Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server and Best Practices

Migration concepts and planning

Migration tools and best practices

Migration technical scenarios

Saida Davies
Aditya P Dutta, Khaled Gamal
Susan Herrmann
David Kadlecek
Augusto Kiramoto
Jayasaikumar Padimiti
Vishnu Selvaraj
Jaime Valencia
Iftekhar Siddiqui
Ratan Siripurapu, Bo Wang

Redbooks

ibm.com/redbooks

International Technical Support Organization

Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server & Best Practices

April 2008

SG24-7416-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xv.

First Edition (April 2008)

This edition applies to:

Version	Release	Modification	Fix pack	product-name
6	0	2		WebSphere Process Server
6	0	2		WebSphere Integration Developer
5	1	1		WebSphere Business Integration Server Foundation
5	1	1		WebSphere Studio Application Developer Integrated Edition
XP			Service Pack 2	Microsoft Windows

Contents

Figures ix

Tables xiii

Notices xv
Trademarks xvi

Preface xvii
The team that wrote this IBM Redbooks publication xvii
Become a published author xxiv
Comments welcome xxiv

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 The organization of this book 4
1.3 Scenarios demonstrated 5
1.4 What is not covered in the book 6
1.4.1 Install and use of software products 6
1.5 Assumptions 6

Chapter 2. Product overview 9
2.1 Server Foundation overview 10
2.1.1 Introduction to Server Foundation 10
2.1.2 Architectural overview of Server Foundation 10
2.1.3 Features of Server Foundation 13
2.2 WebSphere Process Server overview 15
2.2.1 Introduction to WebSphere Process Server 15
2.2.2 Architectural overview of WebSphere Process Server 16
2.2.3 Features of WebSphere Process Server 23

Chapter 3. Migration concepts 27
3.1 Migration concepts overview 28
3.2 Features mapping and migration approach 29
3.2.1 General feature comparison 29
3.2.2 Source artifacts, bindings, and the migration approach 32
3.3 Benefits of migrating to WebSphere Process Server 35

Chapter 4. Migration planning 37

4.1 Migration project considerations 38
4.1.1 Migration project phases 38
4.1.2 Migration project roles 40
4.1.3 Assessment. 41
4.1.4 Environment planning 42
4.1.5 Migration 42
4.1.6 Testing 43
4.1.7 Deployment 43
4.2 Non functional aspects 44
4.2.1 Availability considerations 44
4.2.2 Security considerations 47
4.2.3 Performance considerations 49
4.3 Product coexistence 51

Part 2. Migration procedure 53

Chapter 5. Migration options 55
5.1 Migration paths 56
5.1.1 Migrating business processes described in BPEL 56
5.1.2 Migrating business processes described in FDML 57
5.1.3 Migrating business processes mainly defined in Modeler 58
5.2 Migration tools 59
5.2.1 WebSphere Integration Developer overview. 59
5.2.2 WebSphere Business Modeler 72
5.3 Migration approaches 79
5.3.1 Migrating with dependencies between applications 79
5.3.2 Considerations for long running processes. 82

Chapter 6. Migrating build time artifacts. 91
6.1 Introduction to migration of build time artifacts 92
6.2 General considerations 93
6.2.1 General migration tasks for source artifacts 93
6.3 Preparing source artifacts for migration 94
6.3.1 Project organization changes in Integration Developer. 94
6.3.2 Preparing the Integration Edition workspace 97
6.3.3 Preparing the Integration Developer workspace 99
6.4 Migrating Service projects using the Migration wizard 101
6.4.1 Overview of the Migration wizard 101
6.4.2 Using the Migration wizard 101
6.5 Manually completing the migration 105
6.6 Migrating BPEL processes 106
6.6.1 Migrating BPEL Java snippets 106
6.6.2 Migrating staff activities. 110
6.6.3 Migrating process invocation and binding 110

6.7 Migrating Java services 114
6.7.1 Creating a custom Java component 114
6.7.2 Creating a Java Web service 117
6.8 Migrating EJB services 117
6.8.1 Creating an SCA import with Stateless Session Bean Binding 118
6.8.2 Creating an EJB Web service 122
6.9 Migrating Web services 123
6.9.1 Migrating a Web service (SOAP/JMS) or (SOAP/HTTP) 123
6.9.2 Migrating Web service Security 124
6.10 Migrating clients 125
6.10.1 Migrating to clients using SCA programming model 126
6.10.2 Migrating to clients using Process Choreographer API 127
6.10.3 Client migration approaches 134
6.11 Migrating WSIFMessage API calls 135
6.12 Migrating Business Rule Beans applications 137
6.13 Migrating Container Managed Persistence over Anything applications 138
6.14 Migrating Extended Messaging applications 138
6.15 Migrating CEI applications 139
6.15.1 Business Process Choreographer events 139
6.16 Migrating WebSphere Business Integration Adapters 140
6.16.1 Development tools for WebSphere Business Integration Adapters143
6.16.2 Migration steps 143
6.16.3 Artifacts generated 146

Chapter 7. Migrating runtime components 147
7.1 Introduction to migration of runtime components 148
7.2 General considerations 148
7.2.1 Product upgrade and configuration migration 148
7.3 Process Server topology 149
7.3.1 Messaging in Process Server 149
7.3.2 Databases required by Process Server 151
7.3.3 Selecting a target topology 152
7.4 Migrating Business Process Choreographer 153
7.4.1 Business Process Choreographer applications 154
7.4.2 Business Process Choreographer messaging resources 156
7.4.3 Business Process Choreographer database resources 157
7.4.4 Business Process Choreographer administration 158
7.4.5 Staff support service and staff-plugin configuration 163
7.4.6 Microflows and long running processes 164
7.5 Migrating Business Rule Beans 165
7.6 Migrating Extended Messaging 166
7.7 Migrating the Common Event Infrastructure 166

Chapter 8. Best practices . . . 169
8.1 Planning . . . 170
8.2 Artifacts preparation . . . 170
8.2.1 WSADIE artifact preparation . . . 171
8.2.2 Modeler artifact preparation . . . 171
8.2.3 Schema XSD files . . . 171
8.2.4 WSDL files and Web services . . . 173
8.2.5 BPEL artifacts . . . 174
8.2.6 Client source . . . 174
8.2.7 Modeling migration best practices . . . 175
8.2.8 Naming convention . . . 175
8.2.9 Modeler naming convention . . . 176
8.3 Post migration . . . 179
8.3.1 Refactoring processes . . . 179
8.3.2 Further development . . . 181
8.4 Non-functional considerations . . . 182

Chapter 9. Technical scenarios . . . 185
9.1 Overview of scenarios . . . 186
9.2 Hardware and software environment setup . . . 186
9.3 Travel Operations scenario . . . 188
9.3.1 Overview . . . 189
9.3.2 Preparation . . . 189
9.3.3 Implementation . . . 204
9.3.4 Verification . . . 243
9.4 Travel Booking scenario . . . 246
9.4.1 Overview . . . 247
9.4.2 Preparation . . . 247
9.4.3 Implementation . . . 259
9.4.4 Verification . . . 268

Chapter 10. Troubleshooting . . . 273
10.1 General . . . 274
10.1.1 Import into WebSphere Integration Developer . . . 274
10.1.2 Build the project in WebSphere Integration Developer . . . 276
10.1.3 Deploy and run the process in WebSphere Process Server . . . 277
10.2 Scenarios . . . 277
10.3 Advanced . . . 282
10.3.1 Search in information center or manuals . . . 282
10.3.2 Search in product support Web site . . . 282
10.3.3 Contact IBM support for help . . . 283

Appendix A. Additional material . . . 285
Locating the Web material . . . 285

Using the Web material 286
    How to use the Web material 286

Glossary 287

Abbreviations and acronyms 291

Related publications 293
IBM Redbooks 293
Online resources 293
How to get IBM Redbooks 296
Help from IBM 296

Index 297

Figures

2-1 The WebSphere Business Integration Server Foundation architecture . . 11
2-2 Business process engine container architecture. 12
2-3 Elements of Server Foundation. 14
2-4 Components of WebSphere Process Server 17
2-5 Service-oriented architecture core layer of WebSphere Process Server . 18
2-6 Service Component Architecture 19
2-7 Supporting services layer of WebSphere Process Server 20
2-8 Service component layer of WebSphere Process Server 22
3-1 Migration Process 29
4-1 Activities in Server Foundation to Process Server migration project. Time percentage is indicative and can be different for your project. 39
4-2 Relationship between level of availability and costs 45
4-3 Components to be clustered in WebSphere Process Server 46
5-1 Migration path for BPEL processes. 55
5-2 Migration path for FDML processes 56
5-3 Migration paths from a Modeler V5 project 57
5-4 Welcome page 59
5-5 Workbench 60
5-6 Assembly editor. 61
5-7 Business object editor 61
5-8 Process editor 62
5-9 Interface editor 62
5-10 Interface mapping editor 63
5-11 Business object mapping editor 64
5-12 Relationship editor 64
5-13 Visual snippet editor 65
5-14 Dependency editor 66
5-15 Enterprise service discovery wizard 67
5-16 Integration test client. 68
5-17 Artifact Migration wizard 69
5-18 Backup location definition in Migrate Workspace window. 71
5-19 Migration completed 72
5-20 Modeler Import: Format selection 73
5-21 Modeler import: Project selection 74
5-22 Importing project from Modeler V6 75
5-23 Applications invoke one another, with no circular dependencies 77
5-24 Migration starting first last 77
5-25 Migration starting last first 78

5-26 Process durations 81
5-27 Coexistence approach 82
5-28 Scenario Migrate the longest last 83
6-1 Service Component Architecture Layers. 91
6-2 Module project. 92
6-3 Library project 93
6-4 Export Project Interchange in Integration Edition V5.1 95
6-5 Invoke the Migration wizard within WebSphere Integration Developer V6.0.2 98
6-6 Specify Service project to be migrated 99
6-7 Select Preserve the original BPEL Java snippets in the comments 99
6-8 Migration Results window 100
6-9 Client migration options using SCA programming model 123
6-10 Clients using the Business Process Choreographer API 124
6-11 Business Process Choreographer API in Process Server 129
6-12 Business Objects for Service Component Architecture 132
6-13 WBI Adapters architecture in WebSphere Process Server. 137
6-14 WebSphere Business Integration Adapter Artifact Importer 140
6-15 Specifying initialization properties. 140
6-16 Find and Discover Enterprise Services. 141
7-1 Buses used by Process Server. 147
7-2 Componentized Business Process Choreographer within Process Server . 150
7-3 Admin console: Process Choreographer applications within Process Server 151
7-4 Replay messages runtime function in Server Foundation console 155
7-5 Configuration entry point for Business process and Human task container . 156
7-6 Process Server runtime configuration panel for human task container. . 158
7-7 Staff resolution plug-ins in Server Foundation and Process Server 159
7-8 Selecting Business Rules Beans during Process Server Installation . . . 161
7-9 Selecting Extended Messaging during Process Server Installation 162
7-10 Capabilities of Common Event Infrastructure 163
7-11 Architecture of the Common Event Infrastructure 163
9-1 Physical topology 183
9-2 Migration process to WebSphere Process Server 185
9-3 Services and Navigator views on the projects to migrate 187
9-4 Java Build Path of CommonArtifacts project. 188
9-5 .wsdl files to move in the CommonArtifact Java project 190
9-6 Selection dialog for moving the .wsdl files to the CommonArtifact project191
9-7 Regenerated TravelAgencyProcess_travelPort_EJB.wsdl file 193
9-8 Add and Remove Projects dialog 194
9-9 Process Web Client: Start a TravelOperationsProcess 195

