. Processes that take longer than that might not be valid any more because the loan approval applicant does not wait forever to get a loan. Therefore, it might be possible to terminate these process instances.

In some cases, processing a business operation requires multiple processes, for example, the business operation for loan approval consists of three parts, each represented by a business process. Only the completion of these three process instances completes the business operation.

In this case, the duration of the business operation must also be calculated, which is more difficult to determine than the preceding information. One approach to follow is to get the durations for single processes, as we previously described, and calculate the minimum, average, and maximum duration for the whole business operation based on the values of the single business process instances.

Figure 7-27 shows the process durations.

Coexistence approach

Coexistence is the means of running both environments, WebSphere Business Integration Server Foundation and WebSphere Process Server, for a defined period of time, which ensures that instances of long-running business processes
can finish on the old environment while new processes are started on the new
environment. This approach requires that you adopt your client to manage calls
to both systems for that time.

By using the coexistence approach, as shown in Figure 7-28 on page 158, you
can build up the WebSphere Process Server system and deploy and test your
migrated application. Then you can test a new version of the client that talks to
both systems. At some point, you can go live with both systems by changing the
client to direct new process starts to the WebSphere Process Server while
handling requests for the WebSphere Business Integration Server Foundation
process instances.

The disadvantage of this approach is that additional hardware might be needed
to run the new environment and additional administrative and maintenance
efforts during the time that both environments are running in parallel.

To validate whether the coexistence approach is the best way to go, calculate
how long the WebSphere Business Integration Server Foundation processes
take until completion, which we describe in “Assessing process instances in
WebSphere Business Integration Server Foundation environment” on page 154.
This value defines the time that is needed to run both environments in parallel.
The time for these two systems and the extra effort required must be sized as
part of the migration planning to ensure that the hardware and human resources
are available during this phase of coexistence.

If the process instances of several templates must be migrated, for example,
having a loan approval process and a travel booking process that are running on
the same WebSphere Business Integration Server Foundation environment, you
must determine the order of migration. Migrate the process instances with the
longest duration first, and migrate the processes with the shortest duration last.
By doing this, you minimize the period of coexistence of WebSphere Business
Integration Server Foundation and WebSphere Process Server.
In case one business operation consists of several process instances, decide whether to migrate all of the related processes at one time to the WebSphere Process Server or whether to migrate them one after another. If you migrate all of the processes at one time, the period of coexistence is the time to complete a whole business operation. The second possibility, migration of these processes one after another, has the advantage because it can reduce the time that is required for coexistence, especially when business processes at the end of the call chain have a rather long duration.

Figure 7-29 on page 159 illustrates the scenario to migrate the longest processes last. This approach migrates the last process in the call chain to the WebSphere Process Server, followed by the preceding one, until the first process in the call chain is migrated to the WebSphere Process Server. The disadvantage of this approach is that it requires you to do modifications in the remaining WebSphere Business Integration Server Foundation processes in the old environment to call the processes that are already migrated to the WebSphere Process Server.
Adopting client applications

When the WebSphere Business Integration Server Foundation and WebSphere Process Server environments are running in parallel, ensure that new processes are kicked off only at the new system. Users must be able to work on staff activities or human tasks on both systems.

The client applications that interact with the WebSphere Business Integration Server Foundation must be enhanced to cope with both systems for the time of coexistence. The Java APIs for client interaction have changed. Common clients in WebSphere Business Integration Server Foundation use the business process engine (BPE) Generic API, the Process SessionBean Facade, or the BPE Java Message Service (JMS) API. Because these APIs changed, you must change the client to interact with the WebSphere Process Server system. You can find details about migrating the client in 9.1, “Migrating client applications” on page 194.

In addition to the client migration, you must enable the client applications to cope with the two environments during the period of coexistence. The client must determine where to direct the requests on a per request basis.

Important: Always direct a process start request for a migrated process instance to the WebSphere Process Server.

You might send requests to running process instances to the WebSphere Process Server first. If the API response indicates that the process instance was not found, the client needs to try WebSphere Business Integration Server Foundation. There might be cases where the client can decide upon the request message contents to which system the request must be directed. This decision can take place if the business content of the request allows such a determination, for example, an increasing order number as part of the request.
message. The client needs to know when the first order was placed on the WebSphere Process Server system. Based on the order number, the client can then determine which system to call. If the order number is higher than the first order in the WebSphere Process Server, then the client calls the WebSphere Process Server; otherwise, it calls WebSphere Business Integration Server Foundation.

**Directing human task queries:** Always direct queries for human tasks to both systems until all tasks in WebSphere Business Integration Server Foundation are completed and no new tasks are created while processing the remaining process instances. The results must be incorporated by the client to one list to be displayed to the user.

If the period of coexistence is short, users might be able to afford using two client applications instead of spending development efforts to have a common view of the WebSphere Process Server and WebSphere Business Integration Server Foundation environments, which saves development and test time. The WebSphere Business Integration Server Foundation client can be phased out together with WebSphere Business Integration Server Foundation at the end of the period of coexistence. A decision for this approach highly demands the client functionality and the kind of users to which it is exposed.

**Move-at-once approach**

Another option for handling long running process instances is to move them at a certain point-in-time. You stop the work of the instances in the WebSphere Business Integration Server Foundation environment and start the respective instances in the new WebSphere Process Server environment.

The advantage of this approach is that no additional hardware, administrative, and client development and maintenance effort are needed for a certain period of time like in the coexistence approach. The only additional development effort that is needed is to generate a tool that supports the transition of the process instances to the new environment. This tool must accomplish the following actions for each process instance to migrate:

1. Suspend the instance so that it stays in the same state during transition, which helps to ensure that process instances are not finished during the time of transition and, therefore, might be run twice: once in the old system and once in the new environment.
2. Get the input messages for the process instance and transform them into WebSphere Process Server-style input messages (Service Data Objects (SDO)), if applicable.
3. Initiate a new process instance on the WebSphere Process Server system.
4. Terminate the process instance on the WebSphere Business Integration Server Foundation environment.

This approach applies to few business processes because usually it is not possible to repeat steps. Too much work might be required for a redo of work done through human interaction. Back end systems that participate in the process must also be updated during this transition operation. These disadvantages might hinder the choice of this migration approach. This approach might apply to the following process instances:

- Processes require a short period of time until completion. Resubmitting this process instance in the new environment creates a backlog of work of at a maximum, this time on the new system and for the users.

- The number of running process instances is small, so that the processes can be handled manually after they are resubmitted on WebSphere Process Server. An administrator can manually complete the human tasks that were already completed in the WebSphere Business Integration Server Foundation process to get the new WebSphere Process Server process instance in the same state as the WebSphere Business Integration Server Foundation process instance before it was suspended.

- The processes are structured in a way that enables them so that resubmitting does not cause conflicts. An example of such a process is one that requires one or multiple approval steps before executing work. The work is all automatic, completing quickly without wait times. For such process instances, you can expect that they are, in general, waiting at one of the staff activities for human interaction. Therefore, when you suspend them and resubmit them on the new system, there might be, at most, some staff activities to repeat.

**Analyze:** Before you choose the move-at-once approach, always perform a careful case-by-case analysis to ensure that you choose an acceptable option.

**Advanced move-at-once approach**

Usually the process instances have already taken steps, which must not be repeated, in the new environment. In this case, the move-at-once approach is not acceptable. You want these process instances in the new environment to start at a state in which the corresponding instance in the old environment stopped.

Accomplishing a solution to hook up in the new process instance in the WebSphere Process Server, at the point at which the old processes stopped, is a difficult procedure. Additional logic is in the new required process to ensure that certain activities are skipped that were done already in the WebSphere Business Integration Server Foundation process instance. Considering the complexity of
the processes, it might not be reasonable to spend efforts in the development of such a move-over solution.
Workspace migration

In this chapter, we cover aspects and considerations for the migration of source artifacts within the workspace from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer. We first focus on the capabilities of the migration tools that are provided in the WebSphere Integration Developer. The migration tools include the migration wizard and command line scripts. Those tools automatically migrate WebSphere Business Integration Server Foundation service projects to WebSphere Integration Developer modules.

In this chapter, we include the following topics:

- “Introduction to the workspace migration” on page 164
- “General considerations” on page 165
- “Preparing source artifacts for workspace migration” on page 168
- “Migrating workspace by using the WebSphere Integration Developer migration wizard” on page 175
- “Migrating a workspace using a command-line interface and scripts” on page 182
- “Additional migration considerations” on page 185
8.1 Introduction to the workspace migration

In general, two migration paths are available for the application migration process:

- Runtime migration, also referred as runtime upgrade, moves the deployed application or binary code to the new environment without involving code changes.
  
  This is also called *backward binary compatibility support*.

- Source artifacts’ migration, which uses the tools to help a developer change the source code, to adopt the new programming model and get the existing applications running in the new environment.

Because of the runtime architecture difference and the programming model changes, the runtime migration path from WebSphere Business Integration Server Foundation to WebSphere Process Server is not available. The source artifacts migration is the only option.

There is detailed information about the migration of source artifacts in the information center:

“Migrating source artifacts to WebSphere Integration Developer from WebSphere Studio Application Developer Integration Edition” in the WebSphere Integration Developer Information Center


In WebSphere Integration Developer V6.2, you can only migrate a valid workspace which contains at least one service project. To fully migrate a WebSphere Studio Application Developer Integration Edition workspace, there are three fundamental tasks to complete:

1. Prepare source artifacts for migration.
   
   These actions might need to be performed in WebSphere Studio Application Developer Integration Edition.

2. Use either the Migration wizard or the WSADIEWorkspaceMigration command line script to migrate the workspace.

3. Where needed, use WebSphere Integration Developer to manually complete the migration, which involves fixing any Java code that was not automatically migrated and verifying the wiring of the migrated artifacts.
8.2 General considerations

WebSphere Process Server inherits the following new and enhanced functionality compared to WebSphere Business Integration Server Foundation:

- Enhanced support of the upcoming Web Services Business Process Execution Language (WS-BPEL) version 2.0 specification

  The business process engine that is within WebSphere Process Server is built on Business Process Execution Language for Web Services (BPEL4WS) Version 1.1 with major capabilities of the WS-BPEL Version 2.0 specification. These capabilities include, for example, support for compensation handlers and event handlers.

- Support for modeling and executing business state machines

- Enhanced support for business rules compared to the WebSphere Business Integration Server Foundation Business Rule Beans support

- Support for ease of mapping between interfaces and data types

During a migration project, consider whether the functionality that is available in WebSphere Process Server is valuable for the WebSphere Business Integration Server Foundation application that is to be migrated.

Consideration: As part of a migration project, always consider adapting a migrated WebSphere Business Integration Server Foundation application to use new WebSphere Process Server functionality.

Before you start the migration, review the general limitations of the migration process, which are outlined in the WebSphere Integration Developer Information Center at:


8.2.1 General migration tasks for source artifacts

The following list contains the major tasks sequence for migrating projects from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer:

1. Prepare the source artifacts for migration, which involves reviewing and updating your process and data model for the new programming model. These actions are usually performed in WebSphere Studio Application Developer Integration Edition.
2. Migrate the WebSphere Studio Application Developer Integration Edition workspace to the WebSphere Integration Developer workspace automatically by using the migration tools, the migration wizard, or command line scripts.

3. Using WebSphere Integration Developer, manually complete the migration, which involves fixing any Java code that cannot be automatically migrated and verifying the wiring of the migrated artifacts. The complexity of this is proportional with the following factors:
   - The complexity of the original business processes
   - The amount of Java code in the BPEL-Java snippets
   - The complexity and extent of the client code

4. Refactor the migrated projects to use the new WebSphere Process Server features, if applicable.

5. Manually migrate or change the WebSphere Business Integration Server Foundation artifacts where necessary.

Source migration notes:

- In WebSphere Integration Developer V6.2, you can only migrate a valid workspace that contains at least one service project.
- The migration wizard does not migrate application binaries. It only migrates source artifacts that are found in a WebSphere Studio Application Developer Integration Edition workspace.

8.2.2 Pre-migration considerations

There are numbers of considerations to examine before starting the workspace migration process:

Target workspace preparation
Always start the WebSphere Integration Developer in the empty new workspace if you are using the migration wizard, or specify a new workspace if you are using the command line migration scripts. The empty workspace means that not only are the projects removed from the workspace, but that all folders and files are also removed from the workspace folder on the file system.

Supported projects
The following WebSphere Studio Application Developer Integration Edition project types are supported by the workspace migration wizard:

- Service Projects
- Java Projects
- EJB Projects
There are other project types that might exist in WebSphere Studio Application Developer Integration Edition. Those projects are copied to the WebSphere Integration Developer workspace, but do not have any processing for migration.

**Service project design consideration**

The following practices show how to design WebSphere Studio Application Developer Integration Edition services to ensure that they migrate successfully to the new programming model:

- **Data transformation**
  
  Try to use the Assign activity wherever possible. The transformer service should only be needed when an advanced transformation is needed. You must use this practice because the intermediate component must be constructed in order for the SCA module to invoke a transformer service. Additionally, there is no special tools support in WebSphere Integration Developer for the transformer services created in 5.1; instead, you must use the WSDL or XML editor to modify the XSLT embedded in the WSDL file, if you must change the behavior of the transformer service.

- **Complex Data types**
  
  Ensure that all complex types are given a name and that each complex type can be uniquely identified by its target namespace and name.

- **WSDL and Service generation**

  Use the WSDL doc-literal style because it is the preferred style in 6.x. Specify one part per WSDL message if the WSDL is a document style, as per the Web Services Interoperability (WS-I) spec and the 6.x preferred style.

- **Business process design**

  Avoid using the Business Process Choreographer Generic Messaging API (Generic MDBs) because it is not provided in 6.x. Use the Business Process Choreographer Generic EJB API as opposed to invoking the generated session beans that are specific to a particular version of a process. These session beans are not generated in 6.x.
To learn more about service project design considerations, refer the WebSphere Integration Developer Information Center at the following address:


8.3 Preparing source artifacts for workspace migration

The first step in the source artifacts migration process consists of preparing the WebSphere Studio Application Developer Integration Edition workspaces for migration. In this section, we give an overview of the most important tasks during this step.

One important consideration in this step is the changes in the project organization between WebSphere Studio Application Developer Integration Edition service projects and WebSphere Integration Developer Module projects. These changes usually require a change in the WebSphere Studio Application Developer Integration Edition project structure before you migrate to WebSphere Integration Developer.

For details about this migration step, see “Preparing source artifacts for migration” in the WebSphere Integration Developer Information Center at the following address:


8.3.1 Project organization changes in WebSphere Integration Developer

Service-oriented architecture (SOA) is a loosely-defined industry standard. It presents all business processes, including Web services, Enterprise Information System (EIS) service assets, workflows, databases, and so on, in a service-oriented way.

Service Component Architecture (SCA) is an SOA standard that is defined by many leading software corporations including BEA Systems, IBM, Oracle®, Red Hat® Inc., SAP® AG, Siebel® Systems, Sun™ Microsystems, Sybase, TIBCO Software Inc., and others. One goal of the SCA is to separate business integration logic from implementation logic. By doing this separation, an integration developer can focus on assembling an integrated application rather than on the implementation details.
The SCA proposes a three-layer architecture:

- **Business integration logic**
  In this layer, the developer assembles the service components to build business solutions. The integration developer can graphically assemble the business process without the knowledge of low-level implementation details.

- **Service component**
  This layer offers a new level of abstraction of the service. Technically a service component contains the implementation, one or more interfaces, which define its inputs, outputs and faults, and zero or more references. A reference identifies the interface of another service or component that this component requires or consumes.

- **Implementation**
  This layer is the real implementation of the service and is never exposed to the business layer.

Figure 8-1 on page 170 illustrates the SCA layers.
WebSphere Integration Developer offers a new way to organize the SCA in separate layers. By using the assembly editor, the integration developer visually assembles components together. Integration developers are not exposed to the code within the components. The components are assembled independently of their implementation. Through the component assembly concept, developers can focus on solving business problems by using and reusing components rather than diverting attention to the technology that is implementing the services. These concepts and functionalists differ significantly from the environment that is proposed by WebSphere Studio Application Developer Integration Edition.
WebSphere Integration Developer organizes the project into two project types, Module and Library, when doing business integration development:

- The Module project is the business integration project type for developing an SCA-based application. It contains the artifacts that are shown in Figure 8-2 and explained in the list that follows.

- **Assembly diagram** refers to the module assembly of each module. The diagram is of the integrated business application and includes the components and wires that connect them. By using assembly diagrams, the integration developer can assemble the SCA components into an integration application.

- **Dependencies** shows any resources that this module needs from other Library projects.

- **Business logic** holds the processes, state machines, business rules and rule groups, human tasks, selectors, and Java code that relate to your business logic.

- **Data types** hold the data objects that are used in the module.

- **Interfaces** contain all of the operation interfaces that are used in the module.

- **Mapping** includes data maps, interface maps, relationships, and roles for SCA components interactions.

- **Events** contain the business events that are defined in the business process.
A Module project is packaged into an EAR file. It represents a basic unit of deployment and encapsulates the SCA module, J2EE projects, Java projects, and dependent libraries.

The Module project is equivalent to the service project in WebSphere Studio Application Developer Integration Edition with the enhancement of the SCA programming model.

- The Library project is the business integration project type for storing artifacts that are shared between multiple modules. Figure 8-3 shows the artifacts in the Library project.

![Figure 8-3 Library project](image)

The Library project contains Data Types (business objects), Interfaces, Mappings, and Events. It does not contain the business logic, and cannot be deployed to the server by itself.

With the concept of the Library project, the integration developer can reuse the basic SCA components between different modules.

The project organization in WebSphere Integration Developer implies considerations for migrating service projects from WebSphere Studio Application Developer Integration Edition:

- For WebSphere Integration Developer, all of the artifacts in Business Integration Module projects are local to the module and cannot be shared between the different modules. All artifacts that must be shared must be stored in the Business Integration Library projects.

- For WebSphere Studio Application Developer Integration Edition, shared artifacts, such as *Web Services Description Language* (WSDL) and XML Schema Definition (XSD) files, can be referenced by different service projects. You must ensure that you resolve these dependencies before you migrate to WebSphere Integration Developer by moving all the shared artifacts from the service project.

However, designing the right granularity of WebSphere Process Server modules might require work beyond the creation of Business Integration Libraries, for
example, it might be required to take advantage of the migration work to rearrange BPEL processes in separate modules to improve manageability and to allow for a better handling of versioning issues.

8.3.2 Preparing the WebSphere Studio Application Developer Integration Edition workspace

To prepare the environment before you migrate the source artifacts to WebSphere Integration Developer from WebSphere Studio Application Developer Integration Edition:

1. Back up the entire WebSphere Studio Application Developer Integration Edition V5.1 workspace.

2. Determine the best way to migrate the non-WBI-specific projects in your workspace by reviewing “Migrating from WebSphere Studio V5.1, 5.1.1, or 5.1.2” in the Rational Application Developer Information Center at the following address:

   http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.etools.rad.migration.doc/topics/tmigratefrom51x.html

3. Review “Developing Web services” in the Rational Application Developer Information Center at the following address for background information about the Web service functionality provided by Rational Application Developer:


4. Verify that WSDL and XSD files can be referenced, and reorganize projects if necessary. To fully migrate the BPEL files within a service project, ensure that all WSDL and XSD files that are referenced by the BPEL files can be resolved in a Business Integration project in the new workspace:

   - If WSDL and XSD files are in the same service project as the BPEL file, no further action is required.
   - If WSDL and XSD files are in a different service projects.

   Prior to migration, reorganize the artifacts in WebSphere Studio Application Developer Integration Edition V5.1 because Business Integration Module projects might not share artifacts. Here are two options for reorganizing the V5.1 artifacts:

   • Merge the artifacts into one module:

     i. In WebSphere Studio Application Developer Integration Edition, create a new Java project that holds all of the common artifacts.
     ii. Place all WSDL and XSD files that are shared by more than one service project into this new Java project.
iii. Add a dependency on this new Java project to all service projects that use these common artifacts.

- Keep a local copy of artifacts.

  Each service project has its own local copy of WSDL and XSD files, and there are no dependencies between service projects. This option is not recommended because it needs to maintain the same artifacts many times.

  - If the WSDL and XSD files are in any other type of project (usually other Java projects), the new migration tools in WebSphere Integration Developer V6.2 create a library and move the WSDL and XSD files over.

5. Review 5.2, “Artifacts preparation” on page 80, for the WebSphere Studio Application Developer Integration Edition service project design, and apply changes if necessary. For more details about service project considerations, see “Considerations for the source artifact migration process” in the WebSphere Integration Developer Information Center at the following address:


6. Clean and rebuild the application, and make sure that you are free of errors.

8.3.3 Preparing the WebSphere Integration Developer workspace

The tasks in this section are summarized from the detailed version that is in the WebSphere Integration Developer Information Center at:


To prepare the WebSphere Integration Developer workspace:

1. Use a new workspace.

   Tip: If you select the **Use this as the default and do not ask again** option in with an old workspace, change the default value after you start the product. To change the value, select **Window → Preferences**. In the Preferences window, select **General → Startup and Shutdown** and select the **Prompt for workspace on startup** option.

2. Select **Window → Preferences → General → Capabilities**, and review the environment that has all of the necessary capabilities enabled. See “Business integration capabilities” in the WebSphere Integration Developer Information
Center at the following address, which details exactly which capabilities are needed:


3. Review the “Developing Web services” section of the Rational Application Developer Information Center at the following address for additional background information about the Web service functionality that Rational Application Developer provides:

http://publib.boulder.ibm.com/infocenter/radhelp/v7r0m0/index.jsp?topic=/org.eclipse.jst.ws.doc.user/concepts/cws.html

8.4 Migrating workspace by using the WebSphere Integration Developer migration wizard

You can migrate WebSphere Studio Application Developer Integration Edition workspace with the help of a migration wizard in WebSphere Integration Developer. The migration wizard enables the migration of workspaces, including all projects.

8.4.1 Overview of the workspace migration wizard

The workspace migration wizard migrates all of the projects from the valid workspace into the new target workspace, WebSphere Integration Developer V6.2 workspace. A workspace is considered as a valid workspace if it has at least one WebSphere Studio Application Developer Integration Edition service project. For the service project, the migration wizard performs the following tasks:

1. Creates a new Business Integration module (The module name can be defined.)
2. Migrates the service project’s classpath entries to the new module
3. Copies all WebSphere Business Integration Server Foundation source artifacts from the selected source project to this module
4. Migrates the BPEL extensions in WSDL files
5. Migrates the business processes (BPEL files) from BPEL4WS version 1.1 to the new level that WebSphere Process Server supports, which is built on BPEL4WS version 1.1 with major capabilities of the upcoming WS-BPEL version 2.0 specification
6. Creates an SCA component for each BPEL process
7. Generates a monitoring (.mon) file for each BPEL process to preserve the default monitoring behavior from WebSphere Studio Application Developer Integration Edition (if necessary)

8. Creates imports and exports depending on the deployment options that you choose in WebSphere Studio Application Developer Integration Edition

9. Wires the BPEL component to its partner links (imports, exports, and Java components)

This wizard is meant to migrate source artifacts that are found in an WebSphere Studio Application Developer Integration Edition service project. It does not migrate any application binaries. In addition, the wizard only performs the migration process for the integration-service related projects. For other projects, it just copies the source artifacts into the target workspace but does not have any processing for migration. For what project is supported by the workspace migration wizard, refer to “Supported projects” on page 166 for details.

8.4.2 Using the migration wizard

Prepare the WebSphere Studio Application Developer Integration Edition source artifacts before you use the WebSphere Integration Developer migration wizard. For more information about this step, see 8.3, “Preparing source artifacts for workspace migration” on page 168.

To migrate WebSphere Studio Application Developer Integration Edition workspace, use the migration wizard, in WebSphere Integration Developer:

Note: Use a new WebSphere Integration Developer workspace as the migration target.

1. Invoke the wizard:
   a. Select File → Import.
   b. In the Import window, expand Business Integration, and select WebSphere Studio Application Developer Integration Edition Workspace. Click Next.

2. If the Migration wizard opens, enter the path for the workspace to migrate, or click Browse to find it. Click Next

3. From the migration options page, you can change the option to preserve the original BPEL Java snippets in the comments.

4. Click Finish to begin the migration process. A migration status bar is displayed at the bottom of the Migration wizard's window.

5. After the process completes, the message in Figure 8-4 is displayed.
6. Click **Next** to begin the workspace migration validation process.

7. In the workspace project selection window (Figure 8-5 on page 178), select the projects to migrate, and click **Next**.
8. The project resources that might be impacted by the migration process are listed in the message window, as shown in Figure 8-6 on page 179. Review and verify the list, and click **Next**.
9. In the Server Runtime Selection window, shown in Figure 8-7 on page 180, select the project you want to target in the new WebSphere Process Server V6.2 runtime, and click **Next**.
10. After you are ready, click **Finish** to begin migrating the projects that you selected.

11. After the migration validation process completes, the message in Figure 8-8 is displayed. Click **OK** to close the message window.

12. After the process completes, the Migration Results window opens, as shown in Figure 8-9 on page 181.
13. Optionally, you can generate the “ToDo” list or save all messages in a text file in the file system for future reference by clicking the appropriate buttons. Examine each message to see if any action needs to be taken to immediately fix an artifact that was not fully migrated.

After the Migration wizard finishes, the detailed migration results also show in the Migration Result view (Figure 8-10 on page 182), with color coded messages.
Fix any red error messages, build the workspace that was created, and try to resolve any build errors. Inspect all migrated BPEL files and ensure that they are fully migrated and can be opened in the WebSphere Integration Developer BPEL Editor. There are some BPEL Java snippets that cannot be automatically migrated. If you see any errors in the BPEL Java snippets, see 8.6.1, "Migrating the SCA programming model" on page 185 for steps to fix the errors.

8.5 Migrating a workspace using a command-line interface and scripts

WebSphere Integration Developer provides migration tools for both the graphic user interface (GUI) wizard and command-line interface (CLI) command. In this section, we discuss the CLI script support for migrating the WebSphere Studio Application Developer Integration Edition workspace.