9-10 Process Web Client: Complete the staff activity 196
9-11 WebSphere Integration Developer Welcome page. 199
9-12 Setting Integration Developer Capabilities for migration 200
9-13 EmailSender projects import into Integration Developer workspace. . . 201
9-14 J2EE Migration wizard: Warning dialog 202
9-15 J2EE Migration wizard: Migration options 203
9-16 J2EE Migration wizard: Migration options for EJB projects. 204
9-17 J2EE Migration wizard: Migration completed successfully 205
9-18 Java Build Path of EmailSender: clean up the classpath 206
9-19 The J2EE projects in Interface Editor after classpath clean up. 207
9-20 Creating a Business Library in Integration Developer. 208
9-21 CommonArtifacts project structure after migration 209
9-22 Import the WebSphere Studio Application Developer Integration Edition TravelAgency project.. 210
9-23 Enter the location of the TravelAgency project to be migrated 211
9-24 Select Preserve the original BPEL Java snippets in the comments . . . 212
9-25 Migration Results for TravelAgency service project. 212
9-26 TravelAgency module structure in Integration Developer 213
9-27 Build errors in the HotelReservationToolImpl.java class. 214
9-28 Assembly Diagram of TravelAgency module 217
9-29 TravelAgencyProcess in Process Editor 218
9-30 JNDI name in the EmailSenderBean import. 221
9-31 Dependencies of TravelAgency module. 223
9-32 Add and Remove Projects dialog 224
9-33 BPC Explorer: Start a TravelAgencyProcess 225
9-34 Migration Results for TravelOperations service project 226
9-35 TravelOperations module structure in Integration Developer 227
9-36 Assembly Diagram of TravelOperations module. 230
9-37 TravelOperationsProcess in Process Editor 230
9-38 Migrated Prepare Approval Request Assign activity. 231
9-39 Migrated Prepare Approval Request Assign activity with From Value . 232
9-40 TravelPort import directing to the TravelAgency module 233
9-41 Assembly Diagram of TravelOperations module after deleting the exports 234
9-42 Dependencies of TravelOperations module 235
9-43 BPC Explorer: Start a TravelOperationsProcess 237
9-44 BPC Explorer: Complete the task 238
9-45 Starting instance of TravelOperationsProcess in BPC Explorer 239
9-46 Work on Approve Trip task in BPC Explorer 240
9-47 Email received from the TravelAgencyProcess running in Process Server 241
9-48 Importing model in WebSphere Business Integration Modeler V5 243
9-49 model to migrate Project Tree. 244

9-50 Customer business item 244
9-51 Booking Order business item 245
9-52 Activities in Booking Order 245
9-53 Booking Order process 246
9-54 Importing process in WSADIE. 247
9-55 Implementing the Reservation System activity. Build Service. 248
9-56 Implementing the Reservation System activity. Service port type definition. 249
9-57 Generating the Server Foundation deploy code. 251
9-58 Deploy code generation. Partner selection. 252
9-59 Tables created in server 253
9-60 Process Navigation in Server Foundation. 253
9-61 Importing the project version 5.1.1 in modeler v6.0.2 254
9-62 Migration message 255
9-63 Combine catalogs 255
9-64 Import finished. 255
9-65 Model in WebSphere Business Modeler V6 256
9-66 Exported as Project Interchange V6 257
9-67 Project Interchange Import in WebSphere Integration Developer. 258
9-68 Import projects in WebSphere Integration Developer module and Library. 259
9-69 Generating the Java Implementation 260
9-70 Generating the Java Implementation. Select the package 261
9-71 Adding the project to WebSphere Process Server 263
9-72 Starting instance of the process 264
9-73 Process Name. 264
9-74 Initial activity in process 265
9-75 Initial form.. 265
9-76 Final human task activity. 266
10-1 Migration Results window. 269
10-2 WebSphere Integration Developer “Problems” View. 270
10-3 Error message in Integration Developer 272
10-4 Migrated Prepare Approval Request Assign activity: fixed value within XPath 274
10-5 Migrated Prepare Approval Request Assign activity with fixed value . . 275

Tables

3-1 Features Comparison 30
3-2 Programming Model Extensions and the product containing them in version 6 31
3-3 Binding mapping 33
3-4 Artifacts Mapping 34
4-1 Project roles in detail 40
4-2 Project phases mapped to project roles. 40
6-1 Changes and solutions for migrating WebSphere Business Integration Server Foundation BPEL Java snippets. 103
6-2 Generated namespace 109
6-3 Changes and Solutions for migrating WSIFMessage API calls to SDO APIs 133
6-4 Artifacts generated 142
7-1 Database within Process Server 148
7-2 Security roles availability in BPC in both products 152
8-1 Anonymous reference example 168
8-2 Example name mapping from Modeler to WSADIE 173
8-3 Example name mapping to WebSphere Integration Developer 174
9-1 Hardware environment 183
9-2 Software versions used in the scenarios. 184
9-3 Project dependencies 188
9-4 Components, Exports and Imports generated by the Migration wizard . 216
9-5 Components, Exports and Imports generated by the Migration wizard . 228

Notices

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®
Cloudscape®
DB2 Universal Database™
DB2®
e-business on demand®
IBM®
Parallel Sysplex®
Rational®
Redbooks®
Redbooks (logo) ®
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

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

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

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

Intel, Pentium, Pentium 4, 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.

Preface

In this IBM® Redbooks publication, we discuss the concepts, differences, and migration paths that you must understand before you attempt to migrate the artifacts that you created using the IBM WebSphere® Studio Application Developer Integration Edition 5.1 product to the IBM WebSphere Integration Developer 6.0.2. We also include a discussion on how to migrate models that are developed in WebSphere Business Integration Modeler 5.1 to WebSphere Business Modeler 6.0.2.

In this book, we provide guidance on how to migrate the processes that are installed and running in WebSphere Business Integration Server Foundation to the new integration platform. We also tell you how to bring your components and artifacts to this new generation of integration paradigm.

Part 1, “Products overview and migration planning” on page 1 of this book provides a product overview and helps you to assess your environment and plan WebSphere Business Integration Server Foundation migration to WebSphere Process Server 6.0.2

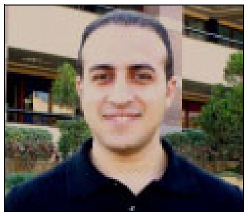
Part 2, “Migration procedure” on page 53 provides detailed steps that are required for migration and includes the tools that are available, the artifacts that are involved, and best practices. In this section, we also demonstrate the migration process using practical migration scenarios that cover most of WebSphere Business Integration Server Foundation capabilities, which includes process choreography, human activities and staff assignment, and exception and error handling.

The team that wrote this IBM Redbooks publication

A team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center produced this book.

Saida Davies is a Project Leader for the International Technical Support Organization (ITSO) and has extensive experience in Information Technology. She has published several IBM Redbooks publications® and Redpapers 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. As a senior IT specialist, her responsibilities included the development of services for WebSphere MQ within the z/OS and Windows® platform, 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 z/OS Systems Programming. Saida supports Women in Technology activities and contributes and participates in their meetings.

Aditya P Dutta is a Senior Software Engineer at IBM India. He has around seven years of experience in application development, systems integration, and consulting assignments. His areas of expertise include J2EE™ and Enterprise Application Integration. He specializes in architecting and designing Enterprise Integration solutions using the IBM WebSphere Business Integration suite of products. He has a Bachelors degree in Electronics and Instrumentation Engineering from College of Engg. and Technology, Bhubaneshwar, India.



Khaled Gamal is a Solution Lead in Technical Competence Group at IBM Egypt. He has seven years of design and development experience, which includes four years of experience with IBM business integration products. He worked in the WebSphere Business Integration Modeler product lab for two releases and since 2006, he is a member of the worldwide team called service-oriented architecture (SOA) Advanced technology. Khaled is an SOA solution designer and a lead at the Egypt SOA design center. He is a Sun-certified Java™ 5 Programmer, IBM Certified Business Process Analyst, and an IBM SOA Solution Designer. Khaled is also designated as an IBM inventor for two patents. He completed his Masters degree in Systems and Biomedical Engineering from Cairo University in Egypt.

Susan Herrmann is a Software Engineer at the IBM Development Lab in Böblingen, Germany. She has six years of experience in the areas of Business Integration and Business Process Management. After joining IBM, she worked on the design and implementation of WebSphere Business Integration for Financial Networks, which is an enterprise banking solution. Her current main area of expertise is WebSphere Process Server, which is the IBM SOA solution. Susan has extensive knowledge in solution architecture and many SOA technologies, which include BPEL, several J2EE technologies, messaging, and message broker solutions. She has several IBM Certified Solution Developer and Specialist certifications. As a member of the WebSphere Process Server SWAT team, she provided vital support in numerous critical customer situations worldwide and took part in complex Proof of Concepts (PoCs). Susan has an Engineering degree in Information Technology (Berufsakademie Stuttgart). She joined IBM Germany in 2001.

The material in this book required additional effort and Susan was instrumental in the completion through her additional contribution and further reviews.

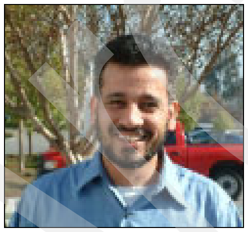
David Kadlecek is an IT architect in Global Technology Services in Prague. He has six years of design and development experience with J2EE and four years of experience with WebSphere Application Server and workflow products. He has broad experience in application and enterprise architecture in a number of industries that include banking, logistics, and telecommunication. His areas of expertise also include Web services, security and performance tuning. He is currently completing PhD in Artificial Intelligence at Czech Technical University in Prague.

Augusto Kiramoto is a Software Engineer and Project Leader at IntVision Solutions LTDA in Sao Paulo, Brazil. He has worked on the design and implementation of Enterprise Application Integration and Business Process Management solutions with WebSphere Business Integration products for five years. He has a bachelor's degree in Computer Engineering and is finishing his Masters in Software Engineering at Escola Politecnica at São Paulo University.

Jayasaikumar Padimiti is a director for Miracle Software Systems (Novi, MI) IBM Business Partner Innovation Center, which is a technological facility that focuses on IBM WebSphere Software and service-oriented architecture. Jayasaikumar has over five years of IT experience of which he spent over three years in service-oriented architecture and Enterprise Service Bus architecture-based implementations, Web services design and development, advanced Java and J2EE technologies, and IBM WebSphere Tools and technologies. His areas of expertise include Business Process Management, Java, J2EE, Application Integration, Business Integration and WebSphere. Jayasaikumar is an IBM certified SOA Solution Designer and an IBM Certified Deployment Professional for WebSphere Process Server V6.0. Jayasaikumar has a Bachelor of Engineering in Computer Science and Engineering from M.V.S.R Engineering College, Hyderabad, India and a Masters of Science degree in Computer Science from The University of Kansas.

Vishnu Selvaraj is a member of the Software Engineering team that works in IBM Burlingame. He has five years of experience in Information Technology, which includes three years working on the design and implementation of WebSphere InterChange Server-based application integration solutions. He is part of the Adapter Development team and is now working in the SWAT team that supports WebSphere Process Server. His current responsibilities include solving critical customer problems and providing feedback to development teams on product enhancements. He completed his Masters degree in Engineering from the University of Cincinnati in Ohio, USA.

Jaime Valencia is a Consulting IT Senior Architect at VC@SOFT Ltd, which is a software developing company in Bogotá, Colombia. He has nine years of Java architecting, designing, and developing experience. Over the past seven years, Jaime focused on WebSphere, J2EE, BPM, and Integration technologies. His expertise also includes IBM and SAP® Integration and BPM platforms and technologies. He is a Sun-certified Java Programmer, WebSphere-certified specialist, and SAP/Netweaver-certified consultant. Jaime has an Engineer's degree in Electronics from the Pontificia Universidad Javeriana.