8.5.1 Overview of the command-line script

The GUI wizard provides a user friendly interface that guides you step-by-step through the migration option settings and starts the migration process. However, this wizard has the following limitations:

- The wizard cannot be automated and scheduled.
- The wizard is not repeatable and not auditable.
- The wizard is prone to potential human error, which can cause the migration to fail.
The WSADIEServiceWorkspaceMigration command script enables the migration of the entire workspace to be invoked from the command-line scripts. Like the wizard tools, the migration command performs the following actions:

- Creates a new business integration module, which you define
- Migrates the classpath entries of the service project to the new module
- Copies all WebSphere Business Integration Server Foundation source artifacts from the selected source project to this module
- Migrates the BPEL extensions in the WSDL files
- Migrates the business processes (BPEL files) from BPEL4WS version 1.1 to the new level supported by WebSphere Process Server, which is built on BPEL4WS version 1.1 with major capabilities of the WS-BPEL version 2.0 specification
- Creates an SCA component for each BPEL process
- Generates a monitoring (.mon) file for each BPEL process to preserve the default monitoring behavior from WebSphere Studio Application Developer Integration Edition (if necessary)
- Creates imports and exports depending on the deployment options that are chosen in WebSphere Studio Application Developer Integration Edition
- Wires the BPEL component to its partner links (imports, exports, and Java components)

For large migration projects, you typically migrate multiple workspaces. Currently the migration command does not have the option for taking multiple projects. You can use the WSADIEWorkspaceMigration command script in the batch or shell scripts for all of the workspaces that you must migrate.

When working with multiple projects in the scripts, you must have the appropriate error handling in place to manage the errors if one or more project migrations fail.

### 8.5.2 Using the command-line script

Prepare the Integration Edition source artifacts before you use the migration command. For more information about this step, see 8.3, “Preparing source artifacts for workspace migration” on page 168.
To run the `WSADIEWorkspaceProjectMigration` command script:

1. Locate the script by opening the shared folder specified during the installation of WebSphere Integration Developer, for example, if during the installation, you set the shared folder as `D:\IBM\SDP70Shared`, then the `WSADIEWorkspaceProjectMigration` script is under `D:\IBM\SDP70Shared\plugins\com.ibm.wbit.migration.wsadie_6.2.0.v20081_126_1250`

2. Invoke the script as follows:

   `WSADIEWorkspaceMigration.bat -WIDstartup eclipse_dir -WIDworkspace WID_target_workspace -WSADIEworkspace source_WSADIE_Workspace_dir`

   Note the following parameter definitions:
   - `-WIDstartup` The location of your WID install root directory, or the location of `wid.exe`.
   - `-WIDworkspace` The full path to the new workspace where the new business integration module is created.
   - `-WSADIEworkspace` The full path to the WebSphere Studio Application Developer Integration Edition 5.1 workspace.

   Consider the following script as an example:

   `WSADIEWorkspaceMigration.bat -WIDstartup "C:\IBM\WID62" -WIDworkspace "C:\WID Workspaces\myWIDWorkspace" -WSADIEworkspace "C:\wsadie workspaces\myWSADIEWorkspace" > migration.log 2>&1`

3. After the command completes, examine the `migration.log` to make sure that it is error free.

4. Start WebSphere Integration Developer in the new created workspace.

5. Build the Business Integration module that was created and try to resolve any build errors. Inspect all migrated BPEL files. Ensure that they are fully migrated and can be opened in the WebSphere Integration Developer BPEL editor.

   Some BPEL Java snippets cannot be automatically migrated. If you see any error messages in the BPEL Java snippets, see 8.6.1, “Migrating the SCA programming model” on page 185, for steps to fix those errors.

6. Open the module dependency editor to ensure that the dependencies are set correctly. To do this, switch to the Business Integration perspective, and double-click the business integration module project. From here, you can add dependencies on business integration library projects, Java projects, and J2EE projects.
8.6 Additional migration considerations

When working on the WebSphere Studio Application Developer Integration Edition source artifacts migration process, some additional considerations must be checked carefully. The workspace migration wizard or scripts attempts to smooth the migration process by migrating most artifacts that are related to the WebSphere Studio Application Developer Integration Edition workspace automatically. Due to the programming model and architectural difference, some of the artifacts cannot be migrated using the tools. In those cases, manual intervention and checking is required to ensure that the migration was complete.

In this section, we discuss the programming model differences between the WebSphere Studio Application Developer Integration Edition and WebSphere Integration Developer, the limitations of the source artifact migration process, and some other additional considerations.

8.6.1 Migrating the SCA programming model

Because of the programming model change, the Java code that interacts with the integration service also must change. In this section, we show you how to migrate from the WebSphere Business Integration Server Foundation (WBISF) programming model to the new SCA programming model.

The programming model differences
The programming model in WebSphere Business Integration Server Foundation is the base on the Web Service Invocation Framwork (WSIF), where the integration service is built with the Web service interfaces, and the client interacts with the Web service through a set of simple Java APIs. The programming model in WebSphere Process Server is base on the SCA programming model, where the data flowing through the application is stored in Eclipse Service Data Objects (SDOs). The next section is a summary of the programming model changes.

The WBISF programming

- WSIF and WSDL based
  WSIF and WSDL define the interfaces for the client to look up and invoke the service
- Generated proxies for service
  The service proxy is built based on the service definition of the integration service, typically the WSDL definition or java service. Service proxy provides a client application with access to a service offered by an integration application.
Beans and format handlers for types

The Java helper classes, beans, and format handlers used marshal and unmarshal byte arrays and pass the data from client to the integration service.

8.6.2 Limitations for the source artifacts migration

There are limitations with the WebSphere Studio Application Developer Integration Edition source artifacts migration process.

General limitations
In this section, we discuss the general limitations with the WebSphere Studio Application Developer Integration Edition source artifacts migration process.

- Supported project types
The WebSphere Studio Application Developer Integration Edition project types that the Migration wizard supports are: Service Projects, Java Projects, EJB Projects, Connector Projects, Enterprise Application Projects, Application Client Projects, Dynamic Web Projects, and Static Web Projects. Any other project types that might exist in WebSphere Studio Application Developer Integration Edition are copied to the WebSphere Integration Developer workspace but do not have any processing for migration.

- Data type enforcing
WebSphere Studio Application Developer Integration Edition did not strictly enforce consistency between the WSDLs and other artifacts in projects. WebSphere Integration Developer is much stricter and reports inconsistencies that WebSphere Studio Application Developer Integration Edition did not (and also which ran on the WebSphere Business Integration Server Foundation runtime without any issue).

- Name and namespace collision
Although WebSphere Studio Application Developer Integration Edition allowed multiple identical Web Service Binding and Service definitions (name and namespace), WebSphere Integration Developer does not. You must resolve these duplicates manually before migration (in WebSphere Studio Application Developer Integration Edition) or after migration (in WebSphere Integration Developer). An example is that in WebSphere Studio Application Developer Integration Edition, all of the generated service definitions in the WSDL files with different names (ending in _EJB, _JMS, and so on) looked like:

<service name="OrderProcessIntfcService">
To fix the duplicate, simply append the binding type to the name attribute. For the *_.EJB.wsdl file, change it to:

```xml
<service name="OrderProcessIntfcServiceEJB">
```

For the *_.JMS.wsdl file, change it to:

```xml
<service name="OrderProcessIntfcServiceJMS">
```

**Note:** After the name is changed, the Export that is generated in WebSphere Integration Developer to use this service also must be changed to use the right name.

- **Source artifacts migration only**
  
  The Migration wizard does not migrate application binaries, it only migrates source artifacts found in a WebSphere Studio Application Developer Integration Edition workspace.

- **Access the JMS header properties**
  
  The standard provided JMS data binding does not provide access to custom JMS header properties. A custom data binding must be written for the SCA services to get access to any custom JMS header properties.

- **XML-SOAP types**
  
  WebSphere Integration Developer does not support XML-SOAP types, as defined in the [http://xml.apache.org/xml-soap](http://xml.apache.org/xml-soap) namespace. You must remove references to these types in WebSphere Studio Application Developer Integration Edition prior to migrating to avoid a migration process failure.

- **Module dependencies**
  
  When a workspace is migrated, some modules might have dependencies on other modules, which WebSphere Integration Developer does not check; however, there are errors similar to the following error after application deployment:

  ```
  ======= TravelOperationsApp ========
  ```

  The application cannot start: TravelOperationsApp
  com.ibm.ws.exception.RuntimeWarning:
  javax.resource.ResourceException: Failed to lookup
  ActivationSpec.sca/TravelOperations/ActivationSpec
  
  After migration, review the modules and manually fix project and build path dependencies between modules, which might require moving some WSDL and Java files into a common library project.
J2EE level

J2EE type projects are not migrated to the most current J2EE level. If you want to upgrade to the most current level, you can use the J2EE tools menu by right-clicking any J2EE type project (Web, EJB, EAR, Application Client) and selecting **Java EE → Specifications Migration Wizard**.

---

**EJB project migration limitation**

You might encounter a migration problem if the source WebSphere Studio Application Developer Integration Edition workspace has an EJB project without an EJB client project. If the EJB project is a dependency of one or more service projects, then the migrated workspace builds okay but does not deploy correctly. This happens because WebSphere Integration Developer tries to deploy the EJB project as an J2EE module and not a utility jar. To solve this problem, follow these steps:

1. Migrate the workspace.
2. In WebSphere Integration Developer, right-click EJB Project, and select **J2EE tools → Create Client Project**. An EJB client project is created.
3. Replace all references to the EJB project in Modules with the EJB client.

---

**SCA programming model limitation**

The limitations for the current SCA programming model are:

- The SDO version 1 specification does not provide access to the COBOL or C byte array, which impacts those working with IMS multi-segments.
- The SDO version 1 specification for serialization does not support COBOL redefines or C unions.
- The document/literal wrapped WSDL style (which is the default style for new artifacts created using the WebSphere Integration Developer tools) does not support method overloading. Use other WSDL style/encoding if you need the method overloading support.
- Native support for arrays is limited; therefore, use the "maxOccurs" attribute for the arrays elements.

---

**BPEL migration limitation**

The limitations for BPEL migration are:

- Multiple replies per BPEL operation

  In WebSphere Business Integration Server Foundation a business process can have one receive activity and multiple reply activities for the same operation. If you have a business process with multiple replies for the same operation, ensure that if any of them have client settings, that all replies for
that operation have the same client settings as in 6.x only one set of client settings is supported per operation reply.

- **Limitations of BPEL Java snippet migration**

  The programming model changed significantly from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer, and not all supported WebSphere Studio Application Developer Integration Edition APIs can be directly migrated to corresponding WebSphere Integration Developer APIs. The automatic migration tool might not be able to convert every BPEL Java snippet to the new programming model. Most of the standard snippet API calls are automatically migrated from the 5.1 Java snippet programming model to the 6.x Java snippet programming model. WSIF API calls are migrated to DataObject API calls where possible. Any custom Java classes that accept WSIFMessage objects need manual migration such that they accept and return commonj.sdo.DataObject objects instead. Refer 9.2, “Migrating API calls” on page 204 for more details.

- **BPEL variable message type**

  A WSDL message type must be specified for all BPEL variables used in Java snippets. Java snippets that access BPEL variables without the "messageType" attribute specified cannot be migrated.

**8.6.3 Unsupported artifacts**

When moving to a new architecture and programming model, lots of the features and artifacts become deprecated. Here we discuss some major artifacts in the WebSphere Business Integration Server Foundation which deprecated in the WebSphere Process Server V6.2 and the recommended migration options. For a full list of the deprecated features, refer to IBM inforcenter at:


**Business Rule Beans**

The Business Rule Beans component is deprecated in WebSphere Process Server 6.x and is replaced with business rules. Because there is no tools support for the old Business Rule Beans, you must manually remove all usage of the Business Rule Beans and move to the new business rule tools that are available in WebSphere Integration Developer to create their business rules according to the 6.x specification.
Extended messaging service

The extended messaging service feature and all of the extended messaging service (CMM) APIs and service provider interfaces (SPIs) are deprecated in WebSphere Process Server. Specifically, the following APIs are deprecated:

- com/ibm/websphere/ems/CMMCorrelator
- com/ibm/websphere/ems/CMMException
- com/ibm/websphere/ems/CMMReplyCorrelator
- com/ibm/websphere/ems/CMMRequest
- com/ibm/websphere/ems/CMMResponseCorrelator
- com/ibm/websphere/ems/ConfigurationException
- com/ibm/websphere/ems/FormatException
- com/ibm/websphere/ems/IllegalStateException
- com/ibm/websphere/ems/InputPort
- com/ibm/websphere/ems/OutputPort
- com/ibm/websphere/ems/transport/jms/JMSRequest
- com/ibm/websphere/ems/TransportException
- com/ibm/ws/spi/ems/CMMFacade
- com/ibm/ws/spi/ems/format/cmm/CMMFormatter
- com/ibm/ws/spi/ems/format/cmm/CMMParser
- com/ibm/ws/spi/ems/format/Formatter
- com/ibm/ws/spi/ems/format/Parser
- com/ibm/ws/spi/ems/transport/CMMReceiver
- com/ibm/ws/spi/ems/transport/CMMReplySender
- com/ibm/ws/spi/ems/transport/CMMSender
- com/ibm/ws/spi/ems/transport/MessageFactory

There is no WebSphere Integration Developer tooling support for the extended messaging applications of WebSphere Business Integration Server Foundation. Instead of using the extended messaging service and its associated tools, you must use standard JMS APIs or equivalent messaging technologies.

Container Manager Persistence over Anything (CMP/A) support

The Container Manager Persistence over Anything (CMP/A) support included with WebSphere Business Integration Server Foundation is deprecated in the WebSphere Process Server V6.2, which includes the runtime support for applications that were customized to use CMP/A, the cmpdeploy.bat/.sh command line tool, and the following public APIs:

- com.ibm.websphere.rsadapter.WSProceduralPushDownHelper
- com.ibm.websphere.rsadapter.WSPushDownHelper
There are three options for migrating the CMP/A application:

- Convert CMP entity beans to use a relational data source.
- Have the CMP entity bean replaced by a different supported data persistence model.
- Use WebSphere Adapters to replace your existing CMP/A applications.

The Adapter tools use a create, retrieve, update, and delete-type architecture for creating service interfaces that is similar to the architecture that CMP/A uses.
Client component migration

In this chapter, we highlight the necessary considerations and tasks that are involved in client component migration and discuss several client application migration methods. We also cover the migration of APIs, the CEI application, and Business Integration adapters.

The topics that we include in this chapter are:

- “Migrating client applications” on page 194
- “Migrating API calls” on page 204
- “Migrating CEI applications” on page 207
- “Migrating WebSphere Business Integration Adapters” on page 208
9.1 Migrating client applications

Migrating client applications might be required because of the differences in the programming models of WebSphere Business Integration Server Foundation and WebSphere Process Server. Any required code changes are manual. The migration tooling does not modify WebSphere Business Integration Server Foundation client applications to interact with WebSphere Process Server.

In WebSphere Business Integration Server Foundation, the most important APIs to consider for a migration are the Business Process Choreographer APIs, which are exposed as stateless session EJBs and as a generic JMS service.

WebSphere Process Server provides several APIs to interact with components and applications. The following migration considerations are the most important:

- The SCA API
- The Business Process Choreographer APIs:
  - Business Flow Manager API
  - Human Task Manager API

In WebSphere Process Server V6.2, there is no generic JMS API for Business Process Choreographer.

There are several new options for client development. Clients can use the SCA API to interact with SCA stand-alone references. They can also interact through several bindings, such as JMS or Web services, with SCA exports. Further, clients can use the Business Process Choreographer APIs to interact with processes and human tasks. There is also the option to use Web services calls to communicate with the Business Process Choreographer.

One decision to make during the migration of client code from WebSphere Business Integration Server Foundation to WebSphere Process Server is whether to adopt the client to one of the new options that WebSphere Process Server provides. In the following sections, we describe some of the migration options.

For a detailed discussion about the tasks that are involved with migrating client applications from WebSphere Business Integration Server Foundation, see the “Migrating WebSphere Business Integration Server Foundation client code” section in the WebSphere Integration Developer Information Center at the following address:

9.1.1 Migrating to client applications using the SCA programming model

In this section, we discuss the option to migrate the WebSphere Business Integration Server Foundation client code by adopting the client application to the new WebSphere Process Server SCA programming model. The code can be partially integrated in the programming model as an SCA component. In this case, define the interaction between this new component and the process by using the assembly editor in the natural environment of WebSphere Process Server.

Figure 9-1 illustrates the client migration options using the SCA programming model.

For elements outside of the SCA module, invoke the process component or any other SCA component using SCA exports.

Example 9-1 on page 196 is an example of calling a process.

```java
try {
    ServiceManager serviceMgr = new ServiceManager();
    Service service = (Service)
serviceMgr.locateService("BookingInterfacePartner");

//defining data
BOFactory bof =

(BOFactory)serviceMgr.locateService("com/ibm/websphere/bo/BOFactory");
DataObject input =
bof.createElement("http://data/BookingOrder","Input");
input.set("message", "data to start the process");
DataObject out = (DataObject) service.invoke("Input", input);
} catch (Exception ex){
...
}

Example 9-1 Calling a process

This case is appropriate for clients that must interact with the process by using only the input and output that is defined in the export of this process. In this case, the client does not require other data of the process or to know the life cycle of the process.

For further information, see "Migrating WebSphere Business Integration Server Foundation client code" in the WebSphere Integration Developer Information Center at the following address:


9.1.2 Migrating to clients using the Business Process Choreographer API

In WebSphere Business Integration Server Foundation, the Business Process Choreographer API offers a rich set of methods to interact with Business Process Choreographer processes and tasks. Figure 9-2 on page 197 illustrates clients using the Business Process Choreographer API.
In WebSphere Business Integration Server Foundation, the Business Process Choreographer provides this API functionality through a stateless session EJB or using JMS invocations. In WebSphere Process Server V6.2, the API is published as two EJBs or using invocations through Web services. Further, the functionality is enhanced and changed in some instances.
Deprecation methods in WebSphere Process Server: Many well-known method names in the Business Process Container API of WebSphere Business Integration Server Foundation V5.1 are maintained. However, some of the methods that are used in V5.1 are deprecated in WebSphere Process Server V6.0.2. Therefore, verify the migrated client code for the usage of these deprecated methods, and replace them accordingly.

Functionality enhancements
In both WebSphere Business Integration Server Foundation and WebSphere Process Server, with the Process Choreographer API functionality, you can perform the following tasks:

- Initiate a process
- Terminate a process
- Delete a process
- Claim, release, transfer, and complete human task activities
- Send messages
- Get and set fault messages
- Query data of processes, activities, and tasks

The initiate() method, for example, returns a process instance identifier (PIID) object. By using the PIID object in the getProcessInstance(PIID) method, an instance of the ProcessInstanceData class is returned. This object is the key for obtaining information, such as creation time, completion time, modification time, faults, starter, actual execution state, input message, and variables of this instance.

To obtain activity information, use getActivityInstance (PIID, nameOfActivity). It returns an ActivityInstanceData instance. This object encapsulates the activity data, for example, activation time, activity name, fault messages, execution state, type of activity (invoke, assign, reply, receive, and so on), owner, and the input and output message types.

WebSphere Process Server provides enhanced support for client interactions with human tasks. For interacting with the Human Task Manager, the following methods are the most important ones:

- callTask
- cancelClaim
- claim
- complete
- createFaultMessage
- createInputMessage
- createOutputMessage
As in WebSphere Business Integration Server Foundation, methods that can be called depend on the authorization of the person that uses the client code and the state of the task.

**Queries**

A powerful functionality of the Business Process Choreographer API is the capability to query the time stamps and states of the activities in the processes. You can ask the Business Process Choreographer for processes that use the query() method or for process templates that use the queryProcessTemplates() method. Both methods require the following items as arguments:

- Select clause, which describes the query result, that is, the names of the table views and columns of the query to be returned. Each entry must be of the form VIEW.COLUMN, where the views can be the following types:
  - ACTIVITY
  - ACTIVITY_ATTRIBUTE
  - EVENT
  - PROCESS_TEMPLATE
  - PROCESS_INSTANCE
  - PROCESS_ATTRIBUTE
  - WORK_ITEM
- Where clause, which is the search condition to be applied to the query.
<table>
<thead>
<tr>
<th>Order-by clause.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of query results.</td>
</tr>
<tr>
<td>Time zone for the constants in the where clause. When its value is null, UTC is assumed.</td>
</tr>
</tbody>
</table>

Example 9-2 shows a query for all of the activities.

```
BusinessProcess process = processHome.create();
QueryResultset rs = process.query(
    "ACTIVITY.STARTED, ACTIVITY.TEMPLATE_NAME, ACTIVITY.OWNER",
    null,
    null,
    null);
while (rs.next()) {
    System.out.println(rs.getString(2)+" @ "+rs.getString(1)+" for "+rs.getString(1));
}
```

**Example 9-2  Query example**

The queries in WebSphere Process Server are offered in the Business Flow Manager and in the Human Task Manager. More table views are provided, such as:

- ACTIVITY_SERVICE view
- APPLICATION_COMP view
- ESCALATION view
- ESCALATION_CPROP view
- ESCALATION_DESC view
- QUERY_PROPERTY view
- TASK view
- TASK_CPROP view
- TASK_DESC view
- TASK_TEMPL view
- TASK_TEMPL_CPROP view
- TASK_TEMPL_DESC view

**Interface changes**
WebSphere Business Integration Server Foundation offers two ways to interact with the container:

- A stateless session EJB BusinessProcess
- A JMS API to invoke a subset of the EJB API functionality
The JMS API is not present in WebSphere Process Server V6.2. There are several ways to migrate clients using this API. See IBM WebSphere Integration Developer information center “Migrating the JMS and the JMS process bindings” section to choose a way to expose the business process to consumers and rewrite the client according to the chosen binding.


The EJB BusinessProcess offers a local and remote interface. Example 9-3 shows the JNDI name of this interface in WebSphere Business Integration Server Foundation.

Example 9-3   Process Choreographer EJB interface

| JNDI Name: com/ibm/bpe/api/BusinessProcessHome |
| Interface: com.ibm.bpe.api.BusinessProcess |

To instantiate and send a message as input to a process, you use code similar to Example 9-4 in WebSphere Business Integration Server Foundation.

Example 9-4   Initiating process in WebSphere Business Integration Server Foundation