Iftekhar Siddiqui is a senior WebSphere IT Specialist with the IBM Technical Channel Sales Team. He joined IBM in 1999. His current job is working with IBM Business Partners to drive IBM service-oriented architecture and to ensure client success by developing their technical skills plan. He delivers education in many WebSphere platform software, which includes WebSphere Application Server, Business Process Integration products. He has 13 years of experience in Information Technology and has worked as a Developer using Object Oriented design, C/C++, and Java / J2EE platform. He has a Bachelor of Science in Computer Science from California State University.

Ratan Siripurapu is a Senior Technical Architect at Sarasu IT Solutions, LTD, based in Bentonville, AR, USA. The company provides consulting services, specializing in IBM WebSphere, service-oriented architecture, Application Integration, and Process Management. He has ten years of experience in Information Technology with a focus on Business Process Management, Enterprise Application Integration, and messaging. He is highly experienced with IBM WebSphere and the IBM WebSphere Business Integration suite of Products and SOA. He has a Bachelor's degree in Computer Science and Engineering from Amaravati University, India and a Master's degree in Computer Engineering from the University of South Carolina, SC, USA.

Bo Wang is a software support professional for the WebSphere product family in IBM China. He has worked on WebSphere Business Integration products since he joined IBM in 2004. His main experience is system architecture design and system maintenance on the IBM WebSphere product family. Prior to this, he worked for an IBM business partner as a Software Engineer for three and a half years. He worked on application design and coding based on some IBM products during this period. He has a Bachelors degree in Computer Science from Peking University.

Sincere thanks to:

Michele Chilanti
Senior Consultant, WebSphere Services Technical Sales, IBM Software Group, Application and Integration Middleware Software, IBM Rochester, USA.

Michele's support through the project was invaluable. He provided material and documentation on migration from WebSphere Business Integration Server Foundation to WebSphere Process Server based on his experiences, which the team all found very useful.

Ed Spring
WebSphere Process Server Migration Lead/Architect, IBM Master Inventor, IBM Software Group, Application and Integration Middleware Software, IBM Austin, USA for his team support especially in resolving problems that we encountered during the project.

Juan Obas
WebSphere Business Integration Development, IBM Software Group, Application and Integration Middleware Software, IBM Austin, USA for his contribution in the review process and providing appropriate feedback.

Kurt Fleckenstein
WSS BPM Competence Center, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany for his contribution in the review process and providing appropriate feedback.

The IBM Redbooks publication team wants to thank the following people for their guidance, assistance, and contributions to this project:

Friedemann Schwenkreis
Architecture of Workflow Application, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany.

Cheryl Sedota
WebSphere Process Server, SCA Development, Software Engineer, IBM Austin, USA.

Jeff Brent
WebSphere Process Server SWAT Technical Lead, Software Engineer, IBM Boca Raton, USA.

Jose A Martinez Rodriguez
WebSphere Process Server SWAT, Software Engineer, IBM Burlingame, USA.

Gunnar Wilmsmann
Software Developer, Business Flow Manager, IBM Boeblingen, Germany.

Kurt Lind
Human Task Manager, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany.

Uta Dresch
WebSphere Workflow Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany.

Uma Rao
WebSphere Business Integration, IBM Software Group, Application and Integration Middleware Software, IBM Toronto, Canada.

Erich Fussi
WebSphere Solutions and Services - BPC Architecture, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany.

Bernd Breier
IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany.

Become a published author

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

Comments welcome

Your comments are important to us!

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

- Use the online **Contact us** review form found at:

  ibm.com/redbooks

- Send your comments in an email 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

Part 1

Products overview and migration planning

In part 1, we provide an introduction to this book and discuss the book’s scope. We provide an overview of 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.

Chapter 1, “Introduction to the book” on page 3 discusses 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 and WebSphere Process Server. In this chapter, we also focus on the new features that are available in WebSphere Process Server.

Chapter 3, “Migration concepts” on page 27 compares the features and functions of the two products and introduces the new features of WebSphere Process Server.

Chapter 4, “Migration planning” on page 37 covers the recommended approach on how to plan a migration to WebSphere Process Server.

1

Introduction to the book

In this chapter, we discuss the following:

- 1.1, “The scope of this book” on page 4
- 1.2, “The organization of this book” on page 4
- 1.3, “Scenarios demonstrated” on page 5
- 1.4, “What is not covered in the book” on page 6
- 1.5, “Assumptions” on page 6

1.1 The scope of this book

The aim of this book is to provide a migration guide for migrating WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.0.2, plan the migration, and identify specific migration issues.

Using WebSphere Process Server migration tools with source artifacts from WebSphere Business Integration Server Foundation V5.1, the scenarios 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 the comparison of WebSphere Business Integration Server Foundation artifacts with WebSphere Process Server artifacts is documented.

1.2 The organization of this book

We divided this book is into two parts.

Part 1, “Products overview and migration planning” on page 1 provides product overview, helps you assess your environment, and helps you plan WebSphere Business Integration Server Foundation V5.1 migration to WebSphere Process Server V6.0.2.

We discuss the following chapters in Part 1, “Products overview and migration planning” on page 1:

- 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 V6.0.2. In this chapter, we also focus on the new capabilities of WebSphere Process Server V6.0.2.
- Chapter 3, “Migration concepts” on page 27 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 V6.0.2.
- Chapter 4, “Migration planning” on page 37 covers the recommended approach on how to plan a migration to WebSphere Process Server V6.0.2.

Part 2, “Migration procedure” on page 53 provides the detailed steps that are required for migration, the tools that are available, the artifacts that are involved, and best practices. In this section, 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.

We discuss the following chapters in Part 2, “Migration procedure” on page 53:

- Chapter 5, “Migration options” on page 55 introduces tools that are used during migration, for example, WebSphere Integration Developer V6.0.2. This chapter focuses on the tools capabilities with regards to migration.
- Chapter 6, “Migrating build time artifacts” on page 91 describes the common and supported paths for migrating WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.0.2.
- Chapter 7, “Migrating runtime components” on page 147 explains how to migrate artifacts that were developed using the WebSphere Application Developer Integration Edition V5.1.
- Chapter 8, “Best practices” on page 169 considers approaches for migrating the runtime components from WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.0.2.
- Chapter 9, “Technical scenarios” on page 185 discusses the best practices to use when you migrate from WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server V6.0.2.
- Chapter 10, “Troubleshooting” on page 273 includes step-by-step instructions on migration scenarios and illustrates how to use the migration tools.

1.3 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.

We used the following products in the migration scenarios that we performed:

- WebSphere Business Integration Server Foundation V5.1.1
- WebSphere Integration Developer V6.0.2 with Integrated WebSphere Process Server development environment.
- WebSphere Process Server V6.0.2

- WebSphere Business Integration Modeler V5.1.1
- WebSphere Business Modeler V6.0.2

1.4 What is not covered in the book

In this section, we provide information about the topics that are not covered in this book.

This book focuses on tasks that relate to the migration to WebSphere Process Server V6.0.2 from WebSphere Business Integration Server Foundation V5.1

This book does not cover:

- Details on the installation of software products, which includes WebSphere Business Integration Server Foundation V5.1, WebSphere Integration Developer, and WebSphere Process Server; therefore, no step-by-step instructions are included.
- Installation of pre-requisite and co-requisite software, and operating system updates.
- Instructions about the installation, configuration, or use of any previous version of WebSphere Process Server.

1.4.1 Install and use of software products

Refer to the installation and system requirements documentation for further details about WebSphere Integration Developer V6.0.2 and WebSphere Process Server V6.0.2 products at:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/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

1.5 Assumptions

In this book, we assume that:

- You are familiar with WebSphere Business Integration Server Foundation V5.1

- You understand Eclipse and Java/J2EE platform.
- You have working knowledge of WebSphere Application Server V6.
- You have a ready environment with both WebSphere Integration Developer V6.0.2 and WebSphere Process Server V6.0.2 installed and configured.

2

Product overview

In this chapter, we discuss the following topics:

- 2.1, “Server Foundation overview” on page 10
- 2.2, “WebSphere Process Server overview” on page 15

2.1 Server Foundation overview

In this chapter, we introduce WebSphere Business Integration Server Foundation concepts and architecture.

2.1.1 Introduction to Server Foundation

WebSphere Business Integration Server Foundation V5.1 is an evolution of the way the WebSphere family products implement 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 e-business on demand®. It represents an important step in the path that IBM proposed as an approach to build and deploy SOA-based applications.

The goal of this product is to increase the business flexibility by leveraging a service-oriented architecture to build modular applications that are designed to adapt quickly to change. WebSphere Business Integration Server Foundation proposes to build solutions as composite applications using a highly-integrated development environment.

2.1.2 Architectural overview of Server Foundation

Figure 2-1 on page 11 shows the architecture of WebSphere Business Integration Server Foundation as an application server with extended capabilities.

WebSphere Business Integration Server Foundation uses the functionalities of the WebSphere Application Server and introduces some J2EE applications in order to offer a complete SOA platform. The key additional services that WebSphere Business Integration Server Foundation provides are:

- The Business Process Choreographer, which is a powerful container for executing business processes.
- Programming Model Extensions, which are a number of valuable extensions to the J2EE specification that are delivered in many different forms, which includes services, APIs, and tooling extensions, for example, Business Rule Beans, which is a business rule engine part of Server Foundation, is one of them. For more information, read 7.5, “Migrating Business Rule Beans” on page 165.
- 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

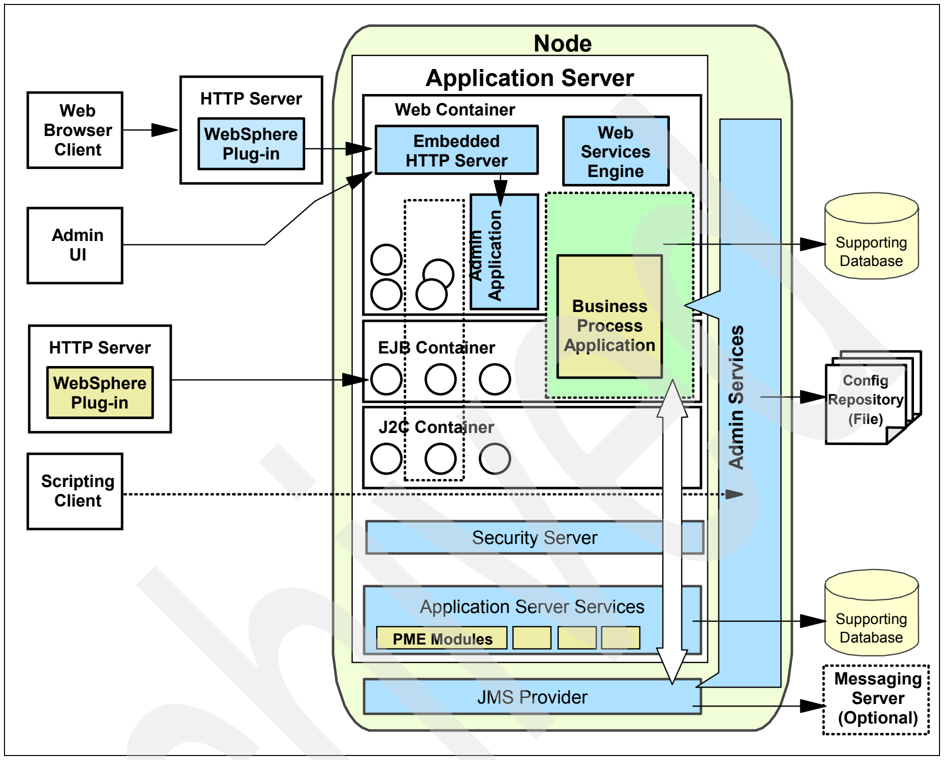


Figure 2-1 The WebSphere Business Integration Server Foundation architecture

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.