```java
try {
    Context ctx = new InitialContext();
    // Using the local reference
    LocalBusinessProcessHome processHome =
        (LocalBusinessProcessHome)
            ctx.lookup("java:comp/env/ejb/LocalBusinessProcessHome");
    LocalBusinessProcess process = processHome.create();
    // Create a message to be passed to the new process
    InputCriteriaMessageMessage input =
        new InputCriteriaMessageMessage();
    input.setContents("booking001");
    ClientObjectWrapper wrapper = new ClientObjectWrapper(input);
    process.initiate("BookingOrder", wrapper);
} catch (Exception exc) {
    exc.printStackTrace();
}
```

The BusinessProcess interface is no longer supported in WebSphere Process Server. WebSphere Process Server provides the BusinessFlowManager interface for all of the functions that are related to business processes and the HumanTaskManager interface for all functions that are related to human tasks, as illustrated in Figure 9-3 on page 202.
Example 9-5 shows the JNDI names of these interfaces.

Example 9-5  Process Choreographer EJB interfaces in WebSphere Process Server

BusinessFlowManager EJB
JNDI Name: com/ibm/bpe/api/BusinessFlowManagerHome

HumanTaskManager EJB
JNDI Name: com/ibm/task/api/TaskManagerHome
Interface: com.ibm.task.api.TaskManager

Conversion is required for any client application that uses the BusinessProcess interface in WebSphere Business Integration Server Foundation.

Web Services API in WebSphere Process Server
To invoke the functionality of the Business Flow Manager and the Human Task Manager, WebSphere Process Server offers two separate Web services interfaces (WSDL port types). By using the Web services API, the client can be a non-Java component, such as .NET.

Message changes
Many of the BusinessProcess interface methods, in WebSphere Business Integration Server Foundation, the BusinessFlowManager, and in the HumanTaskManager interface methods that are in WebSphere Process Server,
require a ClientObjectWrapper as an input parameter. The content of this ClientObjectWrapper object significantly changed from WebSphere Business Integration Server Foundation to WebSphere Process Server.

The data in the process is managed by WebSphere Business Integration Server Foundation by using the WSIF. WebSphere Business Integration Server Foundation uses the WSIFMessage interface to represent a WSDL message. Each message is a collection of parts, and each part has a name, a type, and a value.

WebSphere Studio Application Developer Integration Edition creates a Java class that implements the WSIFMessage interface for each variable of the process. You use this new object in Java snippets or in the client code. To use it in the Business Process Choreographer API, an additional ClientObjectWrapper class is provided. In WebSphere Business Integration Server Foundation, the content of such a ClientObjectWrapper object must be an instance of WSIFMessage. Example 9-6 shows the messages in WebSphere Business Integration Server Foundation.

Example 9-6   Messages in WebSphere Business Integration Server Foundation

// Create a message to be passed to the new process
InputCriteriaMessageMessage input =
    new InputCriteriaMessageMessage();
input.setContents("booking001");
ClientObjectWrapper wrapper = new ClientObjectWrapper(input);
// retrieving data
InputCriteriaMessageMessage input2 = (WSIFMessage) messageWrapper.getObject();

With the introduction of SDOs in WebSphere Process Server, the WSIF is no longer used. The messages in the process are now business objects that adopt the SDO API. The ClientObjectWrapper instance in WebSphere Process Server requires a DataObject as content instead of a WSIFMessage. Example 9-7 shows messages in WebSphere Process Server.

Example 9-7   Messages in WebSphere Process Server

//defining data
BOFactory bof =
    (BOFactory)serviceMgr.locateService("com/ibm/websphere/bo/BOFactory");
DataObject input =
    bof.createElement("http://data/BookingOrder","Input");
input.set("message", "data to start the process");
DataObject out = (DataObject) service.invoke("Input", input);
Client applications that use this API must change the way that they construct their ClientObjectWrapper instances when migrating to WebSphere Process Server.

9.1.3 Client migration approaches

The first step in the planning of the client migration is to define which role this code has in the entire migration project. For information about general migration approaches, see 7.3, “Migration approaches” on page 150. Each project has different constraints, which define the approach of the migration:

- Coexistence and different clients for WebSphere Business Integration Server Foundation and WebSphere Process Server

  In this case, each process uses different client code. The WebSphere Business Integration Server Foundation client code remains working without change, while new code is required for WebSphere Process Server.

- Coexistence and one client for both, WebSphere Business Integration Server Foundation and WebSphere Process Server

  In this case, the client code acts as a façade for the two systems. Therefore, you must develop new client code by using the old code as code fragments to rewrite the new version.

- Move-at-once approach

  In this case, write new client code. For some tasks, you must also consider that the client can use a non-Java programming language, such as .NET.

9.2 Migrating API calls

The data and invocation model in WebSphere Process Server differs from the programming model used in WebSphere Business Integration Server Foundation. Although the data model moved from WSIF and the WSDL, based in WebSphere Business Integration Server Foundation, to SDOs, based in WebSphere Process Server, the programming model moved from the Web service invocation model to the SCA.

WSIF is not supported in WebSphere Process Server V6.1, which implies the need for migration between these two programming models.

SDOs are used as a unified pattern for data representation, transport, and persistence across application layers. WebSphere Process Server includes business objects, which are enhanced SDOs. Business objects include some
extensions that are important for integration solutions and are used to further describe the data that is exchanged between SCA services.

A business object is a set of attributes that represent a business entity (such as an employee), an action on the data (such as create, read, update, and delete operations), and instructions for processing the data. Components of the integration application use business objects to exchange information and trigger actions.

SDO business objects are the primary data abstraction for the SCA, for example, Figure 9-4 illustrates how the SCA provides the framework to define service components and compose these services into an integrated application. It also shows that business objects represent the data that flows between each service.

In the paragraphs that follow, we show how to migrate from the WSIF programming model to the new SCA programming model where the data flowing through the application is stored in SDOs.

Java snippets within WebSphere Business Integration Server Foundation business processes must be migrated from using the old Java snippet API to the new Java snippet API where the data flowing through the application is stored in SDOs. Whenever possible, the snippets are migrated automatically by the migration wizard. However, the migration wizard cannot fully migrate some snippets and, therefore, manual steps are required to complete the migration.
Table 9-1 shows how to migrate from the original WebSphere Business Integration Server Foundation Version 5.1 programming model, where the data flowing through the application is represented as WSIFMessage objects with a generated interface that was strongly typed, to the new WebSphere Process Server V6.0 programming model, where the data is represented as SDOs and no strongly-typed interface is generated. This table shows the changes and solutions for migrating WSIFMessage API calls to SDO APIs.

<table>
<thead>
<tr>
<th>Change</th>
<th>Solution</th>
</tr>
</thead>
</table>
| WSIFMessage-based wrapper classes are no longer generated for WSDL message types. The JavaBean helper classes are no longer generated for complex schema types. | When writing code that interacts with SCA services, the generic SDO APIs must be used to manipulate the common sdo.DataObject messages that hold the data that flows through the application.  
WSDL message definitions that have a single simple-typed part are represented by a simple Java type that directly represents the part instead of having a wrapper around the actual data. If the single message part is a complex type, the data is represented as a DataObject that adheres to the complex type definition.  
WSDL message definitions that have multiple parts now correspond to a DataObject that has properties for all of the message parts, where complexTypes are represented as reference-type properties of the parent DataObject, accessible through the getDataObject and setDataObject methods. |
| Strongly-typed getter methods for WSIFMessage parts and generated JavaBeans must not be used. | Use a weakly-typed SDO API to get the DataObject properties.                                                                                                                                           |
| Strongly-typed setter methods for BPEL variables’ message parts are no longer available. | Use a weakly-typed SDO API to set the DataObject properties.                                                                                                                                               |
| Weakly-typed getter methods for WSIFMessage properties must no longer be used. | Use a weakly-typed SDO API to set the DataObject properties.                                                                                                                                               |
| Weakly-typed setter methods for WSIFMessage properties must no longer be used. | Use a weakly-typed SDO API to set the DataObject properties.                                                                                                                                               |
| All WSIFMessage API calls must be migrated to the SDO API where possible. | Migrate the call to an equivalent SDO API call where possible. Redesign logic if not possible.                                                                                                           |
9.3 Migrating CEI applications

The CEI was introduced as a technical preview in WebSphere Business Integration Server Foundation V5.1 and received a supported component in V5.1.1. This framework provides capabilities to publish Common Base Events from different event sources and provide them to different event consumer applications.

The CEI is also a part of WebSphere Process Server. The APIs of the CEI are the same, in general. Therefore, applications that use these APIs in WebSphere Business Integration Server Foundation do not need to change to WebSphere Process Server during a migration.

9.3.1 Business Process Choreographer events

In WebSphere Business Integration Server Foundation, predefined events can be emitted from business processes. Based on the business relevant value of activities, processes, links, and variables, events were emitted to the CEI.

In WebSphere Process Server, event emission is based on a *.mon file, which allows a more fine-grained choice of the events per activity, process, or variable that must be emitted. Further, the support for CEI was enhanced. It is possible to specify event emission for all WebSphere Process Server component types, and event emission can be enabled dynamically during run time.

The business process events that are emitted in WebSphere Business Integration Server Foundation are still available in WebSphere Process Server. However, they slightly changed and their contents were enriched. Therefore, custom applications that are used to rely on business process events might be adopted to the new business process and human task events that are introduced in WebSphere Process Server. The extended data element WPCEventCode, for example, is named BPCEventCode in WebSphere Process Server.

For more information about the changes in the events, see the following sections in the information center:

- “Monitoring business processes using common base events” in the WebSphere Business Integration Server Foundation Information Center
  
9.4 Migrating WebSphere Business Integration Adapters

WebSphere Business Integration Adapters consist of a collection of software APIs that provide native communication with the back end EIS and tools that enable configuring business objects and adapters. Adapters provide communication between the EIS and the integration broker, which, in this case, is the WebSphere Process Server. Although WebSphere Business Integration Adapters can communicate with other WebSphere broker products, in this section, we focus on WebSphere Process Server.

WebSphere Process Server continues to support the rich portfolio of existing WebSphere Business Integration Adapters, with which customers and business partners can take advantage of their investment and continued use. J2EE Connector Architecture (JCA) 1.5 WebSphere Adapters are staged over time.

If the client is a WebSphere Business Integration Adapter, it must use the Enterprise Service Discovery tooling in WebSphere Integration Developer to create the import with JMS binding by reading the back end. The client must also use this tooling to create an SDO that is serialized and communicated between WebSphere Process Server and the back end. This import uses a special data binding to serialize the SDO to the exact format that the WebSphere Business Integration Adapter expects.

Figure 9-5 on page 209 illustrates the WebSphere Business Integration Adapters support in WebSphere Process Server V6.1.
The integration of these adapters occurs using the same components and programming model as other SCA integration applications. The center of Figure 9-5 represents the WebSphere Process Server with a business integration application. The business integration application is made available for invocation to the other services outside of the SCA module through a JMS export. The business integration application can invoke other services outside of the SCA module through a JMS import. The adapters communicate with the back end systems using the application-specific data structure to the business object. They are configured using the connector configuration file (connector.cfg). When a business object is passed inbound to the WebSphere Process Server through the export, it is converted to a format that is understood by the WebSphere Process Server. When a business object is passed outbound to the adapter, it is converted to a format that is understood by the adapter.

This data synchronization pattern can also incorporate mapping of the business object from an application-specific format to a generic format.

WebSphere Business Integration Adapters communicate with WebSphere Process Server using the JMS protocol. The connector configuration file is configured to use WebSphere Application Server as the broker type. The default messaging support in WebSphere Process Server for service integration technologies messaging that uses MQClientLink is automatically configured the first time an enterprise application (EAR) that contains WebSphere Integration Adapter artifacts is installed. Destinations that are specific to each adapter are automatically configured.

Imports and exports are generated by WebSphere Integration Developer. They facilitate the communication between the adapters and WebSphere Process Server. Exports listen for incoming events on the JMS delivery destination queue or synchronous request destination queue. Through the export and data binding,
the incoming business object is converted from the WebSphere Business Integration format to the WebSphere Process Server business object or business graph format. The business object is then passed to the SCA component. Imports receive a business object or business graph from an SCA component and convert the object from the WebSphere Process Server format to the WebSphere Business Integration business object format. Then the import writes the message to the JMS request destination queue.

Export has the following functions:

- Listens for incoming events on the JMS delivery destination (InboundDelivery) or the synchronous request destination (inboundRequest)
- Converts the WebSphere Business Integration business object format to the WebSphere Process Server business object or business graph format
- Passes the business object or business graph to the SCA component
- Receives a response or exception from the SCA component
- Returns a response or exception to the synchronous response queue

Import has the following functions:

- Receives request business object or business graph from the SCA component
- Converts a business object or business graph to the WebSphere Business Integration format
- Writes a message to the JMS request destination
- Receives a response or exception message from the WebSphere Business Integration adapter on the response destination
- Checks the message for exceptions
- Converts the response message to the business object or business graph
- Returns the response to the calling SCA component or throws an exception

### 9.4.1 WebSphere Business Integration Adapter Artifact Importer

WebSphere Integration Developer is the tool for developing and migrating an application by using adapters. It supports a wide range of SCA adapters and WebSphere Business Integration Adapters. By using the WebSphere Business Integration Adapter Artifact Importer, the integration developer can discover existing enterprise services, business objects, and connector configurations. Then the developer can generate the SCA artifacts, such as import and export files, WSDL interfaces, and business objects, or the business graphs in a format that is compatible with WebSphere Process Server. After all of the artifacts are
imported and created, components are assembled into the integration solution, which results in an EAR file that can be exported and installed on WebSphere Process Server.

### 9.4.2 Migrating the external service by using Artifact Importer

If the WebSphere Business Integration Server Foundation service project connects to another enterprise service, after migrating the service project to the Business Integration module, you must use the WebSphere Business Integration Adapter Artifact Importer to use the WebSphere Business Integration Adapter to connect the same enterprise service.

To migrate:

1. In WebSphere Integration Developer, click **File → New → External Service.**
2. In the New External Service wizard, select **Messaging,** and click **Next.**
3. Select **WBI Adapter Artifact Importer,** and click **Next.**
4. In the Configure Settings for Discovery Agent window (Figure 9-6):
   a. For Connector configuration file, click **Browse,** and locate the file.
   b. For Business object schema directory, click **Browse,** and locate the directory.
   c. Click **Next.**

![Figure 9-6 Specifying initialization properties](image)
5. In the Find and Discover Enterprise Services window (Figure 9-7):
   a. Select the business objects to include in the application. Click **Edit Query** to see options to filter the resulting list of business objects that are discovered and the interaction mode in which they are used, such as outbound or inbound, request/response, or no way. The options determine the appropriate WSDL interfaces that are created.
   b. Select **Run Query** to run the query, after which the objects discovered by the query are displayed.
   c. Select the wanted objects, and click **Add** to add them to the Objects to be imported list.

![Enterprise Service Discovery]

*Figure 9-7  Find and Discover Enterprise Services window*
6. Specify the business object format:
   
a. Use the Generate Artifacts panel to specify the required format, such as business graph or SDO. The business graph provides the verb, change summary, and event summary information and wrappers of the business object. In many scenarios that involve WebSphere Business Integration Adapters, the business graph is the preferred format. However, this format depends on specific application requirements.
   
b. Select the target project and folder name.
   
c. Click **Finish**.
   
For more detailed information about adapter migration, including the planning, tools, steps, and sample scenarios, see *WebSphere InterChange Server migration to WebSphere Process Server*, SG24-7415.

### 9.4.3 Artifacts generated

Table 9-2 lists the artifacts that are generated from the adapter creation process. If the “Generate a business graph” option is selected, the *.xsd and *BG.xsd files are available in the business integration module.

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.xsd</td>
<td>Business object XSD file</td>
</tr>
<tr>
<td>*BG.xsd</td>
<td>Business graph of the supported business objects, which are the migrated versions of the original XSD files</td>
</tr>
<tr>
<td>ApplicationNameDeliveryHub.export</td>
<td>SCA export file that specifies the export for adapter delivery-to-hub communication</td>
</tr>
<tr>
<td>ApplicationNameRequestHub.export</td>
<td>SCA export file that specifies the export for adapter request-to-hub communication</td>
</tr>
<tr>
<td>ApplicationNameAgent.import</td>
<td>SCA import file that specifies the import for hub-to-adapter communication</td>
</tr>
<tr>
<td>ApplicationName.wsdl</td>
<td>The WSDL file that identifies the operations and messages that are used</td>
</tr>
<tr>
<td>ApplicationNameWsadmin.wbia</td>
<td>The file that is used as input to automatically configure MQ-ClientLink and JMS queues to allow communication between the server and the adapter framework</td>
</tr>
</tbody>
</table>
For information about how to integrate the generated artifact from the adapter migration process, see *WebSphere InterChange Server migration to WebSphere Process Server*, SG24-7415.
Post migration tasks

In Chapter 6, “Products runtime comparison” on page 91, Chapter 8, “Workspace migration” on page 163, and Chapter 9, “Client component migration” on page 193, we explain how to migrate applications from WebSphere Business Integration Server Foundation to WebSphere Process Server. After the applications are migrated, you must enable the migrated applications to function in WebSphere Process Server optimally. This is due to the differences between the architectures of WebSphere Process Server and WebSphere Business Integration Server Foundation.

In this chapter, we discuss the tasks and best practices that are necessary to execute post migration. This chapter includes the following topics:

- 10.1, “Refactoring processes” on page 216
- 10.2, “Further and new development of WebSphere Process Server artifacts” on page 217
- 10.3, “Performance tuning of a business process environment” on page 219
10.1 Refactoring processes

Consider the refactoring and tuning business process after migration because the original process is not always optimal for WebSphere Process Server.

In the following list, we summarize the best practices for the refactoring processes:

- Consider optimizing the generated Business Process Execution Language (BPEL) process.
  The migrated process is not always optimal and compact. Consider reworking the generated BPEL process into a more compact form.
- Adjust the migrated names to the conventions of the project. Prefer short and simple names to improve the readability of migrated process models.
- Wrap simple automatic activities, such as Service Component Architecture (SCA) components, and deploy them together with the process.
- Consider refactoring the process to use the WebSphere Process Server business rules engine.
  If the process uses if-then rules logic, then it can be externalized, and modifying processes in run time does not require a redeploy. In this case, the rule engine can simplify process complexity.
- Consider the refactoring process to use the WebSphere Process Server business state machine.
  Think of a business process as a series of sequential actions and the state machine as a series of loosely related stages that are acted upon. If the application is linear, use a business process. However, if the application is primarily event-driven and contains cyclical patterns, use a state machine. Both editors and the languages that they model are equally valid.
- Be as specific in staff assignments as possible.
  The more users that you assign to a task, the more you must update the work lists.
- Choose between inline and stand-alone tasks. There are two ways to implement a human task. If the task is implemented within a business process, it is called an inline task. Otherwise, it is referred to as a stand-alone human task:
  - An inline task is defined within an implementation of a business process. It can be implemented either directly in the process by using a human task activity or as a property of an invoke, pick, receive, event handler, or
message activity. Model tasks as inline tasks if any of the following conditions are present:

- Information from the process logic is necessary to execute human interaction.
- Administrative tasks are necessary.
- Authorization rights on specific activities must be specified.

- A stand-alone task exists independently of a business process and implements human interaction as a service that can be used in many of the different components of the WebSphere Integration Developer family of tools. Model tasks as stand-alone tasks if any of the following conditions are present:
  - Information from the business process is not necessary.
  - The task provides another service.

Choose between microflows and macroflows. Choose the microflow process by default because it is simpler and is executed much faster than a macroflow process. Choose a microflow process instead of a macroflow process if any of the following situations is valid:

- A process requires more than one transaction.
- A process must stop at any point and wait for external input, either in the form of an event or a human task.
- A process does not have IBM extensions enabled. A microflow is an IBM enhancement of the BPEL programming language.

Use compensation pairs and compensation handlers instead of exception handling hard-coded to activities. Remember that compensation handlers cannot be used in microflows, and if you use compensation pairs, switch to IBM extensions.

### 10.2 Further and new development of WebSphere Process Server artifacts

In the following list, we summarize the best practices for further development and customization for WebSphere Process Server artifacts:

- Use WebSphere Integration Developer to manually complete the migration.
  
  This practice involves fixing any Java code that cannot be automatically migrated and verifying the wiring of the migrated artifacts.
Use the development tooling to edit integration artifacts.

Use this practice to create, configure, and modify artifact definitions. Try to avoid manual manipulation of artifact metadata, for example, editing XML files directly, which may corrupt the artifact for migration.

Follow the Java 2 Platform, Enterprise Edition (J2EE) development rules:

- Adhere to J2EE development practices for portability. Make the code as safe as possible by using the appropriate exception handling. Also make the code compatible to run within a J2EE application server environment, even though it is currently running within a Java 2 Platform, Standard Edition (J2SE) environment.

- In Enterprise JavaBeans (EJB), as always, do not perform any functions that might be reserved for an EJB container, such as socket input/output, classloading, loading native libraries, and so on. If necessary, manually convert these snippets to use EJB container functions when migrated.

Enable all of the appropriate WebSphere Integration Developer features and capabilities.

Plan for easy deployment and reusability:

- If the update cycle of components is tied tightly to the process, deploy it with the process.

- If the component can be updated independently of the process, place it in a different module, which allows you to modify it independently of the process.

- If a process depends on an external service that it does not control, ensure the ability to change the details of the implementation.

- Consider using dynamic endpoint references.

- Gain a careful understanding of late and early binding.

- When you develop interfaces and business objects, consider what happens if requirements change in the future.

- When writing code inside the WebSphere Process Server or client code, consider using dynamic invocation mechanisms. If generated classes are used to invoke services, these specific implementations are tied to the artifacts.

- Do not rely on the order of an array in a business object. If you are using an array in a business object, do not rely on the order of the array when indexing into the array in Maps and Relationships.

Follow best practices for business objects design and development.

Be explicit in attribute lengths. Because business objects within WebSphere Process Server can be serialized at run time as they are passed between
components, it is important to be explicit with the required lengths for data attributes to minimize utilization of system resources.

- Follow best practices for map designs and development:
  - Always use WebSphere Integration Developer to create and modify maps, and avoid editing the metadata files directly.
  - Use a submap to reference child business objects in a map.
  - Use constants for SET values instead of Java code.

## 10.3 Performance tuning of a business process environment

In the following sections, we discuss an important but often overlooked step in the migration process, performance tuning. Due to time and deliverable constraints, performance tuning is often left as the last step in the migration process, and therefore is often knocked off the project timeline when time constraints come into play. In some cases, performance tuning is probably the most important step in the migration process.

### 10.3.1 Tuning your business process

Tune your process after migration by following these tips:

- Separate namespaces in business objects.
  - If you are working with more than one business object, it is best to put each one in a separate namespace. If you put them all in the same namespace, all of the business objects are loaded each time one of them is called, and the overall performance of the tool degrades.

- When possible, use microflows instead of long running processes.
  - If you are modeling a business process with a single transaction, consider making it a microflow. Microflows have great performance and run quickly in the runtime environment.

- Disable monitoring and tracing whenever possible.
  - Monitoring and tracing are a significant performance hit. The default configuration WebSphere Process Server has performance critical logs switched off by default. However, double-check that tracing, debugging, and performance monitoring is switched off.
Select a reliable and fast repository for the WebSphere Process Server database.

WebSphere Process Server is installed by default by using Derby. However, this platform is not designed for the production environment. It does not provide the best high availability, transactionality, and performance.

Tune statement caches for long running processes.

BPEL long running processes make extensive use of the database for persisting data that is relevant to the process. Persisting various data results in the usage of many different statements, far more than the default capacity of the data source's cache can result in an excessive number of cache misses, making the caching ineffective. This problem can be resolved by increasing the size of the cache.

Tune threads for messaging and work managers.

Message processing for an application uses properties that are defined in an activation specification of the Platform Messaging Component SPI Resource Adapter. Its custom property, maxConcurrency, under the J2EE Connector architecture (J2C) activation specification is used to specify the number of threads that are available to process messages by the application. If a work manager is used, set the Maximum number of threads to a value that is high enough to prevent the thread pool from running out of threads. One symptom of insufficient concurrency is CPU idleness. Vary the concurrency to achieve maximum CPU utilization and throughput.

Use an appropriate Java heap size and ratios for production environments.

See the technical documentation of the JVM that is used in the production environment to select the best parameters.

Configure WebSphere Process Server for clustering:

- Configure Activation Specification properties.
  
  Each SCA module defines a message-driven bean (MDB) and its corresponding activation specification. The default value for maxConcurrency of the SCA module MDB is 10, which means that only up to 10 asynchronous SCA requests in the module can be processed concurrently. If the server CPU is not maximized out, it sometimes is caused by this setting being too low. Therefore, you must increase it.

- Configure the Object Request Broker (ORB) thread pool.
  
  The ORB thread pool configuration parameter is relevant if the cluster is driven by a driver node through the SCA synchronous binding. Because of the interaction between synchronous SCA, workload manager, and the ORB, the ORB thread pool size on the cluster nodes must be configured to maximize the clustering throughput.
The rule of thumb is to use the same number of ORB threads on all application nodes. It includes having the total number of ORB threads across all application nodes be the same as the number of driver threads on the driver node, for example, if the driver uses 120 concurrent threads, the ORB thread pool size on each application node on a six-node cluster should be 20.

- Configure relationship and business process engine DataSource connection pools.
- The maximum connections property of the relationship DataSource should be large enough to allow concurrent access to the database from all threads.

For more information about performance tuning, see the following Redpaper publications:

- *IBM WebSphere Business Process Management V6.1 Performance Tuning*, REDP-4431
- *WebSphere Business Integration V6 Performance Tuning*, REDP-4195

In conclusion, never overlook performance tuning because it forms a necessary step in the migration of your business processes to the new environment.
Technical scenarios

This chapter includes the following topics:

- “Overview of scenarios” on page 224
- “Hardware and software environment setup” on page 226
- Working technical scenarios that demonstrate migration of various artifacts:
  - “Basic Travel Operations scenario” on page 227
  - “Extended Travel Operations scenario” on page 281
  - “Travel Booking scenario” on page 297
- “Scenario summary” on page 352
11.1 Overview of scenarios

In this chapter, we discuss the technical scenarios that we use in this book. The scenarios demonstrate the simple migration of WebSphere Business Integration Server Foundation solutions using the migration support that WebSphere Process Server provides. It also shows testing and verifying the migrated application and deploying it to the WebSphere Process Server.

In the first scenario, we show the basic steps to migrate a WebSphere Business Integration Server Foundation solution, which starts from artifacts in WebSphere Studio Application Developer Integration Edition. We then extend this scenario to add more functions to it.

In the second scenario, we focus on how to migrate the WebSphere Business Integration Modeler V5 model to WebSphere Business Modeler V6 model. We then export to WebSphere Integration Developer to add the implementation and to test the application.

All of the materials that we present in this chapter are available for download. See Appendix A, “Additional material” on page 373, for more information.

11.1.1 Travel Operations scenario

Using the Travel Operations scenario, we explain how to migrate a WebSphere Studio Application Developer Integration Edition application to WebSphere Integration Developer. A simple travel approval and booking scenario, which is developed in WebSphere Studio Application Developer Integration Edition, is migrated to a WebSphere Integration Developer project and then deployed and run in WebSphere Process Server.

The travel approval and booking sample consists of a Travel Operations process. In this scenario, this process is started when an employee requests approval for a business trip from a Web client. The approval request is reviewed within a staff activity. If the trip is approved, the Travel Operations process calls another process, the TravelAgency process, to book the trip. If the trip is not approved, then the process calls an Enterprise JavaBeans (EJB) service to send an e-mail to the employee to inform about the disapproval. The TravelAgency process invokes two Java services: one to reserve a hotel and one to reserve a flight, and then sends a booking confirmation e-mail to the employee.

After the scenario is fully migrated in WebSphere Integration Developer, a business rule is added to the scenario. It is invoked by the TravelAgency process to calculate the per-diem rate of the traveler.
Next the completed application is deployed to WebSphere Process Server to show the following information:

- How it works in the WebSphere Process Server environment
- How to modify the business rule by using Business Rule Manager
- How to monitor the business process in Business Process Choreographer Explorer

### 11.1.2 Travel Booking scenario

The model of the Travel Booking scenario is created from WebSphere Business Integration Modeler V5. To deploy the model to WebSphere Business Integration Server Foundation, you must import the model into WebSphere Studio Application Developer Integration Edition. You then must generate the service code and deploy code, add the implementation code, and deploy the application to the WebSphere Business Integration Server Foundation server. In this scenario, instead of migrating the service project as we do in the Travel Operations scenario, you migrate the model directly to WebSphere Business Modeler V6.2 and then migrate the V6.2 model to WebSphere Integration Developer.

The Travel Booking model shows a simple booking process. The focus of this scenario is to use the migration tool from version 5 to version 6 and to show the changes to perform in the artifacts to follow the new programming model in WebSphere Process Server. The focus is not on how to have a complete process defined in WebSphere Business Modeler.

The Travel Booking scenario has four human task activities and one service invocation. In this scenario, we concentrate on the process definition, the business items management, and the definition of the four human task activities. You see how to define the human task activities in WebSphere Business Modeler and associate them with the input/output forms for the activities. After the model is fully migrated, it is exported from WebSphere Business Modeler and brought into WebSphere Integration Developer. In there, the detailed implementation for the service invocation from the Booking Order process is completed. The completed application is deployed to WebSphere Process Server so that you can see how it works in the WebSphere Process Server environment.
11.2 Hardware and software environment setup

In this section, we describe the hardware and software for the test environment for the scenarios in this book. Table 11-1 represents the hardware system, which meets the minimum recommended requirements. The hardware and software environment setup is sufficient for the technical scenarios in this book and does not represent an actual production environment.

Table 11-1 shows the hardware configuration that we used.

Table 11-1  Hardware environment

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine type</td>
<td>IBM NetVista</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel® Pentium® 4 1.80 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>1.5 GB</td>
</tr>
<tr>
<td>Hard disk</td>
<td>40 GB</td>
</tr>
<tr>
<td>Operating system</td>
<td>Windows XP Professional</td>
</tr>
<tr>
<td></td>
<td>Service Pack 2</td>
</tr>
</tbody>
</table>

Table 11-2 shows the list of software products that are installed.

Table 11-2  Software versions used in the scenarios

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Process Server</td>
<td>6.2</td>
</tr>
<tr>
<td>WebSphere Business Integration Server Foundation</td>
<td>5.1.1</td>
</tr>
<tr>
<td>WebSphere Integration Developer</td>
<td>6.2</td>
</tr>
<tr>
<td>WebSphere Studio Application Developer Integration Edition</td>
<td>5.1.1</td>
</tr>
<tr>
<td>WebSphere Business Modeler</td>
<td>6.2</td>
</tr>
</tbody>
</table>
### 11.3 Basic Travel Operations scenario

For the Travel Operations scenario, we explain how to migrate a WebSphere Studio Application Developer Integration Edition application to WebSphere Integration Developer. A simple travel approval and booking scenario, which is developed in WebSphere Studio Application Developer Integration Edition, is migrated to a WebSphere Integration Developer project and then deployed and run on WebSphere Process Server.

Figure 11-1 shows the migration process.
The travel approval and booking sample consists of a Travel Operations process. In this scenario, this process is started when an employee requests approval for a business trip from the application client. The approval request is reviewed within a staff activity. If the trip is approved, the Travel Operations process calls another process, which is the TravelAgency process, to book the trip. If the trip is not approved, the Travel Operations process calls an EJB service to send an e-mail to the employee informing the person that the trip was not approved.

The TravelAgency process invokes two Java services, one to reserve a hotel and one to reserve a flight, and then sends a booking confirmation e-mail to the employee.

**Additional material:** The files that you can download for this scenario are available as explained in Appendix A, “Additional material” on page 373

### 11.3.1 Preparation

In this section, we discuss the initial steps that must be completed to migrate the Travel Operations scenario to WebSphere Process Server.

**Examining the WebSphere Studio Application Developer Integration Edition projects**

The WebSphere Studio Application Developer Integration Edition Project Interchange file, TravelOperationsScenario.zip, is in the attached compressed files that are available for download. See Appendix A, “Additional material” on page 373 for details. Import the artifacts into WebSphere Studio Application Developer Integration Edition.

The WebSphere Business Integration Server Foundation application to be migrated consists of two service projects, one Java project and one Web-client project, and an enterprise application:

- The TravelOperationsClient Client project inherits the following items:
  - One servlet that invokes the TravelOperations process
  - One JSP™ that submits to the servlet
- TravelClientEAR enterprise archive (EAR) project
  The TravelClientEAR project contains the TravelOperationsClient project.
- TravelOperations service project inherits the following items:
  - One long running process
  - One staff activity
  - EJB process binding to TravelAgency process
– EJB binding to EmailSender session bean

► TravelAgency service project inherits the following items:
  – One microflow
  – Two Java services
  – Java bindings to these Java services
  – EJB binding to EmailSender session bean

► EmailSender EJB project
The EmailSender EJB project includes a stateless session bean that sends an e-mail when invoked.

**E-mail:** In this example, a session bean is going to send e-mails. Therefore, a proper Simple Mail Transfer Protocol (SMTP) server must be available. In this scenario, the host relay.de.ibm.com is set in the TravelOperationsProcess and TravelAgencyProcess. To send an e-mail when you recreate the scenario, make sure to set a valid host name in the processes in your environment.

If you do not use a valid host, the scenario still runs properly. However, no e-mail is received.

► EmailSenderClient project
The EmailSenderClient project includes a Java project that functions as an EJB client for the EmailSender EJB project.

► EmailSenderEAR EAR project
EmailSenderEAR EAR project contains the EmailSender and EmailSenderClient projects.

► CommonArtifacts Java project
This Java project inherits the Web Services Description Language (WSDL) files for the EmailSender session bean. These files are used by the processes in the TravelAgency and TravelOperations projects to call the EmailSender bean.

Figure 11-2 on page 230 shows the module structure in WebSphere Studio Application Developer Integration Edition.
Examining Service Projects dependencies

In preparation for the migration, clarify all of the dependencies between the service projects to find the dependencies in the service projects:

1. In WebSphere Studio Application Developer Integration Edition, go to the Navigator view, and right-click the TravelAgency project.

2. Click Properties → Project References or Java Build Path. You see the dependencies for project TravelAgency, as shown in Figure 11-3 on page 231. The TravelAgency project has dependencies on CommonArtifacts and EmailSendClient projects.
3. Repeat steps 1 and 2 for the other service projects, and create a list of dependencies. For these service projects, it is sufficient to only look at the service project because the corresponding EJB and EAR projects are generated.

Populate the dependencies, as shown in Table 11-3 on page 232.

Figure 11-3  Project References of the TravelAgency project
Table 11-3  Service Project dependencies

<table>
<thead>
<tr>
<th>Service Project name</th>
<th>Required projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>TravelAgency</td>
<td>CommonArtifacts</td>
</tr>
<tr>
<td></td>
<td>EmailSenderClient</td>
</tr>
<tr>
<td>TravelOperations</td>
<td>CommonArtifacts</td>
</tr>
<tr>
<td></td>
<td>EmailSenderClient</td>
</tr>
<tr>
<td></td>
<td>TravelAgency</td>
</tr>
</tbody>
</table>

Table 11-3 shows that the service project TravelOperations has a dependency on another service project, the TravelAgency project. Dependencies must be resolved between service projects before migration to Integration Developer.

**Resolving dependencies between service projects**

To fully migrate the Business Process Execution Language (BPEL) files within a service project, ensure that all WSDL and XML Schema Definition (XSD) files that the BPEL files reference can be resolved in a business integration project in the new WebSphere Integration Developer workspace:

- If the WSDL or XSD files are in the same service project as the BPEL file, no further action is required.

  As an example for this case, see the WSDL files for the HotelReservationTool and FlightReservationTool Java services in the TravelAgency project. The TravelAgency process uses these files within the TravelAgency service project.

- If the WSDL or XSD files are in a different service project than the one that you are migrating, reorganize the V5.1 artifacts using WebSphere Studio Application Developer Integration Edition prior to migration because Business Integration module projects cannot share artifacts.

  In this scenario, the TravelOperationsProcess from the TravelOperations service project refers to the WSDL file of the TravelAgencyProcess in the TravelAgency service project, which must be reorganized.

  As an example, the option of reorganizing the V5.1 artifacts in WebSphere Studio Application Developer Integration Edition is selected. That is, you place all of the common XSD and WSDL artifacts in a Java project and add a dependency on this Java project to all service projects that use these common artifacts.

  The CommonArtifacts Java project stores the WSDL files for the EmailSender EJB service. This project resolves the dependency between the TravelOperations and the TravelAgency service projects. The TravelOperations project is dependent on the following two WSDL files within
the TravelAgency project, as shown in Figure 11-4. This is the interface of the TravelAgencyProcess, which the TravelOperationsProcess calls through EJB binding:

- TravelAgencyProcessInterface.wsdl
- TravelAgencyProcess_travelPort_EJB.wsdl

![Figure 11-4  WSDL files to move to the CommonArtifact Java project](image)

The two files must be moved to resolve the dependency between the service projects:

1. Open the **Project Navigator view**, and right-click **CommonArtifacts**. Choose **New → Package**, and enter `com.ibm.itso.travelagency`. Click **Finish**.

2. Go to the **TravelAgency** project.

3. Mark the files: TravelAgencyProcessInterface.wsdl and TravelAgencyProcess_travelPort_EJB.wsdl. Right-click the files, and choose **Move**.
4. In the Folder Selection window, Figure 11-5, within the CommonArtifacts project, browse to the travelagency folder. In the Select the destination field, CommonArtifacts/com.ibm.itso/travelagency is entered. Click OK.

![Folder Selection](image.png)

*Figure 11-5  Folder selection: move the WSDL files to CommonArtifact project*

5. From the menu, choose Project → Rebuild all to rebuild all of the projects.

6. Remove the TravelAgency dependency from the TravelOperations project. A new dependency does not need to be added to the CommonArtifacts project because this dependency already exists because both service projects are using the EmailSender EJB service. The WSDL files for this EJB service were already part of the CommonArtifacts project:
   a. Right-click the TravelOperations project, and select Properties.
   b. Go to Java Build Path, and click the Projects tab. Clear the TravelAgency project.
   c. Go to Project References, and clear the TravelAgency project.
   d. Click OK.
7. From the menu, choose **Project → Rebuild all** to rebuild all of the projects.

Now the dependencies between the two service projects are resolved.

An optional task is to verify the changes by running the projects in the WebSphere Studio Application Developer Integration Edition Unit Test Environment:

1. Generate deploy code for the TravelAgency project:
   a. Open the **Services** view.
   b. In the TravelAgency project, right-click the **TravelAgencyProcess.bpel** file, and select **Enterprise Services → Generate Deploy Code**.
   c. Ensure that the Referenced Partners are assigned, and click **OK**. The generation of deployment code is now started.

2. After the Generate Deploy Code wizard finishes, explore the TravelAgency project. Notice, that the **TravelAgencyProcess_travelPort_EJB.wsdl** file is regenerated, as shown in Figure 11-6.
3. Delete the TravelAgencyProcess_travelPort_EJB.wsdl file from the TravelAgency project:
   a. Right-click the TravelAgencyProcess_travelPort_EJB.wsdl file, and select Delete.
   b. In the Confirmation dialog, click Yes.
4. Rebuild all projects by choosing Project → Rebuild all from the menu.
5. Generate deploy code for the TravelOperations project:
   a. Open the Services view, and go to the TravelOperationsProcess.bpel file in TravelOperations project. Right-click this file, and select Enterprise Services → Generate Deploy Code.
   b. Ensure that the Referenced Partners are assigned.
   c. Ensure that travelPort now refers to the TravelAgencyProcess_travelPort_EJB.wsdl file in the CommonArtifacts project.
   d. Click OK to start the generation of the deployment code.
6. Add the projects EmailSenderEAR, TravelClientEAR, TravelAgencyEAR, and TravelOperationsEAR to the Unit Test Environment in WebSphere Studio Application Developer Integration Edition:
   a. In the Servers view, right-click the Unit Test Environment server, and select Add and remove projects.
   b. In the Add and Remove Projects window (Figure 11-7 on page 237), click Add All, and then click Finish.
c. In the Servers view, right-click the Unit Test Environment server, and select **Publish**.

7. In the Servers view, right-click the Unit Test Environment server, and select **Create tables and datasources**.

8. Start the Unit Test Environment server. Right-click the Unit Test Environment server, and select **Start**. A short function test is performed using the Travel Operation Web Client.

9. After the server starts, expand the **TravelOperationsClient** project, right-click **RequestSubmit.jsp**, select **Run On Server**, and select the Unit Test Environment server. Click **Finish**.
10. In the Web explorer that opens in WebSphere Studio Application Developer Integration Edition, complete the Process Input Message fields, as shown in Figure 11-8, and click **Submit**.

![Fill in the form:](image)

![Figure 11-8  Travel Web Client - Submitting a travel request](image)

11. Launch the Business Process Choreographer Web Client. Right-click the server, and select Launch Business Process Web Client. The Web client opens in WebSphere Studio Application Developer Integration Edition. A staff activity is created, as shown in Figure 11-9 on page 239.

12. In the Web client, complete the following steps:

   a. In the Process Web Clients navigation pane, click **My To Dos**. Select the one **ApproveTrip activity**, and click **Claim**.

   b. Click **ApproveTrip** to complete the activity.

   c. In the Activity page that opens, in the Activity Output Message field, type **true**, and click **Complete**.
d. After the TravelAgencyProcess is invoked, look at the SystemOut.log of the Unit Test Environment server, and check whether the processing of TravelAgencyProcess is successful.

**Note:** In case the SMTP host is not reachable, the last part of the output is slightly different. However, in the case presented in this section, the scenario finishes successfully for the reason of simplicity.
Example 11-1 shows the SystemOut log file of the TravelAgencyProcess when setting the Approve Trip to true.

Example 11-1  SystemOut of TravelAgencyProcess: ApproveTrip result is true

FRT: -----------------------
FRT: FlightReservationTool: Reserve a flight for:
FRT:   Destination: San Francisco
FRT:   Departure: New York
FRT:   From Mon Jan 01 10:00:00 CET 2007 till Tue Jan 02 10:00:00 CET 2007
FRT:   Credit Card Information:
FRT:     Card Number: 2314565
FRT:     Card Type: AMEX
FRT: Flight reservation successful.
FRT: -----------------------
HRT: -----------------------
HRT: HotelReservationTool: Reserve a hotel for:
HRT:   Destination: San Francisco
HRT:   From: Mon Jan 01 10:00:00 CET 2007 till: Tue Jan 02 10:00:00 CET 2007
HRT:   Credit Card Information:
HRT:     Card Number: 2314565
HRT:     Card Type: AMEX
HRT: Hotel reservation successful.
HRT: -----------------------
ESB: -----------------------
ESB: EmailSenderBean: Send email as follows:
ESB:   SMTP host: relay.de.ibm.com
ESB:   Sender: TravelAgency@itso.ibm.com
ESB:   Receiver: herrmans@de.ibm.com
ESB:   Subject: Travel Reservation Confirmation
ESB:   Body: Your trip to San Francisco has been booked successfully.
Your Travel Agency
ESB: Email sent successfully.
ESB: -----------------------

The SystemOut.log file of the Unit Test Environment server shows that the TravelAgencyProcess was invoked successfully.

If you redo step a-d on page 239, but type false as a result of the ApproveTrip activity in step c, the TravelAgencyProcess is not invoked. Instead, the TravelOperationsProcess sends a rejection e-mail, as shown in Example 11-11 on page 288.
Example 11-2  SystemOut TravelOperationsProcess: ApproveTrip result is false

ESB: -----------------------
ESB: EmailSenderBean: Send email as follows:
ESB:   SMTP host: relay.de.ibm.com
ESB:   Sender:    TravelOperations@itso.ibm.com
ESB:   Receiver:  herrmans@de.ibm.com
ESB:   Subject:   Your Travel Request Has Been Rejected
ESB:   Body:      Your travel approval request was rejected.
   Please refer to your manager for further details.
ESB: Email sent successfully.
ESB: -----------------------

You completed the verification of changes in the WebSphere Studio Application Developer Integration Edition projects to migrate to WebSphere Integration Developer.

Exporting the projects for migration
To export the projects for migration:
1. Choose File → Export.
2. Select File system, and click Next.
3. In the project browser:
   a. Select the following projects:
      • CommonArtifacts
      • EmailSender
      • EmailSenderClient
      • EmailSenderEAR
      • TravelOperationsClient
      • TravelClientEAR
      • TravelAgency
      • TravelOperations
   b. Choose a target directory, and click Finish.

The generated EJB and EAR projects for the service projects, such as TravelAgencyEAR and TravelAgencyEJB, are not migrated because the deployment code is regenerated for WebSphere Process Server.
Preparing WebSphere Integration Developer for the migration

To prepare WebSphere Integration Developer for the migration:

1. Start WebSphere Integration Developer, and use a new workspace.
   
   After launching WebSphere Integration Developer, the Welcome page, Figure 11-10, is displayed.

2. Disable the automatic build. From the menu, select Project, and clear the Build Automatically option.

3. Click Go to the Business Integration perspective, to open the Business Integration perspective.

---

**Note:** Sometimes we need the TravelAgencyEJB.

If both “JMS” and “IBM Web Service (SOAP/JMS)” bindings are chosen as deploy code options, the generated WSDL files have the same name. The “JMS” WSDL is put in the service project and the “IBM Web Service (SOAP/JMS)” WSDL is put in the generated EJB project. At this time if we do not include the generated EJB projects, we will get an error.
11.3.2 Implementation

In this section, we provide detailed steps to migrate the Travel Operations scenario using WebSphere Integration Developer.

Migrating the workspace

To migrate the workspace:

1. Import WebSphere Studio Application Developer Integration Edition workspace to WebSphere Integration Developer by selecting File → Import → Business Integration → WebSphere Studio Application Developer Integration Edition Workspace. Figure 11-11 on page 244 shows the Import panel.
2. In the **Migration** panel, select **Next**, click **Browse**, and select the WebSphere Studio Application Developer Integration Edition workspace on your machine,
as shown in Figure 11-12.

![Migration window](image)

Figure 11-12  Select WSADIE workspace

3. Select **Next**, and then click **Finish**. Figure 11-13 on page 246 is displayed.
4. The migration is on-going. When the **Workspace Migration** panel is displayed, click **Next**. Figure 11-14 on page 247 is displayed.
5. Make sure all of the projects that are listed on Figure 11-15 on page 248 are selected. Click **Next**.
6. Click **Next**. Figure 11-16 on page 249 is displayed.
7. Click **Next**. Figure 11-17 on page 250 is displayed.
8. Click **Next**. Figure 11-18 on page 251 is displayed.
9. Click **Finish**. The Migration Validation panel is displayed, Figure 11-19, saying that the migration is completed successfully. Click **OK**.

10. When the Migration Results panel, Figure 11-20 on page 252, is displayed, check the result panel, and click **OK**.
After the migration, go to the **Problems** view, there are two errors, as shown in Figure 11-21 on page 253.
12. To resolve the first error that is displayed in the Problems view, as shown in Figure 11-21, fix the classpath for the TravelOperationsClient project. Remove the classpath entry that refers to the V5.jar file:

a. Right-click the TravelOperationsClient project, and select Properties. Go to the Java Build Path entry, and click the Libraries tab, as shown in Figure 11-22 on page 254.

b. Select the jar file that is displayed in the Problems view, and click Remove.

c. Ensure that the JRE™ System Library and WPS Server Target libraries are present. Click OK.
d. Clean and build all projects in the workspace. From the menu, select **Project → Clean**. In the Clean dialog, select **Clean all projects**, and ensure that **Start a build immediately** is selected. Click **OK**.

13. After the build completes, there are four errors in the Problems view, as shown in Figure 11-23 on page 255.
The errors are present because the WSADIE client code uses the EJB Process Binding to invoke a business process. It should now use either the SCA API to invoke the service (the migrated service must have an Export with SCA Binding) or the IBM Web Service Client API to invoke the service (the migrated service must have an Export with Web Service Binding).

SCA API is used here to migrate the client code:

a. To create a new module, click File, select New, and select Module.

b. Use Client as the Module name, click Next, and click Finish.

c. Copy the TravelOperationsPort interface from the TravelOperations module to the Client module.

d. Open the Assembly Diagram of Client module. Drag-and-drop the TravelOperationsProcessExport from TravelOperations module into it, select Import With SCA Binding in the message. Click No in the Manage Project Dependencies message. When the new import appears, change the import name to Import1.

e. On the palette part of the Assembly Diagram, select the Standalone References item. Click the Assembly Editor canvas to create a new standalone reference.
f. Click the **Stand-alone Reference**, and link it to **Import1** import.

g. When the **Add Wire** window asks “A matching reference will be created on the source”, click **OK**.

h. Click **Save** to save the Assembly diagram. Figure 11-24 shows the Assembly Diagram of the Client Module.

![Assembly Diagram of Client Module](image)

Figure 11-24  Assembly Diagram of Client Module

i. Replace the error code of TravelRequestProcessServlet.java in the TravelOperationsClient project.

Example 11-3 shows the client code in the WSADIE workspace.

**Example 11-3  Client code using the WSIF EJB process binding in WSADIE**