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 supports business processes and workflows.

The Business Process Choreographer that is within Server Foundation is the runtime environment for executing business processes. So, 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. The Business Process Choreographer consists of a a J2EE application, the Business Process Engine (BPE) container, a set of client interfaces, and administration and control facilities.

Business Process Engine Container Architecture

The Business Process Engine (BPE) container is a specialized J2EE application that executes business processes using the WebSphere Application Server runtime services and resources. It handles the life cycle of a process, from instantiation of a process template to final deletion of the completed process. Figure 2-2 shows the components of the container.

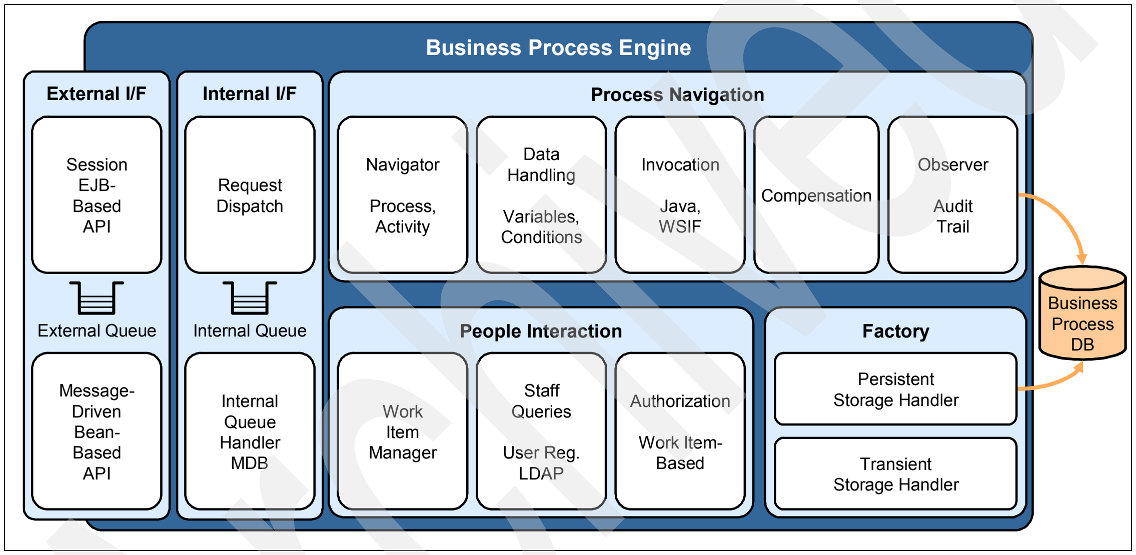


Figure 2-2 Business process engine container architecture

The functionality of the container is published as a Session EJB™ and a message driven bean. It provides an external queue for client interaction using JMS and uses internal queues and a relational database, 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 BPE 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 using its observer and audit trail components.

The navigator also includes functions to perform the following operations:

- Java snippet invocations.
- External process invocations. Allows the invocation of external processes using the Apache Web service invocation Framework.

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.

2.1.3 Features of Server Foundation

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

- Initial enablement for service-oriented architecture
- BPEL4WS process choreography version 1.1 with extensions
- Human workflow support
- Business rules beans support
- Application adapters
- Programming model extensions
- Common Event Infrastructure
- J2EE Application Server
- Integrated J2EE development environment

WebSphere Business Integration Server Foundation uses a 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 the Programming Model Extensions (PMEs).

The PMEs address the following areas:

- A Integrated J2EE based workflow, the Business Process Execution Container
- A common event infrastructure

- 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 the Server Foundation.

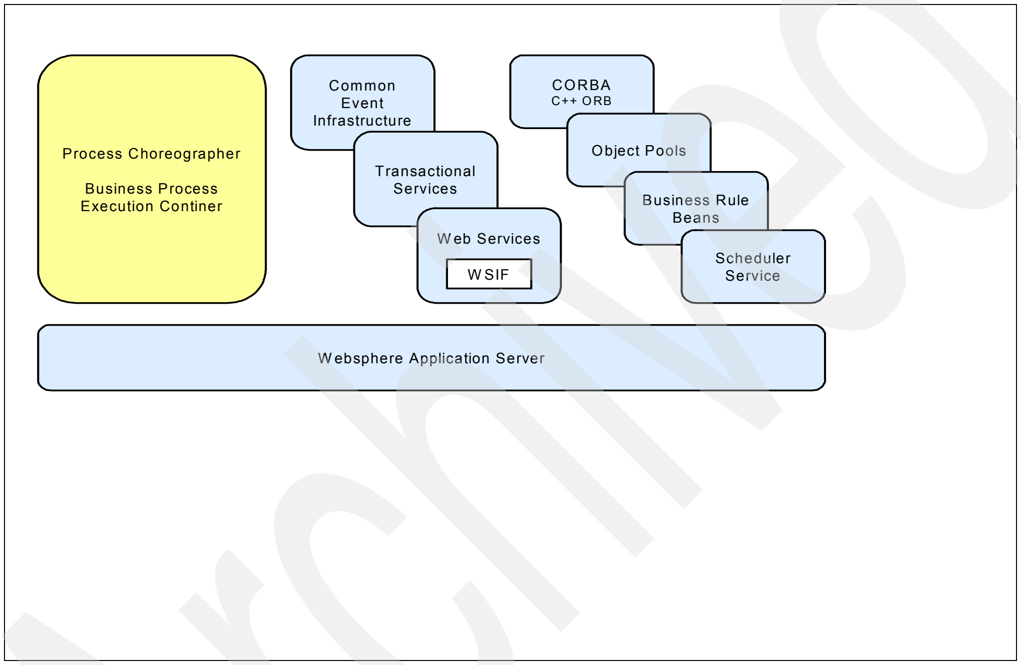


Figure 2-3 Elements of the Server Foundation

Because the scope of this book is oriented to the migration of processes from WebSphere Business Integration Server Foundation to WebSphere Process Server, we do not cover some of the 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.
- WebSphere Process Server offers a new Business Rules Service SCA implementation. This new service is not compatible with the Business Rules from V5.1. If you need to run the deprecated business rules on WebSphere

Process Server, there is an option during WebSphere Process Server install to install support for the deprecated Business Rules features.

- Web services. We do not cover the Web services procedure migration from J2EE 1.3 to J2EE 1.4. The J2EE Migration wizard does not perform the migration of secure Web service. The Web service Invocation Service platform that is used in the Web service implementation of WebSphere Business Integration Server Foundation is not used in the WebSphere Process Server.

2.2 WebSphere Process Server overview

In this section, we introduce WebSphere Process Server concepts and architecture.

2.2.1 Introduction to WebSphere Process Server

IBM WebSphere Process Server is an advanced business process integration server that is built over the industry standard service-oriented architecture.

WebSphere Process Server uses the Service Component Architecture (SCA) programming model and the Service Data Objects (SDOs) data model. Business objects are transformed, routed, and mapped 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' Business Process Execution Language (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 the process server and can be used to provide the connectivity to Web services and the mediation flows. You can use the Common Event Infrastructure (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, and you can use WebSphere Monitor to monitor the business processes that are running on WebSphere Process Server.

IBM WebSphere Process Server is based on the Java 2 Enterprise Edition (J2EE) 1.4 infrastructure and the WebSphere Application Server platform. It is built on WebSphere Application Server, Network Deployment.

2.2.2 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 Service Component Architecture (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 qualities, such as clustering, failover, scalability, and security to WebSphere Process Server.

WebSphere Process Server has three logical layers: the SOA core layer, the supporting services layer, and the service components layer.

The SOA Core layer consists of the Service Component Architecture, the Business Objects, and the Common Event Infrastructure component. Service Component Architecture and the Business Objects provide the universal invocation and the data representation programming models. Common Event Infrastructure generates events for monitoring applications that are running on WebSphere Process Server.

The supporting services layer provides the components that are necessary for transformations of Business Objects and the Interfaces.

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.

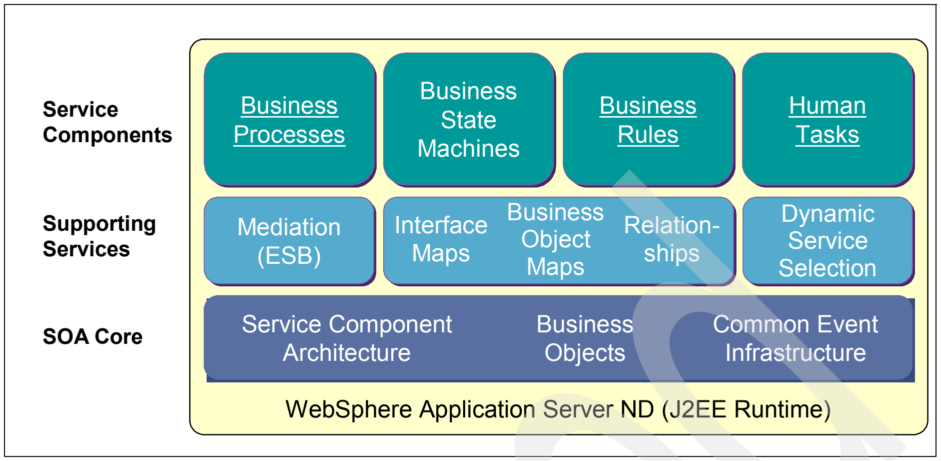


Figure 2-4 Components of the WebSphere Process Server

Service-oriented architecture core

The service-oriented architecture core of the WebSphere Process Server provides the universal invocation (Service Component Architecture) and the data-representation (Business Objects) programming models as well as the monitoring and management capabilities (Common Event Infrastructure) for the applications that are running on WebSphere Process Server. IBM Enterprise Service Bus is also a part of the service-oriented architecture core of WebSphere Process Server, where it provides the transformation and mediation services.

Figure 2-5 shows the service-oriented architecture core layer of the WebSphere Process Server.

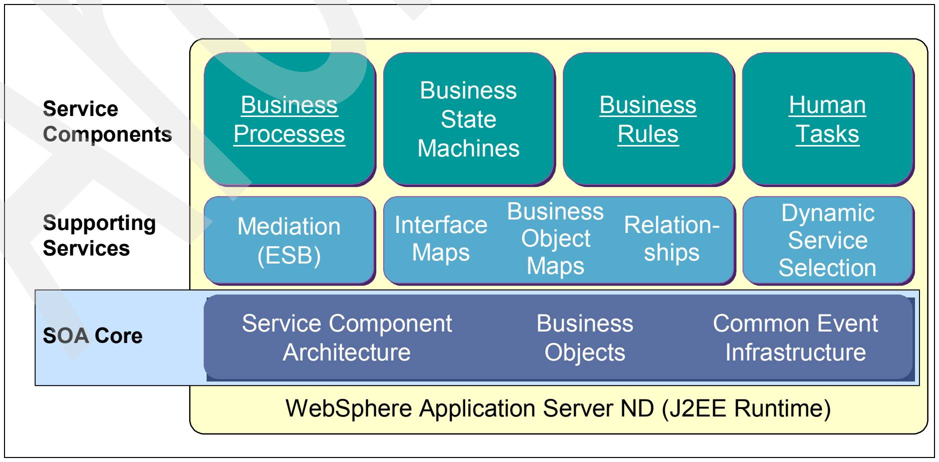


Figure 2-5 Service-oriented architecture core layer of WebSphere Process Server

Service Component Architecture

Service Component Architecture provides an abstraction layer around the Service components that separate the Business logic from the integration logic so that 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 Service Component Architecture has an Interface, so that the service can be used or invoked, an Implementation, and a Reference if the component wants to use the services of another component, as shown in Figure 2-6.