```java
InitialContext context = new InitialContext();
Object result = context.lookup("com/ibm/bpe/api/BusinessProcessHome");
BusinessProcessHome processHome = (BusinessProcessHome)javax.rmi.PortableRemoteObject.narrow(result,BusinessProcessHome.class);
BusinessProcess bp = processHome.create();
WSIFMessage aMessage = new WSIFDefaultMessage();
aMessage.setObjectPart("requestorEmail", requestorEmail);
aMessage.setObjectPart("businessReason", businessReason);
aMessage.setObjectPart("from", from);
aMessage.setObjectPart("to", to);
aMessage.setObjectPart("arrivalDate", arrivalDate);
```


Example 11-4 shows the updated client code in the WID workspace.

**Example 11-4  Client code using the SCA API in WID**

```java
Service service = (Service)
  ServiceManager.INSTANCE.locateService("travelOperationsPortPartner");
BOFactory boFactory = (BOFactory)
  ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory")
  ;
DataObject wrapbo = null;
commonj.sdo.Type inputType =
  service.getReference().getOperationType("requestApprovalAndBook").getInputType();
if
  (service.getReference().getOperationType("requestApprovalAndBook").isWrapperType(inputType)) {
    wrapbo = boFactory.createByType(inputType);
    wrapbo.set("requestorEmail", requestorEmail);
    wrapbo.set("businessReason", businessReason);
    wrapbo.set("from", from);
    wrapbo.set("to", to);
    wrapbo.set("arrivalDate", arrivalDate);
    wrapbo.set("departureDate", departureDate);
  }
  service.invoke("requestApprovalAndBook", wrapbo);
```

j. Fix the classpath problem by replacing the import code of TravelRequestProcessServlet.java in the TravelOperationsClient project.

Example 11-5 shows the import code in the WSADIE workspace.

**Example 11-5  Import code in WSADIE**

```java
import org.apache.wsif.WSIFMessage;
import org.apache.wsif.base.WSIFDefaultMessage;
import com.ibm.bpe.api.ClientObjectWrapper;
```

k. Example 11-6 on page 258 shows the updated import code in the WID workspace.
Example 11-6  Updated import code in the WID workspace

```java
//import org.apache.wsif.WSIFMessage;
//import org.apache.wsif.base.WSIFDefaultMessage;
import com.ibm.bpe.api.ClientObjectWrapper;
import com.ibm.websphere.bo.BOFactory;
import com.ibm.websphere.sca.Service;
import com.ibm.websphere.sca.ServiceManager;
import commonj.sdo.DataObject;
```

I. The TravelClientEAR project belongs to a Version 1.3 J2EE application, and because of this it must be migrated to J2EE 1.4 using the Rational Application Developer Migration wizard:
   i. Right-click the TravelClientEAR project, and select Java EE → Specifications Migration Wizard. Click Yes on the pop-up panel, as shown in Figure 11-25.

![Open Editors](image)

Figure 11-25  Open editors

   ii. Review the warning statements on the page, shown in Figure 11-26 on page 259. Click Next.

**Warning messages:** Because this is a sample scenario, we discarded the recommendations in the warning dialog. However, we do recommend that you follow the recommendations in this warning dialog when you with real projects.
iii. Ensure that the TravelClientEAR project is highlighted as the project to be migrated. Choose J2EE version 1.4, and click Next.

i. In the Projects list, ensure that the TravelClientEAR project is selected. Click Finish.

ii. If this step completes successfully, a successful migration message is displayed. Click Details to review what was migrated. Click OK to close the dialog.
m. Go to the Java EE perspective by selecting **Window → Open perspective → Others → Java EE → OK**.

n. Open the Deployment Descriptor of TravelClientEAR, and click the **Module** section.

o. Add the **ClientEJB** into the **Modules** part. Add Client and CommonArtifacts into the Project Utility JARs part. Save the deployment descriptor.

14. Clean and build all projects in the workspace. From the menu, select **Project → Clean**. In the Clean dialog, select **Clean all projects**, and ensure that **Start a build immediately** is selected. Click **OK**.

15. Make sure there are no errors in the Problems view.

16. Explore the newly generated TravelAgency module in the Business Integration view, as shown in Figure 11-27 on page 261.
   - Under Processes, the migrated TravelAgencyProcess BPEL process is found.
   - Under Java, there are three generated Java components that translate the BPEL calls to the Java implementations for HotelReservationTool and FlightReservationTool and to the EmailSender Session Bean.
   - In Data Types, the .xsd files CreditCardDetails and FlightReservationDetails can be found that are used for the invocation of HotelReservationTool and FlightReservationTool.
   - Under Interfaces, the .wsdl Interfaces for HotelReservationTool and FlightReservationTool reside. There is a new Java Interface that was created to invoke the EmailSender Session Bean from the TravelAgencyProcess.
Figure 11-27  TravelAgency module
17. Inspect the migrated TravelAgency project by opening the Assembly Diagram for the TravelAgency module to see the components that the Migration wizard created:

   a. Click the Business Integration view, select the TravelAgency module, and double-click **Assembly Diagram**:

      - In the Assembly Diagram, SCA components can be wired together to obtain similar functionality to the Version 5.1 wiring functionality during the generation of deployment code.
      - The implementation type of the SCA components is shown by an icon in front of the component name.
      - In the TravelAgency Assembly Diagram, the components, exports, and imports can be observed, which Table 11-4 displays. See also Figure 11-28 on page 264.

*Table 11-4  Components, exports, and imports: Migration wizard generated*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TravelAgencyProcess</td>
<td>Process component</td>
<td>This component represents the migrated TravelAgencyProcess.bpel.</td>
</tr>
<tr>
<td>FlightReservationTool</td>
<td>FlightReservationTool Java component</td>
<td>This component represents the migrated FlightReservationTool Java service.</td>
</tr>
<tr>
<td>HotelReservationTool</td>
<td>Java component</td>
<td>This component represents the migrated HotelReservationTool Java service.</td>
</tr>
</tbody>
</table>
Observe that all these components are readily wired. Open their implementations by double-clicking the components.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmailSenderBeanJava Med</td>
<td>Java component</td>
<td>In the Process Server, there is no EJB binding like in Server Foundation. To call a Session Bean from a BPEL process in Process Server, use an Import with Stateless Session Bean Binding. To translate between the SDO data model that the process uses and the JavaBeans that the EJB uses, the Migration wizard introduced this Java component.</td>
</tr>
<tr>
<td>EmailSenderBean</td>
<td>Import with Stateless Session Bean Binding</td>
<td>This import was generated to call the EmailSenderBean EJB from the TravelAgencyProcess. Together with the above Java component, it replaces the EJB binding used in Server Foundation.</td>
</tr>
<tr>
<td>TravelAgencyProcessExport</td>
<td>Export with SCA Binding</td>
<td>This export enables the TravelAgency module to be called from another SCA module. It is the replacement for the EJB binding that was used to call the TravelAgencyProcess in V5.1.</td>
</tr>
</tbody>
</table>
b. Double-click the TravelAgencyProcess component. When the Process Editor opens, inspect the migrated TravelAgencyProcess, as shown in Figure 11-29.

The Partner, Variable, and Correlation Set definitions are located at the right corner of the editor. Notice that Partners are divided into Interfaces Partners, which represent the interfaces of the processes that are callable by clients and Reference Partners, which are called by the process. The Variable names are the same as they are in the V5.1 project.
c. Review the Assign activity and Snippet activity implementations to ensure that the migration worked correctly.

d. Review the Prepare Credit Card and Flight Details snippet to see how the Migration wizard changed the V5.1 programming model to the V6 snippet programming model.

See the comments in the Java code used in V5.1. Example 11-7 shows the V5.1 code to set the CreditCardDetails in the FlightRequest variable. Compare Example 11-7 to Example 11-8, which shows the Process Server Java programming model to fulfill this operation.

**Example 11-7  Fill the CreditCardDetails in the FlightRequest variable in V5.1**