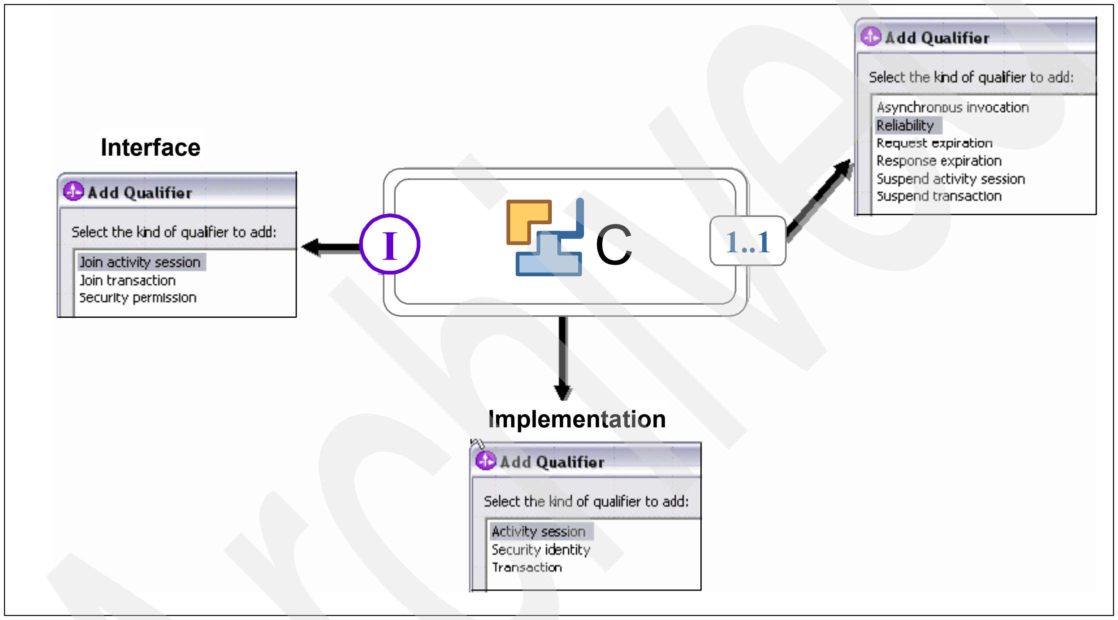


Figure 2-6 Service Component Architecture

Service Data Objects and Business Objects

Data that is exchanged between the components that are running on WebSphere Process Server is represented 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 Service Data Objects (SDOs), which are part of the WebSphere Application Server.

Service Data Object (SDO) is a framework for the data application development. The Service Data Object abstracts the data in a service-oriented way and provides a technology independent representation of the data that enables the developers to work with the data with out having to worry about the technology-specific API to access and use the data.

Business Objects include some extensions to Service Data Objects 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 Common Event Infrastructure that is within the WebSphere Process Server provides the event management services.

The Common Event Infrastructure Provides functionality for generation, propagation, persistence, and consumption of events that represent service component processes. Events are represented using the Common Base Event model, which is a standard, XML-based format that defines the structure of an event. Common Event Infrastructure 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 the service provider.

Supporting services

The supporting services layer in the 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, addressing the transformation challenges for integrating applications.

Figure 2-7 on page 20 shows the supporting services layer of the WebSphere Process Server.

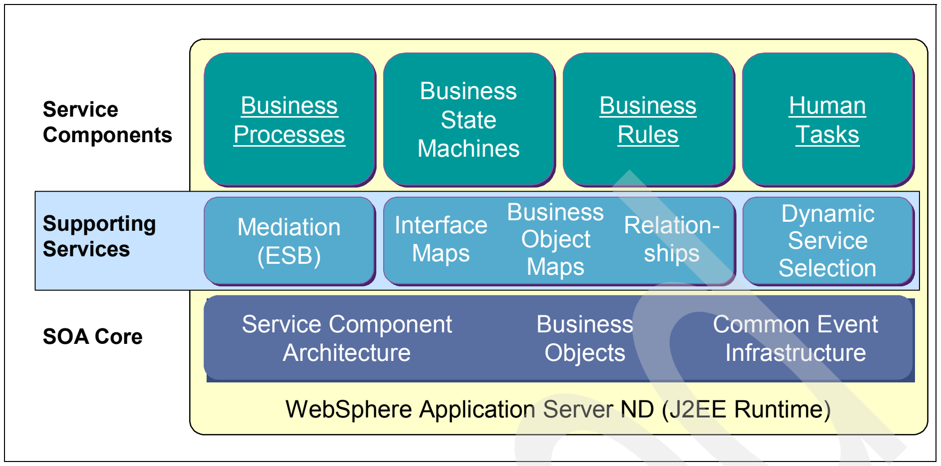


Figure 2-7 Supporting services layer of WebSphere Process Server

Mediation flows

Mediation flows are part of the WebSphere Enterprise Service Bus intercept, and 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 is 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 with out 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, business state machines) are represented as components with well-defined interfaces. Within the Service Component Architecture, 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, which enables you to 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 Enterprise Information Systems (EIS) using WebSphere Adapters and WebSphere Business Integration Adapters.

Figure 2-8 on page 22 shows the service component layer of WebSphere Process Server.

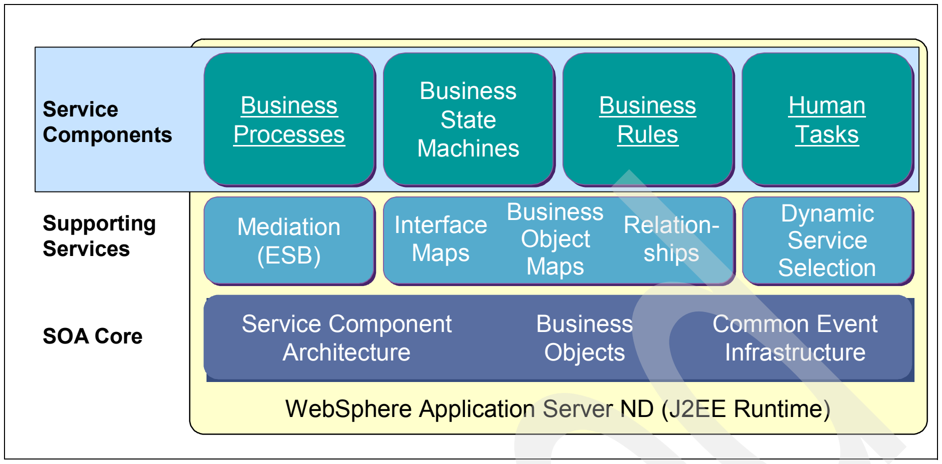


Figure 2-8 Service component layer of WebSphere Process Server

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 Business Process Execution Language (BPEL) engine. WebSphere Process Server includes a Business Process Choreography engine on top of the WebSphere Application Server, which fully supports the BPEL specifications and also 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 as well as 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. Using business rules within a business process allows applications to respond quickly to changing business needs.

2.2.3 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 service-oriented architecture (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 are built on service-oriented architecture and use the Web services Business Process Execution Language.
  - BPEL is an industry standard specification.
- Uses Service Component Architecture, which is a component-based framework that separates business logic from integration logic.
  - Service Component Architecture allows developers to 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 using WebSphere Modeller and WebSphere Monitor.
  - WebSphere Modeler allows business analysts to 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 IBM WebSphere Integration Developer.
  - WebSphere Integration Developer is a simplified development tool with visual editors for component development, assembly, integrated testing, and deployment.
- Supports 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 multi-level approvals.
- Business rules, business state machines, and selectors to dynamically choose interfaces based on business scenarios.
  - 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.
  - The business rules engine allows you to dynamically change the outcome business process without changing the actual business process.
  - The business state machine allows you to model the business process based on states (Ad-Hoc) rather than sequential order.
- Broad reach in integration with the support for IBM 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 Enterprise Information Systems.
- Manage and monitor events using Common Event Infrastructure.
- Service governance with dynamic runtime lookup and invocation of services using WebSphere Service Registry and Repository.
- Ability to change business processes with minimal programming skills, without redeploying the application.
- Dynamic endpoint administration 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 monitor (ARM), and support for the Tivoli ITCAM suite of products.

- Integrated enterprise service bus with pre-built mediations, which increases business flexibility and responsiveness and enhances usability, which saves time and development costs.
- WebSphere Process Server is built over WebSphere Application Server Network Deployment, leveraging the transaction, security, clustering, and workload management capabilities of WebSphere Application Server.

3

Migration concepts

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

In this chapter, we discuss:

- 3.1, "Migration concepts overview" on page 28
- 3.2, "Features mapping and migration approach" on page 29
- 3.3, "Benefits of migrating to WebSphere Process Server" on page 35

3.1 Migration concepts overview

WebSphere Business Integration Server Foundation and WebSphere Process Server are architecturally different products that address the Business Process Management space. Because of this fundamental difference, the migration of WebSphere Business Integration Server Foundation to WebSphere Process Server is semi-automatic and manual intervention might be needed to migrate some features as well as to optimize the processes for WebSphere Process Server.

It is important to know the new and enhanced, changed, or equivalent features of WebSphere Process Server for a WebSphere Business Integration Server Foundation project that 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 also have to 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 a migration process and best practices to make the migration smooth.

The different steps in the migration process are:

- Make an assessment regarding the need for the migration, risks, different options for migration, and the current environment.
- Have the subject matter expert prepare and analyze a plan, keeping the current environment in mind. Make any changes, which are necessary to make a smooth migration, to the WebSphere Business Integration Server Foundation libraries.
- Migrate the environment in a phased manner following the migration plan that was prepared in the previous bullet. The migration is not fully automatic and you need to make the necessary changes for the new WebSphere Process Server environment.
- Regressively test the migrated environment to preserve all of the functionality and the business logic.
- After the testing is complete with satisfactory results, deploy the migrated components into production.

Figure 3-1 shows the migration process.

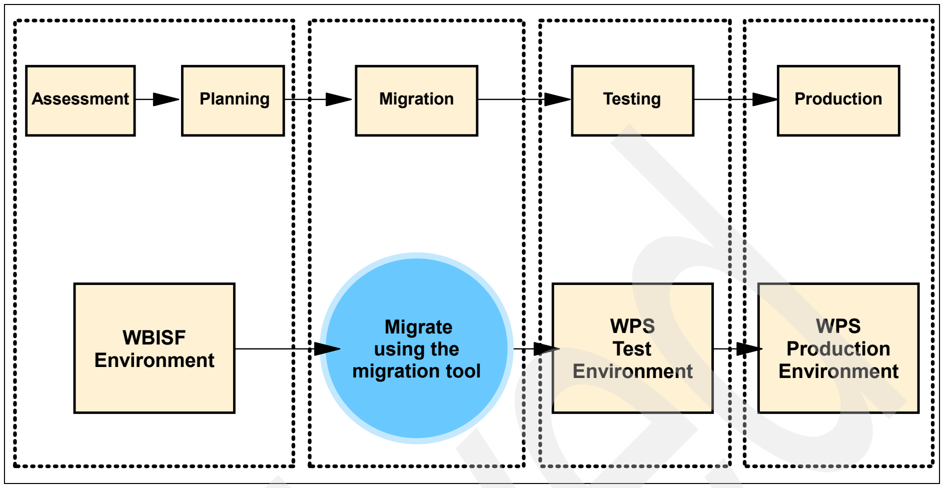


Figure 3-1 Migration process

3.2 Features mapping and migration approach

WebSphere Process Server V6 is a service-oriented architecture 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 that generates events for monitoring applications. WebSphere Process Server V6.0.2 provides many key enhancements around service integration, quality of service, performance, and ease of use.