```java
// Fill credit card details
CreditCardDetails creditCardDetails = new CreditCardDetails();
creditCardDetails.setCardNumber(getTravelRequest().getCardNumber());
creditCardDetails.setCardType(getTravelRequest().getCardType());
creditCardDetails.setValidUntil(getTravelRequest().getCardValidUntil().toString());
getFlightRequest(true).setCreditCardDetails(creditCardDetails);
```

While strongly typed, you can use getter and setter methods to access the CreditCardDetails in V5.1. In V6, the BO and SDO API are used to create and fill the elements of the CreditCardDetails object.

To use the Process Server BO services that are used for copying and creating service data objects, you must first initialize them.

Example 11-8 shows the CreditCardDetails in the FlightRequest variable in V6.

**Example 11-8  Fill in the CreditCardDetails in the FlightRequest variable in V6.1**

```java
com.ibm.websphere.bo.BOCopy boCopy = (com.ibm.websphere.bo.BOCopy)
    com.ibm.websphere.sca.ServiceManager.INSTANCE.locateService("com/ibm
/websphere/bo/BOCopy");
com.ibm.websphere.bo.BOFactory boFactory =
    (com.ibm.websphere.bo.BOFactory)
    com.ibm.websphere.sca.ServiceManager.INSTANCE.locateService("com/ibm
/websphere/bo/BOFactory");
// Fill credit card details
commonj.sdo.DataObject creditCardDetails =
    boFactory.create("http://itso.ibm.com/", "CreditCardDetails");
if (TravelRequest == null) {TravelRequest =
    boFactory.createByType(getVariableType("TravelRequest");}
creditCardDetails.setInt("cardNumber",
    TravelRequest.getInt("cardNumber");
```
creditCardDetails.set("cardType",
TravelRequest.getString("cardType"));
creditCardDetails.set("validUntil", ((java.util.Date)
TravelRequest.get("cardValidUntil")).toString());
if (FlightRequest == null) {FlightRequest =
boFactory.createByType(getVariableType("FlightRequest"));
}
FlightRequest.set("creditCardDetails",
boCopy.copy(creditCardDetails));
// Fill flight details
commonj.sdo.DataObject flightDetails =
boFactory.create("http://flight.itso.ibm.com/",
"FlightReservationDetails");
flightDetails.set("arrivalAirport", TravelRequest.getString("to");
flightDetails.set("arrivalDate", ((java.util.Date) TravelRequest
.get("arrivalDate"));
flightDetails.set("departureAirport",
TravelRequest.getString("from");
flightDetails.set("departureDate", ((java.util.Date) TravelRequest
.get("departureDate"));
FlightRequest.set("flightDetails", boCopy.copy(flightDetails));

e. Close the TravelAgencyProcess by clicking the X on the top of the
process editor.

In the Assembly Diagram, double-click the **HotelReservationTool**
component. When the Java editor with the HotelReservationToolImpl.java
class comes to front, inspect the methods in this file. The Migration wizard
generated this Java class to transform the data model used between calls
from the TravelAgencyProcess to the HotelReservationTool Java
implementation.

f. Complex types, such as the CreditCardDetails, are represented as data
objects in Process Server and accessed using the SDO API. In Server
Foundation, they can be accessed as JavaBeans. The
HotelReservationToolImpl translates the CreditCardDetails SDO that
comes from the TravelAgencyProcess into an CreditCardDetails
JavaBean to be used by the V5.1 HotelReservationTool.java
implementation.

Example 11-9 on page 267 shows the converters for CreditCardDetails
between DataObject and JavaBean.
Example 11-9  Converters for CreditCardDetails between DataObject and JavaBean generated by the Migration wizard

```java
/**
 * This conversion method was generated during migration
 */
private CreditCardDetails _convertTo_CreditCardDetails(DataObject aDataObject) {
    if (aDataObject != null) {
        CreditCardDetails pojo = new CreditCardDetails();
        pojo.setCardType(aDataObject.getString("cardType"));
        pojo.setValidUntil(aDataObject.getString("validUntil"));
        pojo.setCardNumber(aDataObject.getInt("cardNumber"));
        return pojo;
    }
    return null;
}

/**
 * This conversion method was generated during migration
 */
private DataObject _convertFrom_CreditCardDetails(CreditCardDetails aCreditCardDetails, DataObject aDataObject) {
    if (aCreditCardDetails != null) {
        aDataObject.setString("cardType", aCreditCardDetails.getCardType());
        aDataObject.setString("validUntil", aCreditCardDetails.getValidUntil());
        aDataObject.setInt("cardNumber", aCreditCardDetails.getCardNumber());
        return aDataObject;
    }
    return null;
}
```

g. Close the HotelReservationToolImpl.java file by clicking the X on the top of the Java editor.

h. Inspect the FlightReservationTool component.

i. In the Assembly Diagram, select the EmailSenderBean import to display its properties in the Properties pane, as shown in Figure 11-30 on page 268. Inspect its properties:
   - Notice, that the JNDI name for the EmailSenderBean in the bindings section was migrated from the EmailSenderBeans.wsdl file in V5.1.
   - Further notice, that the Interface of the EmailSenderBean import is a Java style interface, which the Migration wizard created.
j. In the Assembly Diagram, double-click the **EmailSenderBeanJavaMed** component.

k. When the Java editor with the `EmailSenderBeanJavaMedImpl.java` opens, inspect the file. The Migration wizard generated this class to mediate between the WSDL style interface that the TravelAgencyProcess is calling and the Java style interface that the EmailSenderBean import requires.

Using SCA operations, this class locates the EmailSenderBean import, as shown in Example 11-10.

**Example 11-10  Calling the sendEmail method on the EmailSenderBean import**

```java
public boolean sendEmail(String host, String from, String to, String subject, String email) {
    try {
        return ref.sendEmail(host, from, to, subject, email);
    } catch (java.rmi.RemoteException e) {
        throw new com.ibm.websphere.sca.ServiceRuntimeException(e);
    }
}
```

l. Close the `EmailSenderBeanJavaMedImpl.java` file by clicking the X on the top of the Java editor.

m. In the Assembly Diagram, select the **TravelAgencyProcessExport** export to display its properties in the Properties pane. Inspect its properties.

This export can be used to call the TravelAgencyProcess from a different SCA module. It replaces the EJB process binding, which was exposed by the TravelAgencyProcess in V5.1.

This export is left in the Assembly Diagram because it can be used later by the TravelOperationsProcess to call the TravelAgencyProcess.
n. Close the Assembly Diagram by clicking \textbf{X} on the top of the Assembly Diagram editor.

o. Open the module dependency editor to ensure that the dependencies are set correctly:
   
i. Double-click \textbf{Dependencies} within the TravelAgency module in the \textbf{Business Integration} view.

   ii. The Dependencies editor opens, as shown in Figure 11-31 on page 270. Observe that the TravelAgency module has one dependency to the CommonArtifacts Business Integration Library and a second dependency to the EmailSender Session Bean Client.

   Both of these dependencies also existed in the former WebSphere Studio Application Developer Integration Edition project.

   iii. Close the Dependencies editor by clicking the \textbf{X} on the top of the Dependency Editor.
This completes your inspection of the TravelAgency module.

Now we inspect the TravelOperations module:

18. Explore the newly generated TravelOperations module in the **Business Integration** view, as shown in Figure 11-32 on page 271.
19. Inspect the migrated TravelOperations project:
   
a. Open the Assembly Diagram for the TravelOperations module to see the components that were created by the Migration wizard by double-clicking the *Assembly Diagram* within the TravelOperations module in the *Business Integration* view.
In the TravelOperations Assembly Diagram, the following components, exports, and imports can be observed, as shown in Table 11-5. See also Figure 11-33 on page 273.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TravelOperationsProcess</td>
<td>Process component</td>
<td>This component represents the migrated TravelOperationsProcess. bpel.</td>
</tr>
<tr>
<td>travelPort</td>
<td>Import with SCA binding</td>
<td>This component is the replacement for the EJB binding that was used to call the TravelAgencyProcess in V5.1. It calls the TravelAgencyProcessExp export in the TravelAgency module.</td>
</tr>
<tr>
<td>EmailSenderBeanJavaMed</td>
<td>Java component</td>
<td>In Process Server, there is no EJB binding like in Server Foundation. To call a Session Bean from a BPEL process in Process Server, use an Import with Stateless Session Bean Binding. To translate between the SDO data model that the process uses and the JavaBeans that the EJB uses, the Migration Wizard introduced this Java component.</td>
</tr>
<tr>
<td>EmailSenderBean</td>
<td>Import with Stateless Session Bean Binding</td>
<td>This import was generated to call the EmailSenderBean EJB from the TravelOperationsProcess. Together with the above Java component, it replaces the EJB binding that is used in Server Foundation.</td>
</tr>
</tbody>
</table>
Observe, that all these components are readily wired. Their implementation opens by double-clicking the components.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TravelOperationsProcessExport</td>
<td>Export with SCA binding</td>
<td>This export enables the TravelOperations module to be called from another SCA module.</td>
</tr>
<tr>
<td>TravelOperationsProcess_JMSExport</td>
<td>Export with JMS binding</td>
<td>This export can be used to call the TravelOperationsProcess through JMS. It is one of the possible replacements for the process JMS API in V5.1. Custom coding and configuration is needed in order to use this export.</td>
</tr>
</tbody>
</table>

b. Double-click the TravelOperationsProcess component. When the Process Editor opens, inspect the migrated TravelOperationsProcess, as shown in Figure 11-34 on page 274.
Notice, that the Migration Wizard created a CatchAll fault handler. The TravelOperationsProcess is a long running process. In V5.1, its Compensation Sphere value was set to Not Supported. WebSphere Process Server supports BPEL compliant Compensation Handlers instead of the V5.1 Compensation Sphere. To preserve the V5.1 behavior to have compensation disabled in the TravelOperationsProcess (by selecting Compensation Sphere Not Supported), the Migration wizard created this fault handler.

c. Look at the Assign activity and Snippet activity implementations to ensure that the migration worked correctly, as shown in Figure 11-35 on page 275.
d. Repeat step c in the Prepare Booking activity.

e. Look at the Approve Trip activity, which was migrated to an inline participating task. Open the Human Task editor for this task by selecting the Approve Trip activity. In the Properties pane, for Approve Trip, click **Open**. The human task editor for Approve Trip opens.

f. Inspect the task, and ensure that the staff settings, the staff provider settings, and the interfaces of the task are properly migrated, as shown in Figure 11-36.

g. Close the ApproveTripTask by clicking X on the top of the task editor pane.

h. When you finish inspecting the TravelOperationsProcess, close it by clicking the X on the top of the process editor.

i. Click **Yes**, when asked for saving the file.

j. In the Assembly Diagram, select the **travelPort** import to display its properties in the Properties pane. Inspect its properties.

Notice, that in the Bindings section, the information needed to call the TravelAgencyProcessExport in the TravelAgency module is stored.
Note: Calling another process using this procedure (store the information of the other process in SCA binding) is so called early binding, which means that the TravelOperationsProcess always calls that specific TravelAgencyProcess, although there might be a more current version of the TravelAgencyProcess in the system. Take advantage of versioning. Remodel that part in the Assembly Diagram for late binding.

For more information, refer to IBM WebSphere Integration Developer Information Center:


k. In the Assembly Diagram, select the EmailSenderBean import to display its properties in the Properties pane. Inspect its properties.

l. In the Assembly Diagram, double-click the EmailSenderBeanJavaMed component. Inspect the file. Close the EmailSenderBeanJavaMedImpl.java file by clicking the X on the top of the Java editor.

m. In the Assembly Diagram, select the TravelOperationsProcessExport export to display its properties in the Properties pane. Inspect its properties. This export can be used to call the TravelOperationsProcess from a different SCA module.

n. In the Assembly Diagram, select the TravelOperationsProcess_JMSExport export to display its properties in the Properties pane. Inspect its properties. This export can be used to call the TravelOperationsProcess through JMS. Additional configuration and changing the calling client is necessary to use this interface because it is not compatible with the Process Choreographer JMS API that is in WebSphere Business Integration Server Foundation. This export is not needed in our scenario; therefore, you can delete it. In the Assembly Diagram, right-click the TravelOperationsProcess_JMSExport, and click Delete.

o. Save the Assembly Diagram. It now looks similar to Figure 11-37 on page 277.
p. Close the Assembly Diagram by clicking the X on the top of the Assembly Diagram editor.

q. Open the module dependency editor to ensure that the dependencies are set correctly. To do this, double-click **Dependencies** within the TravelOperations module in the **Business Integration** view.

The Dependencies editor opens. Observe that the TravelOperations module has a dependency to the CommonArtifacts Business Integration Library and a dependency to the EmailSender Session Bean Client (Figure 11-38 on page 278). Both of these dependencies also existed in the former WebSphere Studio Application Developer Integration Edition project.
r. There is a new dependency to the websphere_default_messaging_provider project, which is used in conjunction with the JMS export that you deleted in the preceding steps. Therefore, this dependency is not necessary anymore. Delete it by selecting it, and clicking Remove.

s. Close the Dependencies editor by clicking the X on the top of the Dependency Editor. Click Yes, when asked to save the changes.

This completes your inspection of the TravelOperations module.

20. Clean and build all projects in the workspace. From the menu, select Project → Clean. In the Clean dialog, select Clean all projects, and ensure that Start a build immediately is selected. Click OK. Make sure there is no error in the Problems view.
Deploying the applications
The final step in the migration process is to export the application EAR files and deploy them to WebSphere Process Server:

1. Click **File → Export**.
2. Select the EAR file, and click **Next**.
3. Specify the EAR Project as `EmailSenderEAR`, and specify a destination. Click **Finish**.
4. Repeat the first three steps for the other three projects: `TravelAgencyApp`, `TravelOperationsApp`, and `TravelClientEAR`.
5. Install all of the four applications by using the administrative console.
6. Start the applications on the server.

### 11.3.3 Verification

In this section, we explain how to test the migrated scenario in WebSphere Process Server and verify the results. We use the Web client and BPC Explorer to test the scenario.

To test the migrated scenario in WebSphere Process Server:

1. Open a supported Web browser. In this scenario, WebSphere Process Server uses the default port 9080. Type the following URL:

   http://localhost:9080/TravelOperationsClient/RequestSubmit.jsp

2. Complete the request form (Figure 11-39 on page 280), and then click **Submit**.
3. Open another Web browser.

4. To start the BPC Explorer, type the following URL:

   http://localhost:9080/bpc/

5. In the BPC Explorer navigation pane, Figure 11-40, under Task Instances, click My Tasks. From the list in the right pane, select the Approve Trip task, and click Work on.
6. In the Task Output Message field, type true, and click Complete. Because true is entered as the result of the ApproveTrip activity, the TravelAgencyProcess is invoked.

7. Look at the SystemOut.log of the server, and see whether the processing of the TravelAgencyProcess is successful. The SystemOut.log is in WPS profile directory/logs/server name. Compare this output with the output of the V5.1 process shown in Example 11-1 on page 240.

   If a valid SMTP host was entered in the TravelAgencyProcess, the SystemOut.log can also verify whether the e-mail was sent by the WebSphere Process Server, as shown in Figure 11-41.

   ![Email Received](image)

   Figure 11-41  E-mail received from the TravelAgencyProcess running in WebSphere Process Server

   By repeating step 1 on page 279 through step 6, but typing false as the result of the ApproveTrip activity, the TravelAgencyProcess is not invoked. Instead, the TravelOperationsProcess calls the EmailSender bean to send a rejection e-mail.

8. Look at the SystemOut.log of the server, and see whether the processing of this step in the TravelOperationsProcess is successful. Compare this output with the output of the V5.1 process shown in Example 11-2 on page 241.

   You completed verification in WebSphere Process Server of the migrated TravelOperations project in WebSphere Integration Developer.

   **Testing:** In real migration projects, you must perform extensive testing to verify the outcome of the migration.

### 11.4 Extended Travel Operations scenario

After migrating the Travel Operations scenario to WebSphere Process Server, a business rule is added to the scenario in WebSphere Integration Developer. It is invoked by the TravelAgency process to calculate the per-diem rate of the
traveler. Then the completed application is deployed to WebSphere Process Server to show how it works in the WebSphere Process Server environment, how to modify the business rule by using Business Rule Manager, and how to monitor the business process in Business Process Choreographer Explorer reporting function.

11.4.1 Implementation

In WebSphere Studio Application Developer Integration Edition, the Java code for providing the per-diem rate to the traveler is written into the TravelAgency project. Whenever the per-diem rate changes, you must update and redeploy the TravelAgency application to WebSphere Business Integration Server Foundation, which is not a simple task.

In this section, we add a business rule to the migrated TravelAgency project in WebSphere Integration Developer. The business rule replaces the Java code to perform the per-diem function. After deploying the changed TravelAgency project to WebSphere Process Server, if the per-diem rate must be changed, you can change the rate in Business Rule Manager at WebSphere Process Server run time instead of changing any application code and doing the redeployment.

To add the business rule:

1. Open the WebSphere Integration Developer workspace which is available at the end of section 11.3.2, “Implementation” on page 243.

2. In the Business Integration perspective, right-click the TravelAgency project, and select New → Interface.

3. As the folder name, enter com/ibm/itso/PerDiemBR. As the name, enter PerDiemRulesInterface. Click Finish.
4. In the interface editor:
   a. Click **Add Request Response Operation**.
   b. Change the action name to `getPerDiem`.
   c. Change Input name to `location`, and verify that Type is `string`.
   d. Change Output name to `PerDiem`, and verify that type is `int`.

   Figure 11-42 shows the completed interface.

   ![Figure 11-42 Interface of the business rule](image)

5. Open the Assembly Diagram of the TravelAgency module. On the palette part of the Assembly Diagram, select the **Rule Group** item. Click the Assembly Editor canvas to create a new rule group.

6. Rename the rule group `PerDiemRules`. Right-click `PerDiemRules`, and select **Add → Interface**. Select `PerDiemRulesInterface`, and click **OK**.

7. Right-click `PerDiemRules`, select **Generate Implementation**. Select the `com/ibm/itso/PerDiemBR` folder, and click **OK**.

8. In the rule group editor, click the `getPerDiem` operation of `PerDiemRulesInterface`.

9. In New Operation Details, click **Create Operation Definition**.

10. Click **Enter Rule Logic**, and select **New decision table**, as shown in Figure 11-43 on page 284.
11. Select the `com/ibm/itso/PerDiemBR` folder, accept the name `getPerdiem`, and click **Finish**.

12. Select the first column of Conditions to location, and select the first column of Actions to PerDiem.
13. Click the first condition part, and type a String value of New York, as shown in Figure 11-44.

![Decision Table and Condition Example](image)

**Figure 11-44 Adding a string to the condition part**

14. Type a String value of San Francisco for the second condition part.
15. Type a Number value of 200 for the first value part.
16. Type a Number value of 180 for the second value part.
17. Select the second condition part, and select **Add Condition Otherwise**. Type a Number value of 150 for the value part.
18. Right-click all of the condition and value parts, and select **Convert to Template**. Figure 11-45 shows the completed decision table.

![GetPerDiem decision table](image)

**Figure 11-45  GetPerDiem decision table**

19. In the Business Process Editor, open TravelAgencyProcess. Add a **Reference Partner** with **PerDiemRulesInterface**.

20. In the Business Process Editor:
   
   a. Click the **Invoke** activity icon on the Palette.
   b. Add the **Invoke** activity icon before the **Prepare Email Body** activity.
   c. Rename the **Invoke** activity icon to **Get PerDiem**.

21. In the Properties view of **Get PerDiem**, click the **Details** tab:

   a. For Partner, click **Browse**, and then click **New**. Type **PerDiemRulesInterfacePartner** as the Partner name, select the interface **PerDiemRulesInterface**, and click **OK**, as shown in Figure 11-46 on page 287.
Figure 11-46 create Reference Partner PerDiemRulesInterfacePartner

b. Clear the **Use data type variables mapping** box.

c. For Request, click **Browse**, click **New**, type PerDiemRequest as the new variable name, and click **OK**.

d. For Response, click **Browse**, click **New**, and type PerDiemResponse as the new variable name. Click **OK**. Figure 11-47 on page 288 shows the new invoke.
22. In the Business Process Editor, on Palette, drag the Assign icon to the editor canvas, and drop it after the Prepare Email activity, rename it to Prepare PerDiem Request.

23. In the Business Process Editor, click the Prepare PerDiem Request activity. In the Properties view, click the Details tab, change Assign From to TravelRequest to, and change Assign To to PerDiemRequest getPerDiemParameters location, as shown in Figure 11-48.

24. Select the Prepare Email Body activity. Click the Details tab, and change the Java code (Example 11-11) to get the per diem from the response of invocation of the business rule.

Example 11-11 Modified Java code in the Prepare Email snippet

```java
com.ibm.websphere.bo.BOFactory boFactory =
    com.ibm.websphere.bo.BOFactory
com.ibm.websphere.sca.ServiceManager.INSTANCE
   .locateService("com/ibm/websphere/bo/BOFactory");
String body = new String();
if (TravelRequest == null) {
    TravelRequest =
        boFactory.createByType(getVariableType("TravelRequest");
}
```
String to = TravelRequest.getString("to");
body = "Your trip to ";
body = body + to;
body = body + " has been booked successfully.\n";
body = body + "You will get Per Diem for ";

if (PerDiemResponse == null) {
    PerDiemResponse =
    boFactory.createByType(getVariableType("PerDiemResponse"));
}

body = body + PerDiemResponse.getString("PerDiem");
body = body + "/day.\nYour Travel Agency";
if (EmailSendRequest == null) {
    EmailSendRequest = boFactory
    .createByType(getVariableType("EmailSendRequest"));
}
    EmailSendRequest.set("email", body);

Now the updated TravelAgencyProcess process looks like Figure 11-49 on page 290.
Figure 11-49  Updated TravelAgencyProcess
25. In the assembly diagram of TravelAgency, wire TravelAgencyProcess to PerDiemRules. Figure 11-50 shows the updated assembly diagram.

![Figure 11-50](image)

26. Save all changes. Rebuild the workspace. There are no errors in the workspace.

**Deploying the applications**

The final step in the migration process is to export the application EAR files and deploy them to WebSphere Process Server:

1. Click File → Export.
2. Select the EAR file, and click Next.
3. Specify the EAR project as EmailSenderEAR, and specify a destination. Click Finish.
4. Repeat steps 1 to 3 for the other three projects: TravelAgencyApp, TravelOperationsApp, and TravelClientEAR.
5. Install all four applications using the administrative console.
6. Start the applications on the server.
11.4.2 Verification

In this section, we explain how to test the extended applications with the added business rule in WebSphere Process Server.

Verification to the added business rule

To test the extended applications with the added business rule in WebSphere Process Server:

1. To verify whether the extended scenario works, follow the steps we explained in 11.3.3, “Verification” on page 279. Figure 11-51 shows the notification mail, which indicates that the per diem is 180/day.

![Email showing per diem is 180/day](image)

2. Open a supported Web browser. In this scenario, WebSphere Process Server uses the default port 9080. Enter the following URL to open the Business Rule Manager:

   http://localhost:9080/br/

3. For the getPerDiem decision table, click **Edit**, as shown in Figure 11-52 on page 293.
4. In the Decision Table section, Figure 11-53, change the PerDiem value of San Francisco to 190. Click **Save**.

5. In the left pane, click **Publish and Revert**, and ensure that the change is selected. Click **Publish**. See Figure 11-54 on page 294.
6. Complete the steps that are explained in 11.3.3, “Verification” on page 279, to verify whether the modified business rule works. Figure 11-55 shows the notification mail. Notice that the per diem is now 190/day.

Using Business Process Choreographer Explorer to check the process status

In WebSphere Process Server V6.2, you can monitor your business process using the Business Process Choreographer Explorer's reporting function. Using the Business Process Choreographer Explorer reporting function is optional; however, before you can use it, you need some extra configurations. See the WebSphere Infocenter at:

To get the business process status report using the Business Process Choreographer Explorer reporting function:

1. Open a supported Web Browser. In this scenario, WebSphere Process Server uses the default port 9080. Enter the following URL:

   http://localhost:9080/bpc

   On the left-top corner of the page, click the **Report** tab. Figure 11-56 shows the overview page of the reporting function, which includes the data that is available for lists, charts, and reports.

   ![Figure 11-56 Overview: Business Process Choreographer Explorer reporting function](image)

2. In the left pane of this page, click **Processes**. In the Search Criteria pane on the right, select the time period, and click **Continue**.

3. Select **TravelOperationsProcess template**, and click **InstanceSnapshot**. Figure 11-57 on page 296 shows four instances of TravelOperationProcess in different status.
4. In the left pane of this page, click **Process snapshot**. In the Search Criteria pane, select the time period, and click **Continue**.

5. Select the **TravelOperationsProcess** template, and click **Continue With Selected**. Figure 11-58 on page 297 shows an instance snapshot of TravelOperationProcess.
11.5 Travel Booking scenario

The components of the Travel Booking scenario are:

- A set of business items that define the data inside of the model:
  - Customer entity: The booking customer.
  - Booking Order: The purchase order of the booking process.
  - Route: Each route in the trip. The booking order has some route lines.
  - Printed Ticked Information: An entity to store the data of the ticket.
  - Booking Order Id: The number or identification of an internal booking order.

- A set of human resources to associate with each human task of the process.
- A Booking Order process that defines the main process of this scenario
- A set of tasks in the Booking Order process:
  - Booking Order Form task, in which the customer completes a Booking Order Form, creates a booking order, and starts the booking process
  - Approval Booking task, in which an approval person approves the booking
  - Ticket Ready task, which issues the ticket to the customer
  - Customer feedback task, which collects feedback from the customer

 Files you can download: The files that you can download for this scenario are available as explained in Appendix A, “Additional material” on page 373

11.5.1 Preparation

In this section, we show how the model was created in WebSphere Business Integration Modeler V5. We also explain how to deploy it to the WebSphere Business Integration Server Foundation environment and verify that the model is working. This process can confirm that we have a good starting point.

Import the model into WebSphere Business Integration Modeler 5.1.1:

1. Go to Appendix A, “Additional material” on page 373, to download the material.
2. Extract the source file, and look for the MigrationModelingScenario.zip file.
3. In WebSphere Business Modeler, to import the model, select File → Import.
4. In the File import wizard, for Import Type, click WebSphere Business Integration Modeler, and for Project Type, select WebSphere Business Integration Modeler V5 Project.
5. Choose the folder that contains the MigrationModelingScenario.zip file, and select the file.
6. Import a new project.
   The new service project uses the default Catalog name for Business Items and Process catalogs, and the imported project uses a different name.
7. In the message window that opens, when you are prompted about combining the catalogs, click Yes for this merge to occur.
8. In the results window, Figure 11-59, review the options, and click **Finish**.
9. Review the business items. Figure 11-61 on page 301 shows the definition of the Customer entity.
The Booking Order business item contains the following elements, as shown in Figure 11-62 on page 302:

- A purchase order number to be provided for the customer
- The booking date
- Customer
- Route to flight information

The cardinality of this element can be 1..n, but to make an easy test in the BPC Explorer that is installed with WebSphere Business Integration Server Foundation, we defined a fix for two lines in the order. If the multiplicity is changed to n, the migration can be continued without testing the process in WebSphere Business Integration Server Foundation because it performs without problems in WebSphere Process Server.
The Booking Order business process consists of the activities, as shown in Figure 11-63.
Figure 11-64 shows the Booking Order process opened in the WebSphere Business Integration Modeler V5.1.
Running the Booking Order process in WebSphere Business Integration Server Foundation

To run the process in WebSphere Business Integration Server Foundation, import the saved model project to WebSphere Studio Application Developer Integration Edition:

1. Import the process:
   b. In the Import wizard, Figure 11-65, click Select All to select all of the model artifacts, and click Finish.

![Import wizard](image)

Figure 11-65  Importing a process
2. Generate the Java service for the Reservation System activity.

3. Open the service project, and select the `BookingOrderInterface.wsdl` file. Right-click the file, and select **New → Build from Service** to create a new service, as shown in Figure 11-66.

![Build from Service](image)

*Figure 11-66  Implementing Reservation System activity: Build from Service*

4. In the new service skeleton wizard, select a Java service skeleton.
5. In the Service Skeleton window, Figure 11-67, in the Port type name, select **ReservationSystemPT**. Click **Next**.

![Service Skeleton Window](image)

Figure 11-67  Implementing Reservation System activity: Service port type definition

6. In the next window, click **Finish**.

7. Open the Java skeleton to add the code in Example 11-12.

**Example 11-12  ReservationSystem activity of the Java implementation**