3.2.1 General feature comparison

The major differences between the WebSphere Business Integration Server Foundation programming model and the WebSphere Process Server programming model are:

- Web service Invocation Framework-based data model in WebSphere Business Integration Server Foundation is now Service Data Object (SDO) in WebSphere Process Server
- Web service Invocation Framework-based invocation model in WebSphere Process Server is now SCA in WebSphere Process Server
  - Web services clients are now pure JAX-RPC clients rather than using Web service Invocation Framework

WebSphere Process Server also contains several new features that do not correspond to WebSphere Business Integration Server Foundation features, such as 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 V6.0.2 and WebSphere Business Integration Server Foundation V5.1.

Table 3-1 Features Comparison

	WebSphere Business Integration Server Foundation V5.1	**WebSphere Process Server V6.0.2**
J2SE™ Version	1.4	1.5
J2EE Version	1.3	1.4
Data model	Web service Invocation Framework	Service Data Object (SDO)
Invocation model	Web service Invocation Framework	Service Component Architecture (SCA)
Business Object Maps	No	Yes
Interface Maps	No	Yes
WebSphere Enterprise Service Bus Mediations	No	Yes
Selectors	No	Yes
Relationships	No	Yes
Business Rules	Partial support through Business Rule Beans	Yes
Business Processes	Business Process Execution Language for Web services (BPEL4WS) version 1.1.	BPEL4WS version 1.1 with major capabilities of the upcoming Web services Business Process Execution Language (WS-BPEL) version 2.0 specification.
Business State Machines	No	Yes

	WebSphere Business Integration Server Foundation V5.1	WebSphere Process Server V6.0.2
WS-Security support	Draft 13	Version 1.0
J2EE Connector Architecture (JCA) support	Version1.0	Version 1.0 and Version 1.5. Enhanced JCA, Version 1.5 resource adapter support that uses IBM WebSphere Adapters.
WebSphere Business Integration Adapters support	Yes	Yes, WebSphere Adapters

WebSphere Process Server V6.0.2 is not an extension for WebSphere Business Integration Server Foundation V5.1; instead, it depends on a new infrastructure, and therefore a direct feature mapping is not possible.

Programming Model Extensions

WebSphere Business Integration Server Foundation provides a number of valuable extensions to the J2EE specification, which 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 Server Foundation Programming Model Extensions and their availability in the Version 6 product stack.

Table 3-2 Programming Model Extensions and the product containing them in Version 6

Extension in Server Foundation	Available in V6 product
Activity Session	Application Server
Application Profiling	Application Server
Asynchronous beans	Application Server
Business Rule Beans	Process Server
Container Managed Persistence over Anything (CMP/A)	Process Server
Dynamic Query	Application Server
Extended Messaging (CMM)	Process Server

Extension in Server Foundation	Available in V6 product
Internationalization (I18N)	Application Server
Last Participant Support	Application Server
Object pools	Application Server
Scheduler	Application Server
Startup Beans	Application Server
Work areas	Application Server