```java
/**
 * sendReservationSystem_InputCriteria
 * @generated
```
public Businessitems.data.BookingOrderId sendReservationSystem_InputCriteria(Businessitems.data.BookingOrder argBookingOrderFormPart) {
    // user code begin {method_content}
    int bookingNumber =
        argBookingOrderFormPart.getBookingOrderNumber();
    System.out.println(" Starting the reservation invocation");
    System.out.println(" booking number= " + bookingNumber);
    backendProcess(argBookingOrderFormPart.getFlight1());
    backendProcess(argBookingOrderFormPart.getFlight2());

    System.out.println(" Reservation system ready");
    BookingOrderId bookingOrderId = new BookingOrderId();
    bookingOrderId.setId("id:" + bookingNumber);
    return bookingOrderId;
    // user code end
}

/**
 * Calling the backend reservation system
 * @param orderLine1
 */

private void backendProcess(Route route) {
    // TODO the backend process in the reservation system
    System.out.print(" reservation Flight: " +
    route.getFlight());
    System.out.println(" seats reserved= " +
    route.getPassengers());
}

8. Generate the deploy code. Right-click the BPEL element, and select Enterprise Service → Generate Deploy Code, as shown in Figure 11-68 on page 308.
9. In the Generate BPEL Deploy Code window, shown in Figure 11-69 on page 309:

   a. In the Referenced Partners section, select 
      ReservationSystemPTJavaService.wsdl and the service and port that are already created.

   b. In the Process Settings section, set the Staff provider JNDI name to bpe/staff/everybodyconfiguration.

   c. Click OK to generate the deploy code.
Figure 11-69  Deploying code generation: Partner selection
10. To deploy, add the project to the server, and create tables and data sources:
   a. Select **Server → Add and remove projects**, and add the project.
   b. Click **Server → Create tables and data sources**.

   Figure 11-70 shows the tables that were created on the server.
11. Start the server, and launch the Business Process Web Client by selecting **Server → Launch Business Process Web Client**. Figure 11-71 shows the Process Navigation in WebSphere Business Integration Server Foundation.

![Figure 11-71   Process navigation: WebSphere Business Integration Server Foundation](image)

You can use the Business Process Web Client to test and verify that the process is working as expected.

### 11.5.2 Implementation

To implement the same process in WebSphere Process Server Version 6.2, we start with migrating the WebSphere Business Integration Modeler V5 model to WebSphere Business Modeler V6.2, and then we improve the business process model using the new features offered in WebSphere Process Server V6.2. Finally we export the updated model to WebSphere Integration Developer for further development and implementation. After we finish the implementation, and testing, we deploy the completed solution to WebSphere Process Server V6.2.
Migrating the WebSphere Business Integration Modeler V5 model

To migrate the WebSphere Business Integration Modeler V5 model to WebSphere Business Modeler V6.2:

1. Extract the compressed scenario file that you downloaded, as explained in Appendix A, “Additional material” on page 373. Locate the V5 model compressed file, MigrationModelingScenario.

2. Open WebSphere Business Modeler V6.2 with a new empty workspace. Select File → Import to open the file import wizard.

3. In the Import - Select window, Figure 11-72, expand WebSphere Business Modeler, and select WebSphere Business Modeler Import. Click Next.

![Figure 11-72 Selecting WebSphere Business Modeler Import](image)
4. In the Import - Select type window, select **WebSphere Business Modeler project (.mar, .zip)**, and click **Next**.

5. In the Import options window, Figure 11-73:
   a. For Source directory, click **Browse**, and select **MigrationModelingScenario.zip** as the file to import.
   b. For Target project, click **New** to create a new target project.
   c. In the Create new business modeling project window, for New project name, type **TravelBooking**, and click **Next**.
   d. In the Specify default editor for processes in this project window, Figure 11-74 on page 314, since we import the old model project, select **Free-form layout** for the new modeling project, and click **Finish** to return to the Import options window.

![Figure 11-73 Set import option for WebSphere Business Modeler](image-url)
e. Back in the Import options window, click **Next**.
6. In the Import preview window, Figure 11-75, expand **Predefined types**, **Business items**, or **data** to check the import elements. When you finish checking the import elements, click **Finish** to start importing the model into WebSphere Business Modeler.

![Figure 11-75  Model import preview](image)

Because of importing an early version of model project, WebSphere Business Modeler shows an alert message.

7. In the alert message window, Figure 11-76, click **OK** to start the migration of the model project.

![Figure 11-76  Model version detection alert](image)
After you import the model into the WebSphere Business Modeler, check the logs for any error or warning messages in the Error view.

**Migration process log**
The migration tool from WebSphere Business Integration Modeler V5 to WebSphere Business Modeler V6 generates a log file in the ./metadata/WBModelerMigration.log project directory. The log file contains any errors or warning messages during the migration process. If the log is empty, it indicates a successful migration.

**Verifying the migrated model**
After the migration, look at the migrated model. Figure 11-77 shows the projects that WebSphere Business Modeler creates during the migration process. Expand the **TravelBooking** project, expand the **Business items**, **data**, and **Resources** to examine the data items that are defined in the model and the resources for the human task.

![TravelBooking project tree view](image)
Double-click **Booking Order** to open the Booking Order process in WebSphere Business Modeler, as shown in Figure 11-78.

![Booking Order process model](image)

*Figure 11-78  Booking Order process model*

**Improving the model using WebSphere Business Modeler**

After we successfully migrate the business process model, we can use new features offered in WebSphere Business Modeler V6.2 and WebSphere Integration Developer V6.2, further improving the current business model.

**Creating the human task**

In WebSphere Business Modeler V6, you can define the human task. In this scenario, we assume that **Booking Order Form**, **Approval Booking**, **Ticket Ready**, and **Customer Feedback** are human tasks.

To convert the normal task activity to a human task:

1. Right-click the selected task, and select **Convert → Local Human Task**, as shown in Figure 11-79 on page 318.

If the human task is required in several business processes, create a global (reusable) human task. If the human task is required in a particular process only, create a local human task.

In the Travel Booking scenario, because activities in the human task are simple and used only in the Booking Order process, the local human task fits well here.
2. In the Convert to a local human task dialog windows, as shown in Figure 11-80, define the name of the human task and the descriptions of the task, and then click OK.

Figure 11-80  Convert to a local human task

Figure 11-81 on page 319 shows the results after we convert Booking Order Form, Approval Booking, Ticket Ready, and Customer Feedback to human tasks.
Adding input for the CreateBookingOrder human task

In WebSphere Integration Developer 6.2, the tools can generate the client applications for testing the human task. The human task client generation wizard only supports the human task that has the input data. Because the CreateBookingOrderHT is the very first task in the process, and by default, it did not have the input data to the task. To utilize the human task client feature that is offered in WebSphere Integration Developer, we must add the input to the first human task.

To add new input data for CreateBookingOrderHT task:

1. Click the edge of the process diagram, and draw a line to the CreateBookingOrderHT task as shown in Figure 11-82.

2. Click the CreateBookingOrderHT task. Locate the Attribute view, click the Input tab, and change the input variable name to OrderNumber and the data type to Text, as shown in Figure 11-83 on page 320.
Creating electronic forms

You can create electronic forms for people to use when they perform a human task. Forms represent the electronic forms that people use to complete a human task. Any forms that you create can be reused with any human task. As in this scenario, we can create a Booking Order Form so that a customer can fill it in and pass it to Approval to update the form for approval. The electronic form is passed to the next activity for further process.

Figure 11-84 on page 321 shows how we can create the form from human task output data.
To create the form from human task output data:

1. Select the `CreateBookingOrderHT` human task, and click the `Attributes` view.

2. Click the `Forms` tab. Because the Booking Order Form task did not have input, clear the `Use the input form as the output form` option, and click `New` to create the forms.
3. In the Create a Form window, Figure 11-85, add the form name and descriptions for the form, and click Finish.

![Create a form window](image)

*Figure 11-85  Create a form window*

Any forms that you create are displayed in the Processes catalog of the Project Tree view, as shown Figure 11-86 on page 323.
We can check the form that we created using the Lotus Forms Viewer, as shown in Figure 11-87.
For the Booking Approval human task, we can reuse the same form that we just created. When you associate a form with a human task, the task inputs, outputs, or both inputs and outputs are based on the form data.

1. In the Attributes view, Figure 11-88, locate the Forms tab, and click the Approval Booking task.

2. Select the Use the input form as the output form option, and then click Browse to select the form in the browse window.

3. Click OK to finish.

Figure 11-88 Associating the form with a human task

**Updating human tasks**

The BookingOrder Business process contains four human task and one external task. If you want to use WebSphere Integration Developer V6.2 to generate the human task client interface, check the human task to make sure that it contains both input and output data.

Checking TicketReadyHT and CustomerFeedbackHT, they all convert form standard task to human task. TicketReadyHT did not have the output data, and CustomerFeedbackHT did not have input data.
To add the missing input and output data to the human task:

1. In the Attributes view Output tab, highlight the TicketReadyHT, click the empty Associate Data field for the ticketOutput variable, and then click ... as shown in Figure 11-89.

2. In the Select type window, select PrintedTicket as the output data type, as Figure 11-90 on page 326, and then click OK.
3. In the Attributes view Input tab, highlight the CustomerFeedbackHT, click the empty Associate Data field for the customerInput variable, and then click the ... icon.

4. In the Select type window, select PrintedTicket as the output data type, and click OK to close the window.

Now the model migration and enhancement process are complete. Save the model, and check the error view for any possible errors.
Exporting to WebSphere Integration Developer

After the model migration and enhancement is complete, we can export the model to WebSphere Integration Developer using the file export wizard, and then use the WebSphere Integration Developer to add the detailed implementation and test it.

To export the model:

1. In the WebSphere Business Integration Modeler, click **File → Export**.
2. In the Export - Select window, select **WebSphere Business Modeler Export**, and click **Next**.
3. In the WebSphere Business Modeler Export window, select **WebSphere Integration Developer**, and click **Next**.
4. In the Export options window, Figure 11-91, select the export target directory, and select the **Export entire project and related projects** option. In the project selection view, select the **TravelBooking** project. Click **Next**.

![Figure 11-91 Export options window](image-url)
5. In the export details window, Figure 11-92, from the Select the export field, select the Model+Library option. Select the Export using the standard project interchange format for other environments option, and type the project interchange name. Click Finish to finish the export.

![WebSphere Business Modeler Export](image)

Figure 11-92 WebSphere Integration Developer export details window

Adding detailed implementation in WebSphere Integration Developer

After the model migration is completed, we must bring the migrated model into WebSphere Integration Developer, and then we can add the detailed implementation for the task, generated client, and testing the application.
**Importing into WebSphere Integration Developer**

Import the model project interchange file into WebSphere Integration Developer to do the final development and implementation:

1. In the WebSphere Integration Developer workbench, click **File → Import**.
2. In the Import window, select **Other → Project Interchange**, and click **Next**.
3. In the Project Interchange Contents window, select the project interchange compressed file that we just exported from WebSphere Business Modeler using the file browser, and click **Select All**. Click **Finish** to complete the import.

**Adding Java implementation**

After the projects are imported into WebSphere Integration Developer, an auto build starts, unless you disabled it. Wait for the build to finish, and then check the problem view in the bottom part of the workbench to see if there are any possible problems with the imported project.

In the Business integration perspective, expand the **TravelBooking** project, and double-click the assembly diagram to open it in the assembly editor, as shown in Figure 11-93.

![Figure 11-93  Assembly diagram for TravelBooking module project](image)
By default, the WebSphere Integration Developer V6.2 created the mediation module for the *ReservationSystem*. In this scenario, we want to use a Java implementation for it. Use the following steps to change the *ReservationSystem* to Java SCA component, and generate the Java implementation for it:

1. Rearrange the assembly diagram by right-clicking the blank area of the assembly diagram, and select the **Automatic Layout** in the context menu.
2. Right-click *ReservationSystem*, and select **Change Type → Java**.
3. Right-click *ReservationSystem* again, and select **Generate Implementation**
4. In the Generate Implementation window, click the new package and use *booking* as the package name, as shown in Figure 11-94. Click **OK**.

![Create Package in Generate Implementation wizard](image)

5. Click **OK** to generate the Java implementation.

   After the code generation finishes, the *ReservationSystemImpl.java* opens in the Java editor for you to add the detailed implementation.

6. In the Java Editor, locate the `DataObject InputCriteria(DataObject)` method at the bottom of the file. Add the code, as shown in Example 11-13.

   **Example 11-13 Sample code: InputCriteria method of ReservationSystemImpl.java**

   ```java
   public DataObject InputCriteria(DataObject reservationInput) {
     //TODO Needs to be implemented.
     int bookingId = reservationInput.getInt("bookingOrderNumber");
     System.out.println("The reservation request is for BookingOrder "+ bookingId + ".");
   }
   ```
DataObject customer = reservationInput.getDataObject("customer");

System.out.println("The customer is: " +
    customer.getString("name");
DataObject orderLine1 = reservationInput.getDataObject("flight1");
DataObject orderLine2 = reservationInput.getDataObject("flight2");
backendProcess(orderLine1);
backendProcess(orderLine2);

/*
 * If the routeLines has multiplicity n you can use:
 * 
 * for (Iterator iter = orderLines.iterator(); iter.hasNext();)
 * { 
 *    DataObject orderLine = (DataObject) iter.next();
 *    backendProcess(orderLine); }
 */

//the output is the purchaseOrderid
ServiceManager serviceManager = new ServiceManager();

BOFactory bof = (BOFactory) serviceManager
    .locateService("com/ibm/websphere/bo/BOFactory");

DataObject output =
    bof.create("http://TravelBooking/Businessitems/data",
        "BookingOrderId");
output.setString("id", "Booking: " + bookingId);
System.out.println("Reservation confirmed for: " +
    output.toString());
return output;
}

/**
 * Calling the backend reservation system
 * 
 * @param orderLine1
 */
private void backendProcess(DataObject orderLine) {
    // TODO the backend process in the reservation system
System.out.print("  | Booking flight: "+
        orderLine.getString("flight "));
System.out.println("  | Number of passengers: "+
        orderLine.getInt("passengers");
}

7. After you add the code, in the Java Editor, right-click and select **Source → Organize Imports**, and add **com.ibm.websphere.bo.BOFactory** to the import.

8. Save the code changes. Also save the changes in the assembly diagram, rebuild the project, and check the problem view for any possible errors.

After the build finishes, review the generated BPEL process by double-clicking the *BookingOrder* process in the assembly diagram. Figure 11-95 shows the BookingOrder process in the BPEL editor.

![BookingOrder BPEL diagram](image-url)
**Generating the human task client**

WebSphere Integration Developer V6.2 provides the client generation wizard to help you create the client application for the human task. It creates the basic user interface to start the process flow, create the new task, claim the open task, and complete the task. WebSphere Integration Developer human task client generator can generate various types of clients for human tasks. Integration developer can use the generated clients to customize a user interface through which you can interact with the tasks in the runtime environment. You can also use generated clients to quickly demonstrate a human workflow, for example as a proof of concept or prototype.

Table 11-6 shows the client types that you can use to test the human tasks and the prerequisites for it.

*Table 11-6  Human task client types and prerequisites*

<table>
<thead>
<tr>
<th>Client Type</th>
<th>Description</th>
<th>Prerequisite</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Lotus Forms Client</td>
<td>Using Lotus Forms you can easily integrate electronic forms with human tasks. You can generate a client based on forms that you specify in the User interface settings</td>
<td>Before you can deploy this user interface to a runtime environment, you must install the Lotus Forms Viewer and the Lotus Notes® Server API on each machine where the client will get deployed, which includes the WebSphere Test Environments and the WebSphere Process Server.</td>
</tr>
<tr>
<td>Business Space Client</td>
<td>Business Space is a new client platform through which a user can interact with an instance of a human task in the runtime environment. You can configure the visualization of this user interface using an HTML/Dojo file. When ported into WebSphere Process Server, the information in this file appears in the Task Information Widget area of Business Space.</td>
<td>You can create an HTML file to be used as a part of the Business Space user interface using WebSphere Integration Developer, but it can only be deployed to, and used in WebSphere Modeler Publishing Server, WebSphere Business Monitor, or WebSphere Process Server.</td>
</tr>
</tbody>
</table>
In this scenario, we use the business space client for testing our Travel booking application.

### Business space concepts

Business Space powered by WebSphere provides an integrated user experience for business users across the IBM WebSphere Business Process Management portfolio. Business Space provides a customizable and collaborative environment for monitoring, reviewing, and administering common business processes, such as human task flows, modeling, and performance indicators.

<table>
<thead>
<tr>
<th>Client Type</th>
<th>Description</th>
<th>Prerequisite</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Portal portlet</td>
<td>Select the page on which the portlet will be placed in the page hierarchy of IBM WebSphere Portal by setting properties in the user interface settings, or you can generate a portlet using the portlet generator.</td>
<td>To generate portlets, you need the Portal Toolkit. When you install WebSphere Integration Developer, you have the option to install the Portal Toolkit.</td>
</tr>
<tr>
<td>JavaServer Faces (JSF) client</td>
<td>The client generator for human tasks generates a JSF-based Web client that is useful for quickly prototyping human task processes or as a starting point for creating a custom client. The JSF client is generated based on data that is described in the interface that the human task implements and does not need any input defined in the User interface settings.</td>
<td>The JSF client generator is included in WebSphere Integration Developer. Use the JSF client generation wizard to generate the client application and deploy it to the WebSphere Process Server.</td>
</tr>
<tr>
<td>Business Process Choreographer explorer</td>
<td>This is the standard client that is delivered with this product. With this client, the task is delivered to the staff member through an HTML-based Web page. The look and feel of this Web page is determined by the JSP values that are specified in the User interface settings. You can modify these values as needed.</td>
<td>No prerequisites. Included in WebSphere Integration Developer.</td>
</tr>
</tbody>
</table>
Business Space is a browser-based graphical user interface that lets business users interact with content from products in the WebSphere Business Process Management portfolio. The business spaces that you create are collections of related Web content that provide you with insight into your business and the capability to react to changes in it. For more information about the business space, refer to the IBM infocenter at:


**Generating the Business Space client**

To generate the client for business space:

1. In the Business integration perspective, right-click TravelBooking project, and select **Generate Human Task User Interface**.

2. In the client generator selection window, select **HTML-Dojo pages or Lotus Forms for Business Space** as the client generator type, select all human tasks, and click **Next**, as shown in Figure 11-96.

3. Select **HTML** as the client page format, click **Browse** to select the web project to store the HTML page.

4. In the folder selection browser, click **New Web Project**.
5. In the New Dynamic Web Project wizard, set the client project name, and click **Finish** to create the new Web projects.

6. Click **No** on the message window, to stay in the business integration perspective.

7. The newly created client project folder should be updated in the client folder selection window. Verify it, and click **OK**.

8. Click **Finish** to generate the human task interface for business space.

After you complete the implementation and client generation, do a clean rebuild on the workspace, and check any errors.

### 11.5.3 Verification

In WebSphere Integration Developer, you can deploy modules to the test environment and work with a set of tools with which you can perform testing and verification of your application. For our Travel Booking application, we deploy the application and run it in the test environment. We use Business Space powered by WebSphere to manage business processes and human tasks and to work with assigned tasks.

To test and verify the TravelBooking business process, we must deploy the application to the WebSphere Integration Developer integrated test environment, and create a business space for testing the process and using the business space widgets to invoke the human tasks in the process.

**Deploying the application to the integrated test environment**

To deploy the application to the integrated test environment on WebSphere Integration Developer:

1. In the server view, right-click **WebSphere Process Server V6.1**, and select **Start** to start the server.

2. After the server is started, right-click the server, and select **Add and Remove Projects**.

3. In the Add and Remove Projects wizard, select **TravelBookingApp**, and click **Add**. Click **Finish**.

Check the server status and server log for any errors or problems of the application deployment. The application should start error free.
Creating a business for testing TravelBooking application

1. To make sure that your server and application start error free, right-click the server, and select Launch → Business Space.

2. In the Security Alert window, click yes since, for WebSphere Process Server V6.1, the security is enabled by default.

3. Login to Business Space using the same user ID and password that you used when you configured the Business Process Choreographer container. The default user ID is admin, and the default password is admin.

4. Create a new business space to test the TravelBooking application. Click Create New Business Space, as shown in Figure 11-97.

![Create new business space](image)

Figure 11-97  Create new business space

5. In the Create New Business Space wizard, as shown in Figure 11-98 on page 338, type the name of the space, choose Empty Space, and click OK.
6. After the new space is created, it displays in the Business Space Manager. As shown in Figure 11-99 on page 339, highlight **TravelBookingBusinessSpace**, and click **New Page** to create a new pages for the space.

**Widgets:** The new business space wizard offers numbers of business space templates. Most of them were targeted for use with WebSphere Business Modeler and WebSphere Business Monitor tools, so some of the widgets loaded in the template might not work in WebSphere Integration Developer environment.
Figure 11-99  Create a new page in the business space.

7. In the New Page wizard, Figure 11-100, type the name of the page, select **Page Layouts**, and click **OK**.

Figure 11-100  Create My Task Page for the business space

8. When the new pages are displayed in the business space, click **Add Widgets** and a palette displays that contains a list of available widgets. Click **Page Down**, as shown in Figure 11-101 on page 340, to find the widgets that you need to be added into the page.
9. Find the MyTask widgets from the palette, and drag-and-drop it on the My Task Pages, as shown in Figure 11-102.

10. After the widget is added to the page, you can minimize or restore the widgets in the pages using the Minimize/Restore button, as shown in Figure 11-103 on page 341.
11. In the same way, find the **Available Tasks** and **Task Information** widgets in the widget lists, and drag-and-drop them on to the My Task page. The page should look similar to Figure 11-104.

12. Click **New Page**, create the Create Task Page, and add Create Tasks, Tasks I Created and Task Information widgets into the page. The final page should look similar to Figure 11-105 on page 342.
Testing the application using business space widgets

The concept of business space is that the user can use the widgets to create the mashups to view the business content that they want to see in the page. In this scenario, we use the:

- Create Task widgets to create the BookingOrderHT task to initiate the TravelBooking process
- Available Task widgets to claim the task
- My Task widgets to work on the human task
- Task Information widget to display and update the related task data that the staff is working on

Creating the task and starting the BookOrder process

To start the new task using the create task widget:

1. Start the server and application, and launch the business space.
2. In the Create Task page, select Create Tasks widget.
3. In the task template list, select BookingOrder Task, and click Create, as shown in Figure 11-106 on page 343.
4. In the Task Information widget, fill in the input data, and click **Submit** to submit the created task, as shown in Figure 11-107.
Working on the human task using business space widgets

After the task is created, the BookingOrder process starts. The first human task, CreateBookingOrderHT, is placed in the available task list, which is waiting for staff to work on. Use the following steps to claim the available task, and complete the human task:

1. In the My Tasks page, open the Available Tasks widget.
2. Check on the CreateBookingOrderHT task, and click Accept, as shown in Figure 11-108.

![Figure 11-108 Claim task using Available Tasks widget](image)

3. As shown in Figure 11-109, open the My Tasks widgets, select the CreateBookingOrderHT option, click Edit to work on the assigned task.

![Figure 11-109 Work on accept the assigned task](image)
4. In the Task Information widget, complete the BookingOrderForm with test data, as shown in Figure 11-110.

![Figure 11-110 Fill in the bookingOrderForm](image)

5. After the bookingOrderForm is completed, the second human task, ApprovalBookingHT, is available for the booking approval officer to approve the booking request. Use the Available Tasks widget to accept the task.

6. Use the My Tasks widget to edit the task, check and verify the data using the Task Information widget, and click **Submit** to approve the booking request.

7. After the booking is approved, add the request route to the Reservation System to process the booking reservation. When the reservation is complete, the TicketReadyHT is available to work on. If the task does not show up in the Available task widget, click **Widget Refresh**, as shown in Figure 11-111 on page 346.
8. When the TicketReadyHT is available, claim the task, and edit the task.

9. In the Task Information widget, set the ticket delivery date, and complete the TicketReadyHT by clicking Submit, as shown in Figure 11-112.

10. When the TicketReadyHT completes, the CustomerFeedbackHT is ready to collect the customer's comments. Claim the available CustomerFeedbackHT, complete it in the Task Information by type in the feedback comments.
11. Check the Task I Started widget to see the status of the Booking Order task. Figure 11-113 shows that the task is completed without error.

![Completed task](image)

Figure 11-113  Completed task

12. Check the server log for any errors.

### 11.5.4 Deployment

Now that we fully implemented and tested the scenario, we must deploy the application to the WebSphere Process Server. First we must export the application in the Java EAR file format. This file format is used by the Java platform Enterprise Edition (Java EE) for packaging one or more modules into a single archive so that the deployment of the various modules onto an application server occurs simultaneously and coherently. It also contains XML files called deployment descriptors that describe how to deploy the modules. WebSphere Integration Developer organizes the artifacts using the Java EE EAR format and can export the application as an EAR file and deploy it to the WebSphere Process Server.

**Exporting the application as an EAR file**

To export the EAR from WebSphere Integration Developer:

1. In the workbench, click **File → Export**.
2. In the Export window, expand **J2EE**, and select **EAR file**. Click **Next**.
3. In the EAR Export window, Figure 11-114 on page 348, choose an EAR application and the export destination. Click **Finish**.
Deploying an application to WebSphere Process Server

After the EAR export is finished, move the EAR file to WebSphere Process Server, and use the Integrated Solutions Console, formerly known as the Administrative Console, to deploy and install the EAR file to the server. While the Integrated Solutions Console framework is the IBM proposed framework for connecting all of the administration tools, the Integrated Solutions Console is just one of the console tools within that framework. It is a browser-based interface that administers applications, services, and other resources at a cell, node, or server scope. You can use the console from stand-alone process servers and from deployment managers that manage all servers in a cell in a networked environment.

To deploy an application to WebSphere Process Server:

1. Start the Integrated Solutions Console in WebSphere Process Server by pointing your Web browser to the following URL:
   
   https://localhost:9044/ibm/console/logon.jsp

2. Login to the console using your process server administrator ID and password.
3. In the Integrated Solutions Console, Figure 11-115, in the left navigation pane, expand **Applications**, and click **Install New Application**. In the right pane, under Path to the new application, click **Browse** to locate the EAR file that you want to install. Click **Next**.

4. **Step 1: Select installation options**, Figure 11-116 on page 350, accept the default options, and click **Next**.
5. In **Step 2: Map modules to servers**, Figure 11-116, select *TravelBookingEJB*, and click **Next**.

6. In **Step 3 Summary**, Figure 11-116, review all of the options, and click **Finish**.

7. After the application is installed, and you see the messages in the console that indicate a successful installation, click **Save** to save the configuration in the server, as shown in Figure 11-117 on page 351.
8. Return to the Integrated Solution Console, in the left navigation pane, under Applications, click **Enterprise Applications**, as shown in Figure 11-118 on page 352.

9. In the right pane, you see all the applications that are already installed on the server. Notice that TravelBookingApp has a red X under Application Status, which indicates that it is not started yet. Select the **TravelBookingApp**, and click **Start** to start the application.
When you see the message box at the top of the console indicating that the application started successfully, you can start testing the application using the Business Space widgets as we did in the WebSphere Integration Developer.

11.6 Scenario summary

In this section, we review the two scenarios that we just finished.
11.6.1 Travel Operations scenario summary

For the Travel Operations scenario migration, we explained how to migrate a WebSphere Studio Application Developer Integration Edition application to WebSphere Integration Developer. We developed the scenario in WebSphere Studio Application Developer Integration Edition and then deployed and ran it on the WebSphere Business Integration Server Foundation environment. Next we migrated the scenario to WebSphere Integration Developer and then deployed and ran it on WebSphere Process Server.

In this scenario, we started by examining the scenario in WebSphere Studio Application Developer Integration Edition and running it on WebSphere Business Integration Server Foundation. We then migrated the scenario to WebSphere Integration Developer. After doing some manual tasks to finish the migration, we deployed and ran the scenario on WebSphere Process Server. Until this point, we performed the migration as-is. Next, we added the new features of WebSphere Process Server to the migrated scenario in WebSphere Integration Developer and deployed and ran the updated scenario on WebSphere Process Server.

In this scenario, we also demonstrated how to migrate a Web client project, EJB project, service project, and common artifacts Java project from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer. Additionally, we showed you how to add a business rule, a new feature of WebSphere Process Server, to the migrated scenario.

11.6.2 Travel Booking scenario summary

For the Travel Booking scenario migration, we used WebSphere Business Modeler V6.2 to migrate the business model directly. WebSphere Business Modeler provides powerful business process modeling, simulation, analysis, and reporting features to help you optimize the performance of your business processes. It is a key component of a service-oriented architecture (SOA) implementation. Whether you want to improve customer satisfaction with services that you offer or reduce costs through increased workforce productivity, the SOA life cycle begins with business process modeling as an input.

Business process models that are created with WebSphere Business Modeler help you to achieve important SOA planning and implementation goals:

- Capture the relationships among people, processes, and information.
- Build your business case by mapping strategic goals to process capabilities and by simulating process changes to assess impacts on costs, resource utilization, and cycle time prior to deployment.
Identify where skills are most needed.
Reduce risk by simulating market changes against current process models.
Accurately and effectively communicate requirements from the business to IT.

For detailed information about features and benefits of using WebSphere Business Modeler V6.2, see the WebSphere Information Center at the following address:


In this scenario, we started the migration by importing the V5 model. With WebSphere Business Integration Modeler V5, we can only define the staff resource for the business model, but the actual staff activities or human task must be defined in WebSphere Studio Application Developer Integration Edition.

With WebSphere Business Modeler V6.2, we can include human tasks in the process models to visually identify and document tasks that a system assigns a person to do. If a human task requires electronic forms for its completion, as in our Travel Booking scenario, we can create these forms based on the data flowing into and out of the human task. We can also associate a form that you create or import with a human task and even create a human task based on a form. For application development purposes, we can provide specifications for human tasks and hand them off to IT along with any associated forms for further implementation and development.

The benefit of starting the migration from the model is starting earlier in the architecture design stage. If the goal of migration to WebSphere Process Server is to adopt the SOA, then the best place to start is with migration of the business model.

By using an SOA, you can build, deploy, and integrate individual business functions and processes independently of the applications and computing platforms on which they run. As a key component of such an architecture, WebSphere Business Modeler helps bridge the gap between business objectives and process implementation.

An SOA approach separates business tasks (such as checking customer credit or opening a new bank account) into reusable components or services. It also separates the underlying IT that supports those tasks. You can combine and recombine these services as business needs change. Even when the supporting systems and technologies are quite different, SOA allows services to exchange information by using an open communication environment. This connection of interrelated business processes makes it possible to share information across an
enterprise and with customers, suppliers, and partners. Figure 11-119 shows the life cycle of developing an SOA solution. You can use WebSphere Business Modeler to complete the model stage of the SOA cycle and pass business process models to the next stage in the cycle. Now, you have a direct link from business process modeling to the implementation of software services. WebSphere Business Modeler also complements the SOA modeling capabilities of tools such as Rational Software Architect.

![SOA life cycle diagram]

*Figure 11-119  SOA life cycle*

In the Travel Booking scenario, we changed the four tasks in the Booking Order process into human tasks: Booking Order Form, Approval Booking, Ticket Ready, and Customer Feedbacks. We also added the electronic forms to the human task for handling the data exchanges in the process. After we brought the model into WebSphere Integration Developer, we added the Java implementation to the ReservationSystem and enhanced the business process by adding two Java snippets. Figure 11-120 on page 356 and Figure 11-121 on page 357 show the difference between the WebSphere Business Modeler model and WebSphere Integration Developer BPEL model. Other than the layout orientations, the two processes look similar.
Figure 11-120  Booking Order model in WebSphere Business Modeler V6.1.2
The process model from WebSphere Business Modeler runs from left to right and has four human tasks and a service invocation task. The process model from WebSphere Integration Developer runs from top to bottom. The activities are matched with the model from WebSphere Business Modeler.

![Booking Order BPEL process in WebSphere Integration Developer](image)

*Figure 11-121  Booking Order BPEL process in WebSphere Integration Developer*

After we completed the implementation, we ran the test for Booking Order process with the integrated test environment on WebSphere Integration Developer. When we satisfied the test result, we exported the application by using the EAR export and installed the application to WebSphere Process Server by using the Integrated Solution Console.
Troubleshooting

In this chapter, we discuss the troubleshooting for the migration from WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server. The chapter is divided into the following sections:

- “General troubleshooting” on page 360
- “Migration troubleshooting” on page 361
- “Scenario troubleshooting” on page 366
- “Advanced troubleshooting” on page 369
12.1 General troubleshooting

In this section, we provide information about how to troubleshoot a problem by using the logs from the WebSphere product. *Troubleshooting* is the process of finding and eliminating the cause of a problem.

Whenever you have a problem, the troubleshooting process begins by asking: What happened? A basic troubleshooting strategy at a high level involves the following actions:

- Recording the symptoms
- Recreating the problem
- Eliminating possible causes
- Using diagnostic or debugging tools
- Contacting IBM Support

Depending on the type of problem you have, whether it is with your application, server, or your tools, you might receive a message that indicates something is wrong. Always record the error message that you see. As simple as this sounds, error messages often contain codes that make more sense as you investigate the problem further. You might also receive multiple error messages that look similar, but that have subtle differences. By recording the details of each message, you can learn more about where your problem exists.

You can look in the following places to find more detailed information:

- The Problems view in the WebSphere Integration Developer Workbench
- The Server Consoles in the Workbench
- Log files in your workspace
  Look for .log files in your workspace .metadata directory.
- WebSphere server logs
  Look for *.log files in the *WPS_install*/*profiles/*profile_name*/logs/*server_name* folder
- Error message windows

For most problems, the error messages or log file can give a clear indication for the source of the problem. These items are the starting point for the troubleshooting process.
For more details, see the troubleshooting guides on the IBM Support Web pages:

- *Troubleshooting Guide for WebSphere Process Server*

- *Troubleshooting Guide for WebSphere Application Server*

## 12.2 Migration troubleshooting

In this section, we introduce the general troubleshooting methodology for the migration process.

### 12.2.1 Importing into WebSphere Integration Developer

If the source artifact migration fails while importing from WebSphere Studio Application Developer Integration Edition, you can find more information about the problem in the log file and Migration Results window.

**Log file**

If you receive the message shown in Example 12-1 while migrating the original WebSphere Studio Application Developer Integration Edition, check the WebSphere Integration Developer log file in the .metadata folder of the new workspace to see the details of the error message. If possible, resolve the cause of the error, delete the module that was created in the new workspace, and try the migration again.

*Example 12-1 Migration error message in the original WebSphere Studio Application Developer Integration Edition*