For more detailed information about migrating applications using Programming Model Extensions that are now available in Application Server, see the WebSphere Application Server Info Center:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/
com.ibm.websphere.nd.doc/info/ae/ae/rins_migratepme.html
```

For information about migrating the Programming Model Extensions, which are part of Process Server, choose the following information resources:

- Migrating Business Rule Beans. For details, see 6.12, "Migrating Business Rule Beans applications" on page 137 and 7.5, "Migrating Business Rule Beans" on page 165.
- Container Managed Persistence over Anything (CMP/A). For details, see 6.13, "Migrating Container Managed Persistence over Anything applications" on page 138.
- Migrating Extended Messaging (CMM). For details, see 6.14, "Migrating Extended Messaging applications" on page 138 and 7.6, "Migrating Extended Messaging" on page 166.

3.2.2 Source artifacts, bindings, and the migration approach

Use a general source artifacts migration approach when migrating between the Server Foundation and the Process Server. To get an overview of how the bindings and artifacts, which are available in Server Foundation, are to be mapped and migrated to Process Server, we provided Table 3-3 on page 33 and Table 3-4 on page 34.

Table 3-3 on page 33 shows the binding mapping comparison between WebSphere Process Server V6.0.2 and WebSphere Business Integration Server Foundation V5.1 and the migration approach.

Table 3-3 Binding mapping

WebSphere Business Integration Server Foundation V5.1 binding	WebSphere Process Server V6 construct	Migration approach
EJB	Export with SCA binding	When a BPEL process in Server Foundation calls the EJB service, the Migration wizard creates an export with EJB binding and wires to the BPEL component in the Assembly Editor.
JMS	Export with JMS binding	The Migration wizard creates an export with JMS binding and wires it to the BPEL component in the Assembly Editor.
IBM Web service with (SOAP/HTTP)	Export with Web service binding (SOAP/HTTP)	The Migration wizard creates two exports, which are wired to the BPEL component. No import is created.
IBM Web service with (SOAP/JMS)	Export with Web service binding (SOAP/JMS)	The Migration wizard creates two exports, which are wired to the BPEL component. No import is created.
Apache Web service with (SOAP/HTTP)	Export with Web service binding (SOAP/HTTP)	The Migration wizard creates an import along with two exports. The exports are wired to the BPEL component, which then is wired to the import.

Table 3-4 shows the artifacts mapping between WebSphere Process Server V6.0.2 and WebSphere Business Integration Server Foundation V5.1 and the migration approach.

Table 3-4 Artifacts mapping

WebSphere Business Integration Server Foundation V5.1 artifact	**WebSphere Process Server V6 construct**	**Migration approach**
BPEL service	Process (BPEL) component	Automatically handled by the Migration wizard.
BPEL-to-BPEL invocation	Wire connection in the Assembly Editor if the BPEL exists in the same project. Using import with SCA binding if the BPEL is in another project.	Automatically handled by the Migration wizard.
EJB service	Import with EJB binding	Manual through the Assembly Editor. When the EJB service was called by a BPEL process in Server Foundation, the Migration wizard creates an export with EJB binding and wires to the BPEL component in the Assembly Editor.
JMS service	Import with JMS binding	Manual through the Assembly Editor. The Migration wizard creates an export with JMS binding and wires to the BPEL component in the Assembly Editor, if a BPEL process was targeted by this JMS service.
Web service (SOAP/HTTP or SOAP/JMS)	Import with Web service binding	Manual through the Assembly Editor. The Migration wizard creates two exports, which are wired to the BPEL component, if a BPEL process was targeted by this Web service.

WebSphere Business Integration Server Foundation V5.1 artifact	WebSphere Process Server V6 construct	Migration approach
Java service	Java component	Manual through the Assembly Editor. The Migration wizard creates a Java component wires it to the BPEL component, if a BPEL process was calling this Java service.
Transformer service	Data Maps and Interface Maps Or use the proxy that was generated in Server Foundation in a Java component in Process Server.	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.

3.3 Benefits of migrating to WebSphere Process Server

The following list contains the benefits of migrating to WebSphere Process Server:

- Superior architecture that is built over service-oriented architecture.
- Converged SOA Platform for Process and Application Integration. A single-integrated platform that supports a vast array of features that are needed for business integration, which includes Process Choreography, Business State Machines, Business Rules, Enterprise Service Bus (ESB), and Common Event Infrastructure (CEI).
- Supports 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.
- WebSphere Adapters are more efficient (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 using WebSphere Integration Developer.
- Robust server platform that is based on WebSphere Application Server Network Deployment.
- Simplified administration using the Admin Console.
- Supports dynamic runtime change of service endpoints.

For more benefits of WebSphere Process Server, refer to 2.2.3, “Features of WebSphere Process Server” on page 23.

4

Migration planning

In this chapter, we cover the most important aspects and considerations of a WebSphere Business Integration Server Foundation to WebSphere Process Server migration project.

In this chapter, we discuss:

- ► Migration project considerations
- ► Non functional aspects
- ► Product coexistence

4.1 Migration project considerations

In this section, we describe the complete life cycle of a WebSphere Business Integration Server Foundation to WebSphere Process Server migration project and important considerations to undertake.

4.1.1 Migration project phases

There are several approaches to migration, and the approach that we describe in this chapter is one of the proven safe approaches on migration projects because:

- It supports business continuity and reduces the risk of data loss because the new production environment is implemented separate from the old one.
- Changes have low impact on production.
- The included piloting phase helps to solve production problems, while impacting only selected few users.

Figure 4-1 on page 39 illustrates this migration approach.

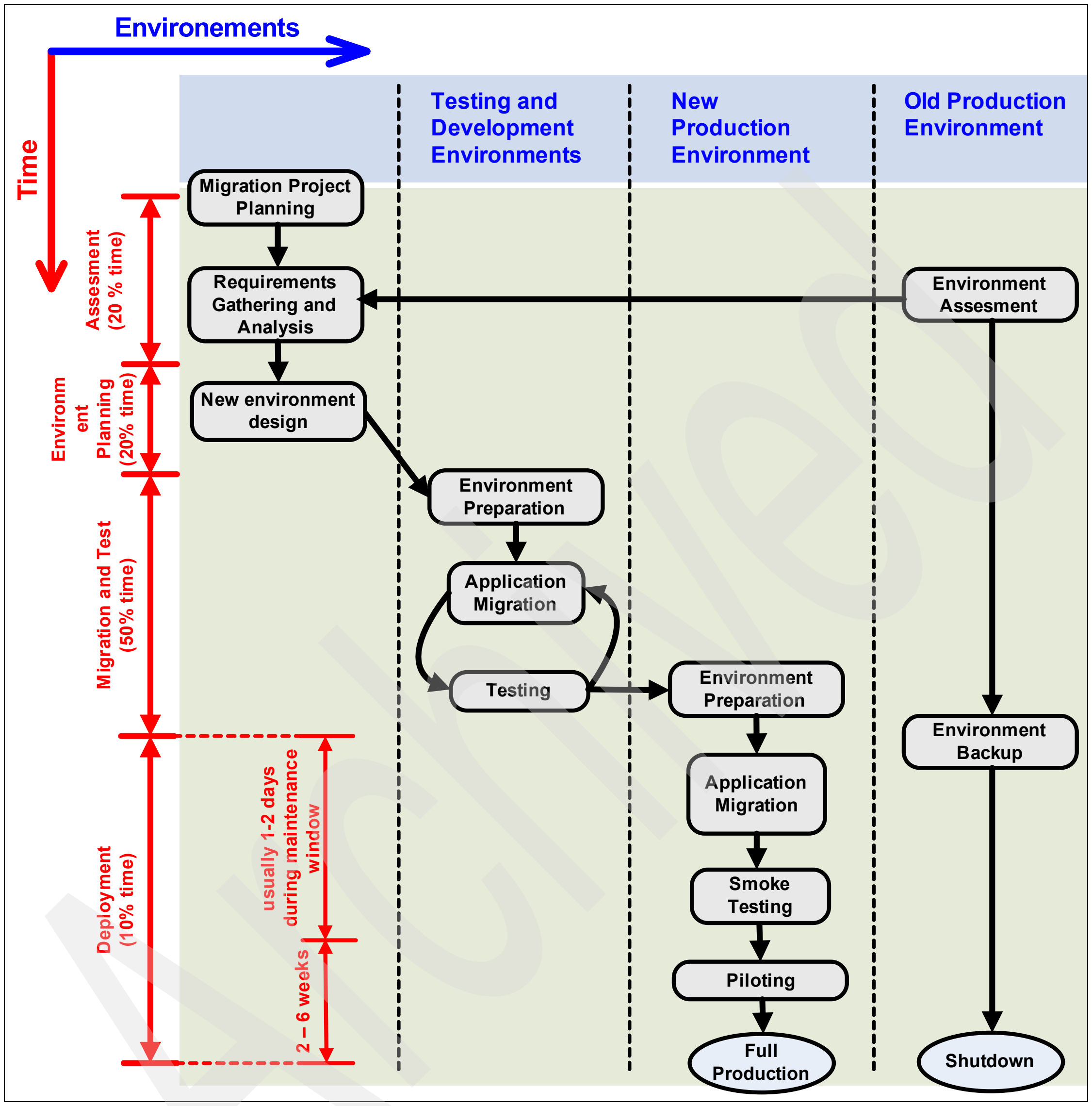


Figure 4-1 Activities in Server Foundation to Process Server migration project. Time percentage is indicative and can be different for your project.

4.1.2 Migration project roles

Table 4-1 describes the recommended roles for the migration project.

Table 4-1 Project roles in detail

Role	Description
Project manager	Project management, budget, and staffing coordination.
Architect	Designs a new environment. Keeps overall picture. Helps in designing integration and performance tests. Experience with infrastructure design, WebSphere Process Server, messaging, and databases and security are preferred.
Analyst	Gathers and analyzes requirements on the new environment, helps to design test scenarios, and verifies functional test results. Process modeling experience is worthy.
Server Foundation product specialists	Team of specialists that have hands-on-experience with Server Foundation, WebSphere MQ (if applicable), and databases.
WebSphere Process Server product specialists	Team of specialists that have hands-on-experience with WebSphere Process Server, service integration bus, and databases.
Developers	Team that does the migration of the applications - running migration tools, developing customizations, fixes, and others.
Testers	Testing team that designs and prepares tests and performs the tests.
Maintenance team	Team combined from developers, testers, product specialists, and administrators that can support the new environment and quickly develop and apply fixes.

Note: The project manager is used during the entire project life cycle.

Note: We recommend that you staff the project with experienced Process Server Specialists. Developers and specialists need to be experienced with Process Server projects already. We do not recommend that you perform a migration project as the first project on Process Server because there are changes to tooling and the runtime environment that require additional skills.

Table 4-2 describes which roles are required in each project phase and the expected percentage of utilization of role during the phase.

Table 4-2 Project phases mapped to project roles.

Phase	Roles
Migration project planning	Project Manager (PM) 60% *, Architect 40%
Requirements gathering and assessment	Analyst 70%, Architect 20%, PM 10%
Environment assessment	Server Foundation product specialists 80%, architect 20%
New environment design	Architect 40%, WebSphere Process Server specialists
Infrastructure preparation	Product specialists (WebSphere Process Server 40%, messaging 30%, database 30%)
Application migration	Developers 80%, product specialists 20%
Full testing	Testers 60%, 20% analysts, 20% developers
Environment backup	Maintenance team
Smoke testing	Testers 80%, analysts 20%
Piloting	Maintenance team 100%

4.1.3 Assessment

The tasks that need to be performed during the assessment phase are:

- Planning:
  - Migration project planning.
  - Preparation of the migration project plan (phases, dependencies, staffing, and budget).
- Assess the old environment. Assess the following items:
  - Service Level Agreement (SLA) - the same SLA is most likely provided by the new environment.
  - Availability and scalability of the old environment.
  - Performance - throughput, utilization, response times, and others.
  - Security - security zones, users, roles and groups, and access rights.
- Requirements gathering and analysis:
  - Gather additional requirements from Information Technology (IT) and business departments and consolidate them with old environment

assessment results. These additional requirements include changes in SLA, functionality changes that can be connected with the migration, technological changes, and others.

4.1.4 Environment planning

Within this phase, you plan your new environment. Incorporate the requirements that you obtained in the previous phase to the planning. Based on the results of this phase, the new environment is prepared.

Perform the following steps to achieve the expected results:

1. Draw and document the hardware and software topology of your new environment, which includes hardware and middleware software components. This step meets all of the requirements that were gathered. The most important considerations to be made are:
   - Performance considerations.
   - Availability and scalability considerations.
   - Security considerations.
2. Prepare a list of required hardware and software upgrades.
3. Define the migration path, and decide which components are to be migrated one-to-one. Also decide what needs to be customized and newly developed. You can also consider complete new modules.

4.1.5 Migration

In this phase, prepare the environment and perform the migration (in development or testing environment). All of the necessary development and configuration work is completed in this phase.

Perform the following tasks in this phase:

1. Prepare the environment:
   - Prepare the new environment (hardware infrastructure, data, middleware, user accounts, and others).
2. Migrate the application:
   - Process modeling.
   - Migrate applications from WebSphere Business Integration Server Foundation to WebSphere Process Server.

  - Configuration work.
  - Customization and new development.
  - Preparation of additional migration scripts.

4.1.6 Testing

Testing is done in iterations together with migration until acceptance criteria are met. Testing assures that the migration approach is correct and that the applications work correctly in the new environment.

Perform the following tasks in this phase:

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

Types of tests to be performed:

- Functional testing
- Integration testing
- Performance testing
- Infrastructure testing

4.1.7 Deployment

Deployment is probably the most complex and risky phase of the migration. It is necessary to assure business continuity, no data loss, and still have the possibility to roll back if significant problems appear.

Perform the following tasks in this phase:

1. Environment backup

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

2. Application migration

   Migrating production applications and deploying them to the new (preproduction) environment. Both the migration and the deployment are properly tested upfront and automatized 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. Smoke testing

   Quick retest that the applications were migrated correctly.

4. Piloting

   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 end users are using the new environment. This approach reduces risk by the following:

   – 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. Shutdown 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.

Note: There might be cases where a longer period of coexistence is needed to, for example, finish long running processes on the Server Foundation environment. We do not cover this case here. See Chapter 5, "Migration options" on page 55 for more information about this approach.

4.2 Non functional aspects

In this section, we describe important non functional aspects that you need consider in your planning.

4.2.1 Availability considerations

The deployment topology of Server Foundation is slightly different than WebSphere Process Server. Therefore you need to asses 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. Figure 4-2 on page 45 shows the relationship between the level-of-availability and costs.

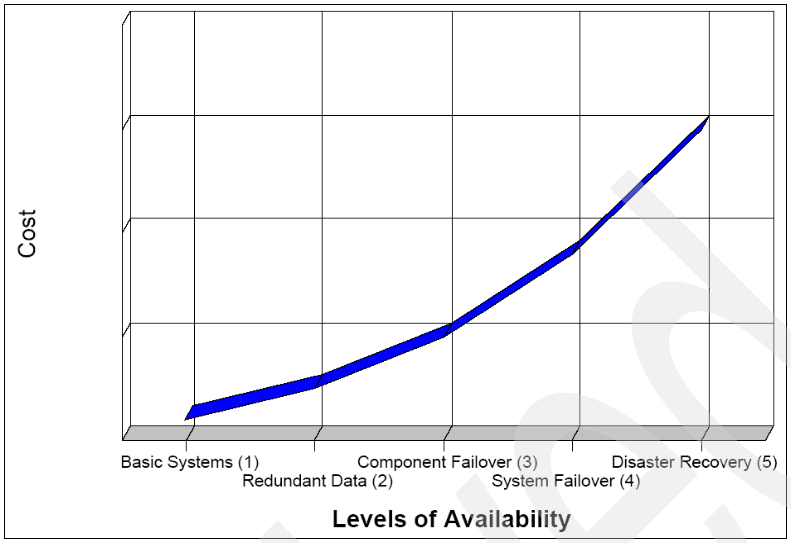


Figure 4-2 Relationship between level-of-availability and costs

The levels-of-availability are:

Basic systems	No protections of data and services, backups only.
Redundant data	Disk redundancy or disk mirroring are used to protect against data loss.
Component failover	Hardware and software components are doubled, which means that there are at least two instances of the same component in the system. In case of failure of one, the second takes over the load. 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	Similar to system failover, but 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.