```
“Migration Error Message”
Reason: Fatal Migration Failure
Message: Contact your IBM Representative
```

**Migration Results window**

If the migration wizard completes without the message shown in Example 12-1, a list of information, warning, and error messages are displayed that signify that some portion of the service project could not be automatically migrated and that you must perform manual changes to complete the migration.

The Migration Results window (Figure 12-1 on page 362) is displayed if migration messages are generated during the migration process. In this window, the
migration messages that are generated during the migration process are displayed. By selecting a message from the upper message list, you can learn more information about that message in the lower Message Description pane.

To keep all messages for future reference:
1. Click **Generate ToDo’s** to create a list of ToDo tasks in the task view.
2. Click **Save as** to save the messages in a text file in the file system.
3. Examine each message to see if you need to take any action to immediately fix an artifact that could not be fully migrated.

To verify that a portion of the migration is complete:
1. Switch to the Business Integration perspective. Ensure that all processes and Web Services Description Language (WSDL) interfaces from the old service project are displayed in the new module.
2. Build the project, and fix any errors that prevent the project from building.

After you perform the manual migration steps that are required to complete the migration of the business integration application, export the application as an
EAR file and install it to a WebSphere Process Server, configuring the appropriate resources. Perform the manual migration steps that are required to migrate any client code, or generate new client code using WebSphere Integration Developer. Ensure that the client can access the application and that the application exhibits the same behavior as it did in the previous runtime environment.

See Chapter 6, “Products runtime comparison” on page 91, for the required manual migration steps.

### 12.2.2 Building the project in WebSphere Integration Developer

After you import the WebSphere Studio Application Developer Integration Edition workspace into WebSphere Integration Developer, the new modules are generated. To build the workspace:

1. From the menu, select **Project → Build Automatically**. Otherwise, click **Project → Clean** to build the workspace. After you build the workspace, some errors might be reported in the Problems view (Figure 12-2).

   ![Figure 12-2 Problems view in WebSphere Integration Developer](image)

2. Double-click the error message, or right-click the message, and select the **Go To** command. The place where the error occurs is displayed in the main view. Also a red flag for an error or a yellow flag for a warning on the error item is displayed.

3. Read the error message description, and check the properties of the error item.
Sometimes the problems are caused by a conflict with some compiled files. Select **Project → Clean** to clean all of the compiled files, and force WebSphere Integration Developer to recompile all files. The problem might be resolved.

**Classpath problem**
After migrating the WebSphere Studio Application Developer Integration Edition workspace into WebSphere Integration Developer, you might see the classpath problem (Figure 12-3). To resolve this problem, remove all classpath entries that refer to V5 .jar files or libraries, and add the JRE System Library and WPS Server Target libraries to the class path:

1. Right-click the project, and select **Properties**.
2. Go to the Java build path entry, and click the **Libraries** tab.
3. Select the JAR files that are displayed in the Problems view, and click **Remove**. Ensure that the JRE System Library and WPS Server Target libraries are present. Click **OK**.

![Properties for EmailSender](image)

*Figure 12-3  Java build path - Cleaning up the class path*

4. Clean and build all projects in the workspace:
a. From the menu, select **Project → Clean**.

b. In the Clean window, select **Clean all projects**, and ensure that **Start a build immediately** is selected. Click **OK**.

After the build completes, only warning messages remain in the Problems view (Figure 12-4).

![Figure 12-4](image)

**Figure 12-4** The J2EE projects in Interface Editor after classpath clean up

### 12.2.3 Deploying and running the process in WebSphere Process Server

While deploying or running the process in WebSphere Process Server, some problems might occur. The logs of WebSphere Process Server are the most pivotal key in diagnosing a problem. The logs are in the `WPS_install/profiles/profile_name/logs/server_name` directory.

Generally eight files are in this directory:

- **ServerName.pid**, the process ID of the Java process of WebSphere Process Server
- **SystemOut.log**, the log of all activity within the system
- **SystemErr.log**, a record of system errors
- **startServer.log**, a log of start server events
- **stopServer.log**, a log of stop server events
- **native_stdout.log**, a stdout of the Java process
- **native_stderr.log**, a stderr of the Java process
- **trace.log**, a log of all traced events within the system

The **SystemOut.log** and **SystemErr.log** are useful for diagnosing the problems when deploying and running the process.
12.3 Scenario troubleshooting

In this section, we discuss the problems that you may encounter in technical scenarios, causes, and solutions.

Technical scenarios use the following product versions:

- Windows XP Professional with Service Pack 2
- DB2 Universal Database v8.2
- WebSphere Studio Application Developer Integration Edition V5.1.1
- WebSphere Business Integration Server Foundation V5.1.1
- WebSphere Business Integration Modeler V5.1.1
- WebSphere Business Modeler V6.2
- WebSphere Integration Developer V6.2
- WebSphere Process Server V6.2

**Important:** Follow the same product version to replay the scenarios. When you encounter any problems, check the product version first.

java.lang.NullPointerException observed

In this section, we explain the problem, cause, and solution of the java.lang.NullPointerException exception.

**Problem**

The java.lang.NullPointerException exception is observed while testing the Java implementation for ReservationSystem in the TravelBooking scenario. Example 12-2 shows the logs in the WebSphere Process Server SystemOut.log.

**Example 12-2  SystemOut.log for the java.lang.NullPointerException exception**

```
BOCore W EClassifier "http://Businessitems/data#BookingOrderId" not found in ClassLoaderScope:
com.ibm.ws.classloader.CompoundClassLoader@23b223b2
Local ClassPath:
D:\WIDWS\WS61\booking\ws\TravelBooking_implEJB\ejbModule;D:\WIDWS\WS61\booking\ws\TravelBooking_impl;D:\WIDWS\WS61\booking\ws\TravelBooking_lib
Delegation Mode: PARENT_FIRST
[7/31/08 16:33:14:562 EDT] 0000004b ExceptionUtil E CNTR0020E: EJB threw an unexpected (non-declared) exception during invocation of method "transactionNotSupportedActivitySessionSupports" on bean "BeanId(TravelBooking_implApp#TravelBooking_implEJB.jar#Module, null)". Exception data: java.lang.NullPointerException
```
at
booking.ReservationSystemImpl.InputCriteria(ReservationSystemImpl.java: 61)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.ja
va:79)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccesso
rImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:618)
at
com.ibm.ws.sca.internal.java.handler.JavaReflectionAdapter$2.run(JavaRe
flectionAdapter.java:152)

---

**Cause**

Example 12-3 shows that the exception is caused by the InputCriteria method in ReservationSystemImpl.java, at line 61. Example 12-4 on page 368 shows the code for the ReservationSystemImpl.java with the line 61 shown in bold.

**Example 12-3  ReservationSystemImpl.java code segment**

```java
//the output is the purchaseOrderId
ServiceManager serviceManager = new ServiceManager();

BOFactory bof = (BOFactory) serviceManager
.locateservice("com.ibm.websphere.bo/BOFactory");

DataObject output =
bof.create("http://TravelBooking/Businessitems/data",
"BookingOrderId");
output.setString("id", "Booking: " + bookingId);
```

The cause of the problem is in the code. The namespace of the business object “BookingOrderId” does not match the definition that is in the data.xsd files. In the implementation code, we use the namespace http://Businessitems/data to create a new business object, named “BookingOrderId.” However, in the
data.xsd definition, the namespace for business object “BookingOrderId” is defined as http://TravelBooking/Businessitems/data. Example 12-4 shows the definition of the BookingOrderId business object in the data.xsd file.

Example 12-4  XSD definition of BookingOrderId business object

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://TravelBooking/Businessitems/data"
    targetNamespace="http://TravelBooking/Businessitems/data">
  <xsd:complexType name="BookingOrderId">
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="1" name="id" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

In Example 12-4, the namespace for BookingOrderId is actually "http://TravelBooking/Businessitems/data".

**Solution**

Change the ReservationSystemImpl.java code to use the correct namespace for creating the BookingOrderId business object.

**Exception J2CA0138E observed**

In this section, we explain the problem, cause, and solution of the observed J2CA0138E exception.

**Problem**

The J2CA0138E exception is observed during the deployment of the TravelOperationsApp application in the TravelOperations scenario. Example 12-5 shows the logs in the WebSphere Process Server SystemOut.log.

Example 12-5  SystemOut.log for the J2CA0138E exception

```
J2CA0138E: The MessageEndpoint activation failed for ActivationSpec
    sca/TravelOperations/ActivationSpec
    (com.ibm.ws.sib.ra.inbound.impl.SibRaActivationSpecImpl) and MDB
    Application
    TravelOperationsApp#TravelOperationsEJB.jar#ServiceSIBusMessageBean,
due to the following exception: javax.resource.NotSupportedException:
    CWSIV0757E: The destination sca/TravelOperations on bus
    SCA.SYSTEM.widCell.Bus was not found.
```
Cause
TravelOperationsApp is dependent on TravelAgencyApp.

Solution
Resolve the dependence by following the guidance in “Resolving dependencies between service projects” on page 232.

12.4 Advanced troubleshooting

For unpredictable problems, use the following steps as a guide for resolving them:

1. Search the message identification or related topic in the product information center or manuals.
2. Search the message identification or keyword in the product support Web site.
3. Contact IBM Support to help resolve the problem.

12.4.1 Information center and manuals

Sometimes the explanation of the error message can help you to understand the extra meanings of the message. Also, user response for a given error message is provided in the manual of some products. Follow the suggested user response to resolve the problems.

The following links are to the product's information center or manuals:

- WebSphere Business Integration Server Foundation V5.1 Information Center: http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp
12.4.2 Product support Web site

IBM has a support Web site for all products. On the support Web site, you can find fix packs, technotes, Authorized Program Analysis Reports (APARs), and other useful resources.

When you encounter problems, take the following actions:

1. Search for the problem solution with the keyword or the message identification on the support Web site. For many cases, this solution locates some APARs or technotes that are related to the problem.

2. Read the problem description of these APARs and technotes to verify whether it is the same problem. If it is the same problem, try the solution provided in the APAR or the technote.

The following links go to the products' information center or manuals:

- WebSphere Process Server V6 support Web site

- WebSphere Integration Developer V6 support Web site

- WebSphere Business Integration Server Foundation support Web site

- WebSphere Studio Application Developer Integration Edition support Web site

- WebSphere Business Modeler V6 support Web site
12.4.3 IBM Support

If you still cannot resolve the problem, contact IBM Support for help by going to the following Web page to contact IBM Support in your country or region:

http://techsupport.services.ibm.com/guides/contacts.html

In the following steps, we provide information to help you prepare and consider before you contact IBM Support:

1. Submit a clear problem description that includes the following details:
   - The software versions that were running when the problem occurred
   - Any logs, traces, and messages that are related to the problem symptoms
   - Whether you can recreate the problem, and if so, the steps that led to the failure
   - Whether any changes have been made to the system, for example, hardware, operating system, and networking software
   - Whether you are using any workaround for this problem
   If so, be prepared to explain the workaround when you report the problem.

2. Collect the necessary documentation for the problem analysis:
   - For the development toolkit, such as WebSphere Integration Developer, WebSphere Business Modeler, and WebSphere Studio Application Developer Integration Edition, few logs are recorded. Therefore, it is difficult to analyze the problem through logs. Record the detailed steps you took to reproduce the problem, which is a key for IBM Support to resolve the problem, for example, some window captures of the problem are quite helpful.

   - For some runtime products, such as WebSphere Process Server and WebSphere Application Server, error logs and traces are available to record the problem. The output of some special commands can also help to analyze the problem. On the support Web sites of these products, MustGathers are often provided. MustGathers are technote that provide a detailed document list that is required to analyze different problems. The following links are the MustGathers for WebSphere Process Server, WebSphere Application Server, and WebSphere Integration Developer:
   - WebSphere Process Server MustGather
• WebSphere Application Server MustGather
  http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599

• WebSphere Integration Developer MustGather
  http://www-1.ibm.com/support/docview.wss?rs=2308&uid=swg21260690
Additional material

In this appendix, we refer to additional material that you can download from the Internet.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:


Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select Additional materials and open the directory that corresponds with the IBM Redbooks form number, SG247673-01.
Using the Web material

The additional Web material that accompanies this book includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All.zip</td>
<td>All scenarios in Chapter 11</td>
</tr>
<tr>
<td>TravelOperations.zip</td>
<td>Travel Operations scenario in Chapter 11</td>
</tr>
<tr>
<td>TravelBookingScenario.zip</td>
<td>Travel Booking scenario in Chapter 11</td>
</tr>
</tbody>
</table>

How to use the Web material

Create a subdirectory (folder) on your workstation, and decompress the contents of the Web material zip file into this folder.
## Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APARs</td>
<td>Authorized Program Analysis Reports</td>
</tr>
<tr>
<td>ARM</td>
<td>Application Response Measurement</td>
</tr>
<tr>
<td>BPE</td>
<td>business process engine</td>
</tr>
<tr>
<td>BPEDB</td>
<td>Business Process Choreographer database</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Execution Language</td>
</tr>
<tr>
<td>BPM</td>
<td>business process management</td>
</tr>
<tr>
<td>CEI</td>
<td>Common Event Infrastructure</td>
</tr>
<tr>
<td>CLI</td>
<td>command-line interface</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions System</td>
</tr>
<tr>
<td>CoE</td>
<td>Center of Excellence</td>
</tr>
<tr>
<td>DBMSs</td>
<td>database management systems</td>
</tr>
<tr>
<td>DDL</td>
<td>Data Definition Language</td>
</tr>
<tr>
<td>EAR</td>
<td>enterprise archive</td>
</tr>
<tr>
<td>EIS</td>
<td>Enterprise Information System</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise JavaBean</td>
</tr>
<tr>
<td>ESB</td>
<td>enterprise service bus</td>
</tr>
<tr>
<td>FDCC</td>
<td>Federal Desktop Core Configuration</td>
</tr>
<tr>
<td>FDML</td>
<td>Flow Definition Markup Language</td>
</tr>
<tr>
<td>GUI</td>
<td>graphic user interface</td>
</tr>
<tr>
<td>HA</td>
<td>high availability</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>IE</td>
<td>Integration Edition (WebSphere Studio)</td>
</tr>
<tr>
<td>IIP</td>
<td>Integrated Install Package</td>
</tr>
<tr>
<td>ITSO</td>
<td>International Technical Support Organization</td>
</tr>
<tr>
<td>J2C</td>
<td>J2EE Connector architecture</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Enterprise Edition</td>
</tr>
<tr>
<td>J2SE</td>
<td>Java 2 Platform, Standard Edition</td>
</tr>
<tr>
<td>JAR</td>
<td>Java Archive</td>
</tr>
<tr>
<td>JCA</td>
<td>J2EE Connector Architecture</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service</td>
</tr>
<tr>
<td>JSF</td>
<td>JavaServer Faces</td>
</tr>
<tr>
<td>KPIs</td>
<td>key performance indicators</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>MDB</td>
<td>message-driven bean</td>
</tr>
<tr>
<td>ORB</td>
<td>Object Request Broker</td>
</tr>
<tr>
<td>PIID</td>
<td>process instance identifier</td>
</tr>
<tr>
<td>PMEs</td>
<td>programming model extensions</td>
</tr>
<tr>
<td>PMI</td>
<td>Performance Monitoring Infrastructure</td>
</tr>
<tr>
<td>PMR</td>
<td>problem management record</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>ROI</td>
<td>return on investment</td>
</tr>
<tr>
<td>RUP</td>
<td>Rational Unified Process</td>
</tr>
<tr>
<td>SCA</td>
<td>Service Component Architecture</td>
</tr>
<tr>
<td>SCM</td>
<td>software configuration management</td>
</tr>
<tr>
<td>SDO</td>
<td>Service Data Object</td>
</tr>
<tr>
<td>SLA</td>
<td>service level agreement</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>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>SOMA</td>
<td>service-oriented modeling and architecture</td>
</tr>
<tr>
<td>SPIs</td>
<td>service provider interfaces</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>UDDI I</td>
<td>Java Naming and Directory Interface</td>
</tr>
<tr>
<td>WAR</td>
<td>Web Archive</td>
</tr>
<tr>
<td>WBISF</td>
<td>WebSphere Business Integration Server Foundation</td>
</tr>
<tr>
<td>WID</td>
<td>WebSphere Integration Developer</td>
</tr>
<tr>
<td>WISF</td>
<td>Web Service Invocation Framework</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>WS-Security</td>
<td>Web service Security</td>
</tr>
<tr>
<td>XMI</td>
<td>XML Metadata Interchange</td>
</tr>
<tr>
<td>XML</td>
<td>Extended Markup Language</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
</tbody>
</table>
Glossary

activity  A unit of work or a building block that performs a specific, discrete programmatic task. An element of a process, such as a task, a subprocess, a loop, or a decision. Activities are represented as nodes in process diagrams. See also task.

adapter  An intermediary software component that allows two other software components to communicate with one another.

API  See application programming interface.

application client  In Java Platform, Enterprise Edition (Java EE), a first-tier client component that runs in its own Java virtual machine (JVM). Has access to some Java EE platform APIs, for example Java Naming and Directory Interface (JNDI), Java Database Connectivity (JDBC), Remote Method Invocation over Internet InterORB Protocol (RMI-IIOP), and Java Message Service (JMS).

application programming interface (API)  A source code interface that a computer system or program library provides to support requests for services to be made of it by a computer program. The software that provides the functionality described by an API is said to be an implementation of the API. The API itself is abstract, in that it specifies an interface and does not get involved with implementation details.

artifact  An entity that is used or produced by a software development process. Examples of artifacts are models, source files, scripts, and binary executable files.

Binding  Defines the kind of interaction, the caller, and the sender. It is used to connect different SCA modules together.

BO  A Business Objects are representations of data structure used in business processes.

BPE  Business Process Engine

BPEL  See Business Process Execution Language.


BPM  Business Process Management is a concept of management around aligning organizations with the wants and needs of their clients.

Breakpoint  Used as a point to stop the flow of a message in a message flow when the flow debugger is attached.

Broker  A set of execution processes that host and run message flows.

bus  Provides the possibility to transfer data between different nodes that are connected to a bus.

Business Model  A graphical representation of a business process, which is the starting point for development of a business process application.

business object  In a development or production environment, a set of attributes that represents a business entity, such as an invoice, and the definition of actions that can be performed on those attributes, such as the create and update operations. A representation of a data structure that is used in business processes.

Business process  Defines a flow of actions to solve a special task. It can be stored in a Business Process Execution Language (BPEL) file.
**Business Process Choreographer**  A component of WebSphere Process Server runtime architecture that provides support for business processes and human tasks. It offers a way to model your business process, based on the Web Services Business Process Execution Language (WS-BPEL) specification, and to model interactions that involve humans, such as human-to-human, human-to-machine, and machine-to-human interactions. Business processes and human tasks are exposed as services in a service-oriented architecture (SOA).

**Business Process Execution Language (BPEL)**  An open standard to store process definitions in a file.

**business process management (BPM)**  The services and tools that support process management (for example, process analysis, definition, execution, monitoring and administration), including support for human and application-level interaction. BPM tools can eliminate manual processes and automate the routing of requests between departments and applications.

**business rule**  A representation of how business policies or practices apply to a business activity.

**business service**  An abstract representation of a business function that hides the specifics of the function interfaces.

**Business Space**  A Graphical User Interface allowing users to customize their interface for interaction with the Business Process management suite of applications.

**CEI**  See *Common Event Infrastructure*.

**Common Event Infrastructure (CEI)**  Part of the SOA core. Can help to capture events that you can use to monitor applications, for example, in IBM WebSphere Business Monitor or IBM Tivoli products. WebSphere Process Server builds on and leverages CEI to emit a specific set of events for each Service Component Architecture (SCA) service component.

**component**  A reusable object or program that performs a specific function and works with other components and applications.

**composite applications**  Applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes.

**connector**  In Java EE, a standard extension mechanism for containers to provide connectivity to enterprise information systems (EISs). A connector consists of a resource adapter and application development tools.

**dependency**  A requirement that one managed resource has on another managed resource in order to operate correctly.

**deploy**  To place files or install software into an operational environment. In Java EE, this involves creating a deployment descriptor that is suitable to the type of application that is being deployed.

**deployment code**  Additional code that enables bean implementation code written by an application developer to work in a particular Enterprise JavaBeans (EJB) runtime environment. Can be generated by tools that the application server vendor supplies.

**deployment descriptor**  An XML file that describes how to deploy a module or application by specifying configuration and container options. For example, an EJB deployment descriptor passes information to an EJB container about how to manage and control an enterprise bean.
**directory**  A set of information with similar attributes that are organized in a logical and hierarchical manner. The most common example is the telephone directory, which consists of a series of names (either of a person or organization) that are organized alphabetically, with an address and phone number attached.

**EAR**  See enterprise archive.

**EIS**  See enterprise information system.

**enterprise archive (EAR)**  A specialized type of JAR file, defined by the Java EE standard, used to deploy Java EE applications to Java EE application servers. An EAR file contains EJB components, a deployment descriptor, and Web archive (WAR) files for individual Web applications.

**enterprise information system (EIS)**  The applications that comprise an enterprise’s existing system for handling company-wide information. Offers a well-defined set of services that are exposed as local interface, remote interfaces, or both.

**enterprise service**  A service that typically accesses one or more EISs.

**export**  An exposed interface from an SCA module that offers a business service to the outside world. Has a binding that defines how the service can be accessed by service requesters, for example, as a Web service.

**import**  The point at which an SCA module accesses an external service (a service outside the SCA module) as though it were local. Defines interactions between the SCA module and the service provider. Has a binding and one or more interfaces. A development artifact that imports a service that is external to a module.

**Integrated Solutions Console**  Provides a single, common interface for system administration. Provides the main platform on which IBM and non-IBM products can build administrative user interfaces as individual plug-ins to a common console framework. Gives administrators a more common design, use, and consistent behavior, thereby reducing the learning curve and adoption as new management components are introduced.

**interface**  A collection of operations that are used to specify a service of a class or a component.

**J2EE**  Java 2 Platform, Enterprise Edition. The former name up to version 1.5 of the currently known Java EE. See Java Platform, Enterprise Edition.

**J2EE Connector Architecture (JCA)**  A Java-based technology solution for connecting application servers and EISs.

**Java EE**  See Java Platform, Enterprise Edition.

**Java Message Service (JMS)**  Part of the J2EE suite that provides standard APIs that Java developers can use to access the common features of enterprise message systems. Supports publish/subscribe and point-to-point models and allows the creation of message types that consist of arbitrary Java objects.

**Java Naming and Directory Interface (JNDI)**  An API that is specified in Java technology that provides naming and directory functionality to applications that are written in the Java programming language. It is designed especially for the Java platform by using the Java object model. Provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. By using JNDI, applications that are based on Java technology can store and retrieve named Java objects of any type.
Java Platform, Enterprise Edition (Java EE)  An environment for developing and deploying enterprise applications, defined by Sun Microsystems Inc. Consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications, based largely on modular software components that are running on an application server. For more details about Java EE, see the following Web page: http://java.sun.com/javaee/

Java snippet  A small part of a program code that is written in Java.

JavaServer Pages (JSP)  A server-side scripting technology that enables Java code to be dynamically embedded within Web pages (HTML files) and executed when the page is served, returning dynamic content to a client.

JCA  See J2EE Connector Architecture.

JDBC  Java Database Connection

JMS  See Java Message Service.

JNDI  See Java Naming and Directory Interface.

JSP  See JavaServer Pages.


LDAP Data Interchange Format (LDIF)  A standard data interchange format for representing (LDAP) directory content and directory update (add, modify, delete, rename) requests. Represents update requests as a set of records, one record for each update request. In both cases, the data is presented in a plain text form.

LDAP directory  Often reflects various political, geographic, and organizational boundaries, depending on the model chosen. LDAP deployments today tend to use Domain Name System (DNS) names for structuring the topmost levels of the hierarchy. Deeper inside the directory, entries might appear that represent people, organizational units, printers, documents, groups of people, or anything that represents a given tree entry (or multiple entries).

LDIF  See LDAP Data Interchange Format.

Lightweight Directory Access Protocol (LDAP)  A networking protocol for querying and modifying directory services that run over TCP/IP.

MDB  See message-driven bean.

message-driven bean (MDB)  An EJB that is similar to a session bean, except that it responds to a JMS message rather than an RMI event. Were introduced in the EJB 2.0 specification, which is supported by J2EE 1.3 and later. The message bean represents the integration of JMS with EJB to create an entirely new type of bean that is can handle asynchronous JMS messages.

namespace  A logical container in which all the names are unique. The unique identifier for an artifact is composed of the namespace and the local name of the artifact.

Object Request Broker (ORB)  In object-oriented programming, software that serves as an intermediary by transparently enabling objects to exchange requests and responses. Enables service objects to locate and communicate with one another.

ORB  See Object Request Broker.

process  A progressively continuing procedure that consists of a series of controlled activities that are systematically directed toward a particular result or end. The sequence of documents or messages to be exchanged between community managers and participants to run a business transaction.
project  An organized collection that is used to group folders or packages. Used for building, version management, sharing, and organizing resources that are related to a single work effort.

queue manager  A system program that provides queuing services to applications. Enables communication between the WebSphere Message Broker components, and each component requires access to a queue manager.

Rational Software Architect  An Eclipse-based development tool for modeling application software components.

refactor  To make changes across a set of artifacts without changing the behavior of the application or its relationships to other elements.

SCA  See Service Component Architecture.

SCA component  A part of an SCA module that has a specific functionality. Can be assembled to a meaningful service.

SCA export  A possibility to provide an interface that can be called by other SCA modules by using an import.

SCA import  A possibility to call an interface of another SCA module that is provided by using an export.

SCA module  A logical unit of different SCA components that build a service.

SDO  See Service Data Object.

Service Component Architecture (SCA)  The base for an SOA because different services can be easily connected to solutions.

Service Data Objects (SDO)  A data representation that is mostly used to connect to an EIS. Cache data so that the consumer does not have to be concerned about interacting with the EIS directly.

service integration logic  Integration logic on an ESB to mediate between requesters and providers. The logic performs a number of functions such as to transform and augment requests, convert transport protocols, and route requests and replies automatically.

service-oriented architecture (SOA)  A business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks or services. Helps to build composite applications. See composite applications.

SOA  See service-oriented architecture.

SOAP  A protocol for exchanging XML-based messages over computer networks, normally by using HTTP. Forms the foundation layer of the Web services stack and provides a basic messaging framework upon which more abstract layers can build. The original acronym was spelled “Simple Object Access Protocol.” The spelling was dropped with Version 1.2 of the standard, which became a W3C recommendation on 24 June 2003 because it was considered to be misleading.

task  The basic building blocks in a model. Each task performs some function. Visually, a task represents the lowest level of work that can be portrayed in a process.

Terminal  Each node in a message flow has a number of terminals. Messages are output to different terminals on a node depending upon the results of processing in the node.

terminal  Each node in a message flow has a number of terminals. Messages are output to different terminals on a node depending upon the results of processing in the node.

Valid Workspace  The WebSphere Studio Application Developer Integration Edition Workspace which contains at least one service project is called a Valid Workspace for migration.
**WebSphere Application Server**  A powerful IBM application server that is included as a powerful foundation in other products, such as WebSphere Portal and WebSphere Process Server.

**WebSphere MQ**  A messaging application which enables the Message Brokers Toolkit, Configuration Manager, and brokers to communicate. WebSphere MQ provides many of the available transport protocols between business applications and message flows.

**WebSphere Process Server**  A state of the art Business Process execution environment.

**Widget**  Object within a Graphical User Interface with which a user interacts.

**Wire**  A connection between SCA components in a SCA module. The wire always connect Reference to interface.

**Workspace**  In Eclipse, the collection of projects and other resources that the user is currently developing in the workbench. Metadata about these resources resides in a directory on the file system. The resources might reside in the same directory.

**Workspace Migration Wizard**  Tool utilized for migrating WSADIE workspaces into WID projects.

**WSADIE**  WebSphere Studio Application Developer Integration Edition
Related publications

The publications that we list in this section are considered particularly suitable for a more detailed discussion of the topics that we covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 390. Note that some of the documents referenced here might be available in softcopy only.

- *Migrating IBM WebSphere Business Integration Server Foundation to WebSphere Process Server V6.1*, SG24-7673
- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *IBM WebSphere Business Process Management V6.1 Performance Tuning*, REDP-4431
- *WebSphere Business Integration V6 Performance Tuning*, REDP-4195

Online resources

These Web sites are also relevant as further information sources:

- IBM SOA Governance Life Cycle
- BPM and SOA: Better Together paper
- Extended Entitlement
- Product Support LifeCycle
- Supported platforms for the WebSphere Process Server run time
IBM WebSphere Developer Technical Journal: Basic steps for clustering WebSphere Process Server

WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern

WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 2: My first WebSphere Process Server cluster

WebSphere Application Server Network Deployment V6: High Availability Solutions, SG24-6688
http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SG246688

What is new for security specialists

Creating a secure WebSphere Process Server environment

WebSphere tuning for the impatient: How to get 80% of the performance improvement with 20% of the effort

Performance Tuning WebSphere Application Server

Coexisting

Profile concepts
- Messaging within WebSphere Business Integration Server Foundation

- Creating the queue manager and queues for Business Process Choreographer

- Common database configurations

- For detailed information about Business Process Choreographer
  http://www-128.ibm.com/developerworks/websphere/zones/was/wpc.html

- Instance-based roles for business processes and activities

- The basic aspects of using the staff service

- Administering WebSphere Process Server

- Administering servers and clusters

- Scripting the application serving environment (wsadmin)

- Exporting deployment environment definitions by using the console

- Importing deployment environment definitions by using the console
- Exporting deployment environment definitions by using the command line

- Importing deployment environment definitions by using the command line

- Administering deployment environments

- Querying and replaying failed messages, using the administrative console

- Using the bpeconfig.jacl script to configure Business Process Choreographer

- Deleting audit log entries, using administrative scripts

- Deleting process templates that are unused

- Deleting human task templates that are unused

- Deleting completed process instances

- Deleting data from the reporter database

- Querying and replaying failed messages, using administrative scripts
- Refreshing people query results, using administrative scripts

- Using scripts to administer Business Process Choreographer

- Enabling the event service using scripting

- Disabling the event service using scripting

- Creating an emitter factory with scripting

- Creating an event group with scripting

- Creating an event filter with scripting

- Purging events from the event database

- Administering Common Event Infrastructure

- Administering business processes and human tasks
► Overview of troubleshooting business state machines

► Administering business rules and selectors

► Enhanced audit log data analysis and query for BPEL processes with Process Choreographer 5.1

► General limitations of the migration process

► Migrating source artifacts to WebSphere Integration Developer from WebSphere Studio Application Developer Integration Edition

► Considerations for the source artifact migration process

► Preparing the WebSphere Integration developer workspace

► Business integration capabilities

► Migrating from WebSphere Studio V5.1, 5.1.1, or 5.1.2
http://publib.boulder.ibm.com/infocenter/rtlhelp/v6r0m0/index.jsp?topic=/com.ibm.etools.rad.migration.doc/topics/tmigratefrom51x.html

► Developing Web services
http://publib.boulder.ibm.com/infocenter/radhelp/v7r0m0/index.jsp?topic=/org.eclipse.jst.ws.doc.user/concepts/cws.html

► For a full list of the deprecated features in WebSphere Business Integration Server Foundation
- Migrating WebSphere Business Integration Server Foundation client code

- Monitoring business processes using common base events

- Monitoring business process events

- WebSphere 6.2 Infocenter

- For more detailed about the business space

- Troubleshooting Guide for WebSphere Process Server
  http://www-1.ibm.com/support/docview.wss?rs=2307&uid=swg27008779

- Troubleshooting Guide for WebSphere Application Server
  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27005324

- WebSphere Process Server V6.2 Information Center

- WebSphere Integration Developer V6.2 Information Center

- WebSphere Business Integration Server Foundation V5.1 Information Center
  http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp

- WebSphere Studio Application Developer Integration Edition V5.1 Information Center
  http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp
How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications, Additional materials, and order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks
Help from IBM

IBM Support and downloads
ibm.com/support

IBM Global Services
ibm.com/services
Index

A
abbreviations 375
acronyms 375
activities 21, 24, 61
advanced business process integration server 15
anonymous reference 82
APARs 370
API 194
Application Response Measurement 23
arguments 199
array 218
Artifact Importer 211
artifacts 35, 62, 120, 125, 127, 141–142, 149–150,
166, 168, 171–173, 176, 183, 189, 213, 218
modifying 129
artifacts migration 187
Assembly Diagram 171
assembly diagram 130
Assembly Diagram editor 269, 277
assembly editor 129
assessment phase 40
authorization alias 103
Authorized Program Analysis Reports 370
availability 67

B
benchmarking 53
BPC Explorer 279–280, 301
BPEL 121, 142
BPEL processes 72, 121–122, 155, 173
BPEL4WS 13, 121
Business Flow Manager 101
Business Integration Libraries 172
Business Integration Module project 150
Business Integration perspective 125–126, 184,
242, 282, 362
Business Integration project 173
Business object 20, 32, 127, 131, 134, 211, 213
business object 205, 219
business object editor 131
business object mapping editor 134
business objects 18–20, 134, 172
Business process 12, 85, 110, 120, 132, 167, 324,
353
business process 22–23, 188, 217
tuning 219
Business Process Choreographer 12, 71, 194
applications 101
database 98
features 100
security roles 102
Business Process choreographer
security roles 103
Business Process Choreographer API 196
queries 199
Business Process Choreographer bus 95
Business Process Choreographer database 12,
98–99, 112
business process container 110–111
Business Process Container API 198
business process engine container 12
business process management 30, 42
business process solutions 26
Business processes 32, 117
business processes 5, 10–12, 15, 20–23, 25, 37,
42, 48, 55, 72, 80, 96, 98–104, 111, 113, 117,
120–123, 132, 142–143, 156, 158, 161, 166, 168,
175, 183, 201, 205, 207, 221, 317, 334, 336,
353–354
business requirements 42
Business Rule Beans 10, 13, 32–33, 105, 120, 165,
189
Business Rule component 120
Business rules 32, 100, 117–118
business rules 22–23, 62, 189
business rules engine 23
Business Rules Manager 102
Business Rules Service 15
business solutions 125
Business Space 24–25
business state machine 22–23, 117, 216
BusinessFlowManager interface 201

C
cache 220
canvas 136
capacity planning 54
Center of Excellence 47
classpath 364–365
client code 204
client migration 204
ClientObjectWrapper 203–204
Clients xiv, 194, 238
cluster 110
clustering 61–62, 68–69, 220
CoE 47
Coexistence 65, 150
coeexistence 52, 55–56, 66, 74, 156–157, 160, 204
command-line interface 182
command-line script 182
Common Base Event model 19
Common Event Infrastructure 11, 19, 31, 62, 93, 96, 100, 102, 106, 111, 114, 117
   API 106–107
   architecture 107
CommonEventInfrastructure bus 96
comparison 92
   features 92
component 140
components 68, 71, 141
concurrency 220
Concurrent Versions System 143
configuration parameters 72
container 200
Container Managed Persistence 33
Container Manager Persistence over Anything 190
CORBA 14, 85
CORBA IDL interface API 14
costs 48
CVS 143, 148
disk mirroring 67
Eclipse and tutorials 125
editor 132
editor tool 129
EJB services 84
Enterprise Information Systems 15
error logs 371
error message 360, 363, 369
event emission 207
event management services 19
events 207
exception 367–368
Export 148
export 210
extended data element 207
Extended Messaging 93
extended messaging service 190
failover 67–68
feature comparison 31
features comparison 32
fix packs 52, 56
Flow Definition Markup Language 122
funding 48
gap analysis 62–63
getter methods 206
Glossary 377
goal-service modeling 59
governance 47
HACMP 68
High-Availability Cluster Multi-Processing 68
human task 216
human task activities 142, 198, 225
Human Task Manager 110–111
human tasks 21
HumanTaskManager interface 201
IBM Enterprise Service Bus 17
IBM SOA Foundation 37
IBM Support 371
implementation phase 63
import 210
inline task 216
inline tasks 217
install 86, 184, 333, 348–349, 363
integration debugger 140–141
integration test client 139
interface 133
interface editor 133, 283, 365
interface mapping editor 133
Interface maps 20, 32

J
J2CA0138E exception 368
J2EE applications 10, 12, 14, 71
Java build path 364
Java heap size 220
Java services 26, 84, 224, 228, 232
java.lang.NullPointerException exception 366
Java2WSDL 83
JMS API 201
JMS provider 97
JRE System Library 364
Jython 116
Jython language 115

K
key performance indicators 143

L
Launch Business Process Web Client 238, 311
library 127–128
log file 361
log files 360
logs 365
long running process instances 160
long running processes 55, 66, 73–74, 150, 153–154, 219–220

M
macroflows 217
manual migration steps 362
mapping 88, 171
maps 219
Mediation flows 19–20
messaging 94, 220
method overloading 188
methodology 50, 61
methods 198–199
microflows 217, 219
migrate 123, 144–145, 148, 173, 183
   multiple workspaces 183
   workspace 187
migrate clients 201
migrate FDML processes 122
migrated components 60
migrating 37
migration 30, 43, 61, 72, 74, 124, 143
   client source 85
   considerations 185
   modeling 85
migration approaches 204
migration comparison 35
migration messages 362
Migration process 31
migration process 30–31, 219, 227
   API calls 204
   BPEL limitations 188
   CEI applications 207
   client applications 194
   client code 194
   CMP/A application 191
   EJB project limitations 188
   limitations 186
   performance tuning 219
   SCA programming model limitations 188
   troubleshooting 361
Migration Results window 361–362
migration tasks 165
migration tool 143
migration wizard 5, 35–36, 84, 120–122, 124,
   141–142, 150, 163, 166, 175–177, 185, 205, 259,
   361
module 20, 24–25, 84, 95, 126–129, 141–142, 167,
   171–173, 175, 183–184, 188, 195, 209, 211, 213,
   218, 220, 229, 232, 255, 260–261, 263, 268–273,
monitoring 219
Move-at-once approach 160, 204
MustGathers 371

N
namespace 82–83, 367
namespace collision 186
namespaces 219
naming conventions
  name mapping 87–88

O
Object Request Broker 220
operation 133
ORB 221
ORB thread pool 220

P
packaging 54
People Directory Provider 103
People Interaction component 13
performance monitoring 219
Performance Monitoring Infrastructure 23
performance tests 53, 72
performance tuning 219, 221
pilot 74, 80
pilot project 61
planning 60
post migration 215
preparation phase 61
Process Choreographer database 12, 98–99, 112, 154–155
process durations 156
process editor 132, 266, 275
process instance 13
process instance identifier object 198
process instances 161
Process Navigation 13
process templates 199
product support 370
programming model extensions 14, 33
project types 186
projects
  naming conventions 86
  modeler 86

Q
queries 200

R
Rational Software Development Platform 51
ratios 220
Redbooks Web site 390
  Contact us xv
refactoring 216
  best practices 216
reference 82, 169
regression testing 53
relationship 135
relationship editor 135
Relationships 20, 32, 127
relationships 59, 218
requirements 46, 59, 72
risk assessment 59
roadmap 49
ROI 59
roles 44
run time 55–56

S
SCA 22, 51
SCA components 20–21
SCA exports 195
SCA programming model 195, 205
SCA proposes 169
schemas 81
script 113, 116
scripts 108, 111, 115, 149
SDO 18, 203–205
SDO API 203
Secure Sockets Layer 71
  roles 103
Selector 86, 100
Selectors 20, 32, 118
Service Component Architecture 18, 51, 168
service components 21–22, 117
service components layer 16
Service Data Objects (SDOs) 15
service decomposition 56–58
service integration bus 94
service integration buses 96
Service level agreement 50
service level agreement 80
service provider interfaces 190
service proxy 185
service-oriented architecture core 17
service-oriented modeling and architecture 51
shared library 129
SLA 80
SOA xii–xiii, 10, 13, 15–17, 22, 25, 31, 37, 42, 45, 47–48, 50–51, 58, 60, 121, 124–125, 168, 353–354
SOMA 51
source artifact migration 361
source artifacts 187
SSL 71
staff plug-in 103–104
staff service 105
Staff Support Service 103–104
staff verbs 104–105
stand-alone human task 216
stand-alone task 217
support 370
supporting services components 20
supporting services layer 16, 19

T
task assignments 23
templates 111–112, 117
test suite editor 137
test suites 137
testing 53, 139
threads 220
tool	editor 129
topology 54, 62, 66, 69, 80
traces 371
tracing 219
Troubleshooting 360
troubleshooting 74, 359, 361, 369
t.technical scenarios 366
tuning 73–74

U
unit testing 53

V
verb sets 105
visual snippet editor 136–137

W
Web material 373
Web service 10, 14–15, 19, 35, 83, 173, 175, 204
Web Service Invocation Framwork (WISF) 185
Web services xi, 10–12, 15, 21–22, 32, 37, 71, 75, 83, 95, 121, 124, 168, 173, 175, 194, 197, 202
Web services API 202
Web Services Invocation Framework 84, 121
WebSphere Application Server 94
MustGathers 371
WebSphere Business Integration Adapter Artifact Importer 210
WebSphere Business Integration Adapters 208–209
WebSphere Business Integration Modeler xi, 6, 85, 121, 123–124, 143–145, 148, 225, 227, 298, 303, 327, 366
MustGathers 371
WebSphere Enterprise Service Bus 19
MustGathers 371
Workbench 360
WebSphere Modeler 15, 22, 333
WebSphere Monitor 15, 22
WebSphere MQ 69, 97
  clusters 109
deprecated features 189
MustGathers 371
WebSphere Process Server artifacts
  best practices 217
WebSphere Process Server layers 16
WebSphere Studio Application Developer Integration Edition 176
  APIs 189
WISF 185
workspace 144, 174–175, 363
workspace migration 177
workspace migration wizard 175
WSDL 84
WSDL files 83
WSDL message 203
WSIFMessage API 206
WSIFMessage interface 203

X
XSD 81
Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server V6.2 and Best Practices
Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server V6.2 and Best Practices

In this IBM Redbooks publication, we discuss the concepts, planning, and migration paths that you must understand before attempting to migrate source artifacts, which are created by using IBM WebSphere Studio Application Developer Integration Edition 5.1, to IBM WebSphere Integration Developer 6.2. We also discuss how to migrate models that are developed in WebSphere Business Integration Modeler 5.1 to WebSphere Business Modeler 6.2.

In this book, we discuss the new features and differences between WebSphere Business Integration Server Foundation and WebSphere Process Server 6.2. We cover discussions around the considerations for target topologies to be migrated to, and we also cover the deprecated features of WebSphere Process Server 6.1. We also give you information that is critical to any upgrade path and help you realize the benefits of the IBM new breed of integration products.

International Technical Support Organization

Building Technical Information Based on Practical Experience

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information: ibm.com/redbooks