High availability in 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 WebSphere clustering, 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 4-3.

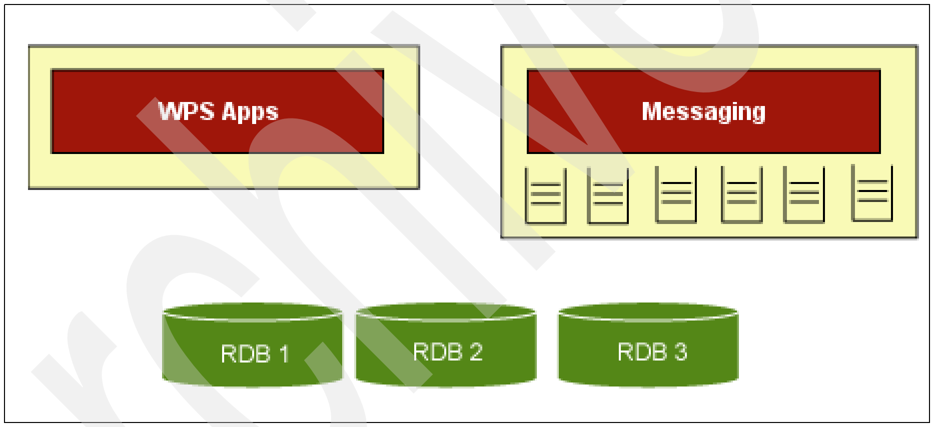


Figure 4-3 Components to be clustered in WebSphere Process Server

- WebSphere Process Server applications

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

- Relational databases

  WebSphere Process Server requires certain application configuration and runtime information to be stored in relational database tables. The messaging infrastructure also uses relational database tables for persistence.

- 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 (SI 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. But since the service integration bus is used, which requires a WebSphere Application Server to run, the messaging infrastructure can also be clustered using the WebSphere clustering techniques. However, there are a number of aspects to consider when you select the topology to adopt. If, for the Business Process Choreographer, you choose the messaging infrastructure WebSphere MQ, you have additional considerations on this one as well.

Refer to the following articles for or more details and recommendations about a Process Server environment for your specific requirements:

- *Basic steps for clustering WebSphere Process Server*

  `http://www-128.ibm.com/developerworks/websphere/techjournal/0604_chilanti/0604_chilanti.html`

- *Selecting your deployment pattern*

  `http://www-128.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html`

- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-66888

  `http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SG246688`

4.2.2 Security considerations

Server Foundation and Process Server security 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.

Securing the environment

Securing the environment involves enabling global security, creating profiles with security, and restricting access to critical functions to selected users:

- User registry - WebSphere Process Server supports various user registries (local operating system, Lightweight Directory Access Protocol (LDAP), custom registry, and federated repositories) that stay the same in comparison to Server Foundation.
- Transport layer security (Secure Socket Layer - SSL).
- Service Integration Bus security:
  - JMS or WebSphere MQ security.
  - Web services security.

Securing applications

Securing an application involves four main aspects:

- Authentication: A user or a process that invokes an application must be authenticated.
- Access control: Does the authenticated user have permission to perform the operation?
- Integrity and privacy of the data that an application accesses.
- Identity propagation with single sign on: Permits a user to provide authentication data one time and then passes this authentication information to downstream components.

The Business Process Choreographer components are implemented as WebSphere J2EE applications and thus adopt security mechanisms of WebSphere Application Server. WebSphere Application Server implements and extends Java 2 and J2EE security standards. Consider and configure security of these artifacts:

- Application security: Securing application resources, mapping security roles to users and groups, and others. Securing both Web and EJB applications.
- Client security.

Refer to the WebSphere Process Server Info Center for more comprehensive information about creating a secure WebSphere Process Server environment.

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp`

The following developer works article provides information about requirements for security in WebSphere Process Server and how to leverage security features of WebSphere Application Server V6.

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0602_khangaonkar/0602_khangaonkar.html

4.2.3 Performance considerations

In this section, we provide details that help you to assess and to 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 Server Foundation and Process Server.

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

The performance considerations are:

- The safest approach to size your WebSphere Process Server environment is definitely to prepare and run performance tests on a proof-of-concept and derive your sizing based on its result. Your proof of concept needs to:
  - Cover the typical transactions that are provided by your application.
  - Testing environment and transaction distribution are as close to the production environment as possible.
  - The sizing of your testing environment can 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).
  - 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.

- Sometimes it is better to modify or to 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.

Tuning recommendations

In this section, we discuss performance tuning tips for WebSphere Process Server.

- 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) datastores on high-performance DBMS.
- Configure threads for messaging and work managers.
- Configure the statement cache for long running processes.
- Use the appropriate Java heap size in all environments (collect verbose gc and graph it).
- JVM™
  - 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.
  - Adjust the heap size, if java.lang.OutofMemory occurs.

The Redpaper *WebSphere Business Integration V6 Performance Tuning*, REDP-4195 provides a more detailed explanation on this topic.

http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=REDP-4195

4.3 Product coexistence

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

- Both environments are up and running during the piloting phase.
- Migrate your applications in several stages.
- Achieve a period of coexistence to finish long running processes on Server Foundation while you start and work on migrated processes in the new Process Server environment.

A Server Foundation and Process Server installation can run on the same machine; therefore, ensure that they are not running in port conflicts. Refer to the documentation available at the following Web site for further detail:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/iins_coexist_top.html

You might need to call services or even processes on the Server Foundation environment from the new Process Server environment. This is possible using usual methods, such as RMI calls or Web services calls. However, make these scenarios part of your test plan to assure the interoperability between both systems.

Part 2

Migration procedure

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

Chapter 5, “Migration options” on page 55 introduces tools that are used during migration, for example WebSphere Integration Developer. In this chapter, we focus on the tools’ capabilities with regards to migration. We also describe the common and supported paths for migrating WebSphere Business Integration Server Foundation to WebSphere Process Server.

Chapter 6, “Migrating build time artifacts” on page 91 explains how to migrate artifacts that were developed in WebSphere Application Developer Integration Edition or WebSphere Business Integration Modeler.

Chapter 7, “Migrating runtime components” on page 147 considers restrictions and approaches for migrating the runtime topology from WebSphere Business Integration Server Foundation to WebSphere Process Server.

Chapter 8, “Best practices” on page 169 discusses best practices for the migration procedure.

Chapter 9, “Technical scenarios” on page 185 includes step-by-step instructions on two migration scenarios and illustrates how to use the migration tools.

Chapter 10, “Troubleshooting” on page 273 covers errors observed during migration and their resolution.

Migration options

In this chapter, we discuss:

- 5.1, “Migration paths” on page 56
- 5.2, “Migration tools” on page 59
- 5.3, “Migration approaches” on page 79

5.1 Migration paths

For all of the similarities in goals and functionality between WebSphere Business Integration Server Foundation V5.1 and WebSphere Process Server V6.0.2, the implementation differences between both products are substantial. There is no support for a binary or runtime migration from WebSphere Business Integration Server Foundation to the WebSphere Process Server; therefore, you must perform a source artifact migration.

There are several paths to migrate business processes and staff artifacts to WebSphere Process Server. They vary in the kind of business process that is used and the way this artifact was initially developed. The migration paths are:

- Migrating business processes that are described in BPEL
- Migrating business processes that are described in FDML
- Migrating business processes that are mainly developed in WebSphere Business Integration Modeler

We explain these paths and the tools that assist with migrating in this chapter.

WebSphere Business Integration Server Foundation has more capabilities than its component, Business Process Choreographer. However, because this platform is the basis for process choreography, this section focuses on migration paths for business processes.

For migration considerations of other Server Foundation components, refer to the following:

- 6.12, “Migrating Business Rule Beans applications” on page 137
- 6.14, “Migrating Extended Messaging applications” on page 138
- 6.13, “Migrating Container Managed Persistence over Anything applications” on page 138
- 7.5, “Migrating Business Rule Beans” on page 165
- 7.6, “Migrating Extended Messaging” on page 166

5.1.1 Migrating business processes described in BPEL

You can migrate BPEL business processes that are developed in WebSphere Studio Application Developer Integration Edition with the assistance of the WebSphere Integration Developer Migration wizard, which is a tool to migrate WebSphere Business Integration Server Foundation service project source artifacts.

Figure 5-1 shows the migration path for BPEL processes.

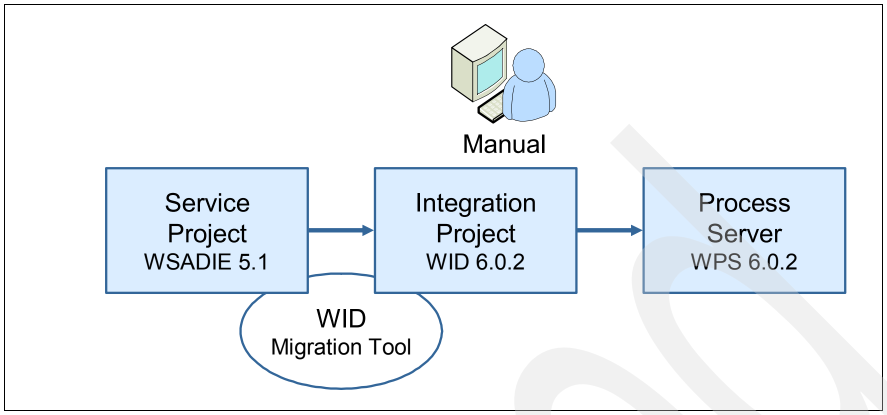


Figure 5-1 Migration path for BPEL processes

For more information about the Integration Developer Migration wizard, in general, see 5.2.1, "WebSphere Integration Developer overview" on page 59.

For information about using this Migration wizard, refer to:

- 6.4.1, "Overview of the Migration wizard" on page 101
- 6.4.2, "Using the Migration wizard" on page 101

5.1.2 Migrating business processes described in FDML

In WebSphere Studio Application Developer Integration Edition, processes are described either using BPEL or using a proprietary format known as Flow Definition Markup Language (FDML). FDML is deprecated but is still supported in WebSphere Business Integration Server Foundation 5.1.1.

When you attempt to migrate FDML processes to the Process Server, an additional migration step is necessary when you use the provided migration tools. To use the Integration Developer Migration wizard, you must describe the business process in BPEL. Therefore, you must first migrate the FDML process to a Server Foundation-compliant BPEL process. Afterwards, migrate this BPEL process to the Process Server.

Figure 5-2 illustrates the migration path for FDML processes.

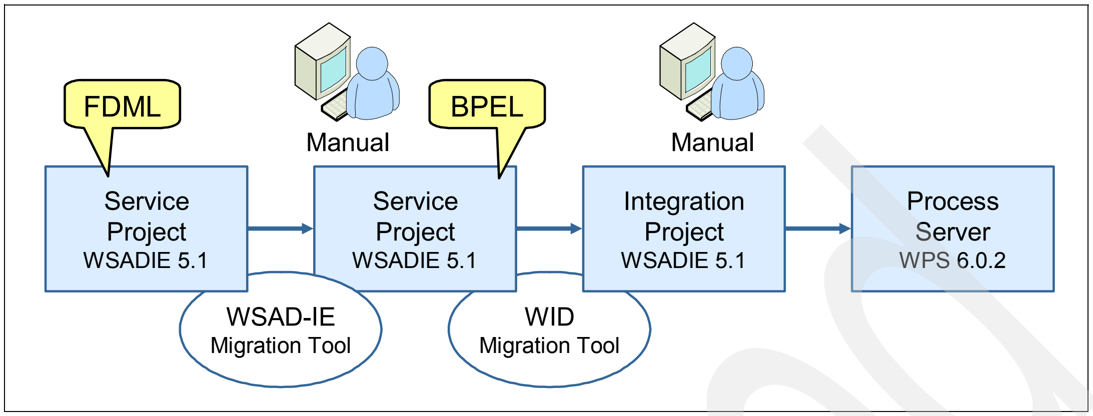


Figure 5-2 Migration path for FDML processes

The WebSphere Studio Application Developer Integration Edition Info Center provides a description of the steps for migrating the FDML process to a BPEL business process:

`http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.etools.iemigrate.doc/tasks/tmigin.html`

5.1.3 Migrating business processes mainly defined in 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.

In this case, migrate to WebSphere Business Modeler Version 6, and from this model, export to WebSphere Integration Developer, and manually integrate the services and bindings in the new Service Component Architecture (SCA) platform. This path is useful in the following scenarios:

- The project needs to use the modeling features of WebSphere Business Modeler, for example, when Modeler is used to improve the model, run simulations, or define metrics.
- Many activities of the project are human-task activities.
- The project wants to explore the monitoring functionalities of WebSphere Business Monitor V6.

Figure 5-3 illustrates the migration paths from a Modeler V5 project.

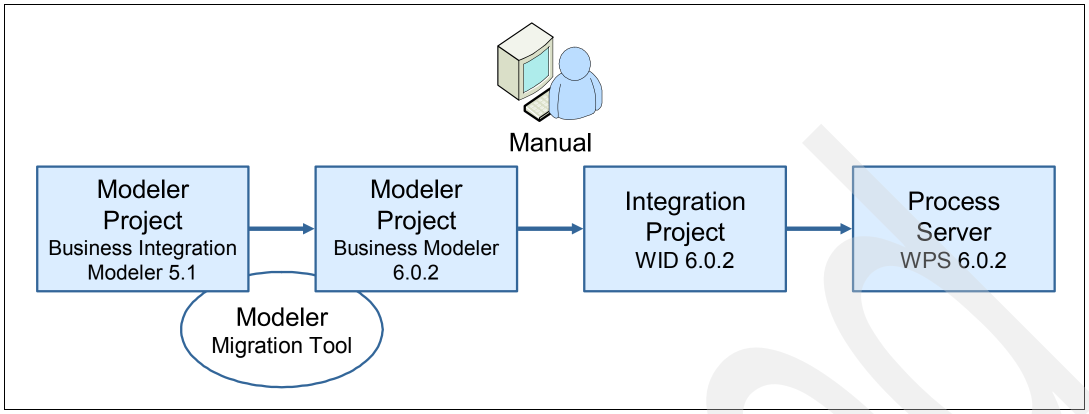

Manual
Modeler Project Business Integration Modeler 5.1
Modeler Project Business Modeler 6.0.2
Integration Project WID 6.0.2
Process Server WPS 6.0.2
Modeler Migration Tool

Figure 5-3 Migration paths from a Modeler V5 project

WebSphere Business Modeler V6 provides several options to migrate Modeler V5 projects. For more information, see 5.2.2, “WebSphere Business Modeler” on page 72.

5.2 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.

5.2.1 WebSphere Integration Developer overview

IBM WebSphere Integration Developer is the development environment for building integrated business applications that are targeted for WebSphere Process Server. It contains a broad array of capabilities that include all of the tools and development aids that are necessary to build a service-oriented architecture and Process Integration solutions.

In this section, we introduce general capabilities of WebSphere Integration Developer that includes general information about the Migration wizard. For detailed information about how to use the Integration Developer Migration wizard, refer to 6.4.2, “Using the Migration wizard” on page 101.

WebSphere Integration Developer is built on the Rational® Software Development Platform, which is based on Eclipse 3.0 technology. It provides full integration with other IBM tools and products. One of the primary purposes of

WebSphere Integration Developer is to provide the appropriate tools to easily build and test Service Component Architecture-based applications.

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

`http://www.eclipse.org`

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

We discuss the following items:

- Workbench
- Module
- Shared library
- Editors - Assembly, Process, Business object, Interface, Interface mapping, Relationship, and Visual snippet
- Enterprise service discovery wizard
- Integration test client

Workbench

When a new workspace is started, the Welcome page is displayed, as shown in Figure 5-4 on page 61. From this page, you can access information, such as the product overview, cheat sheets, tutorials, samples, migration information, and Web resources.

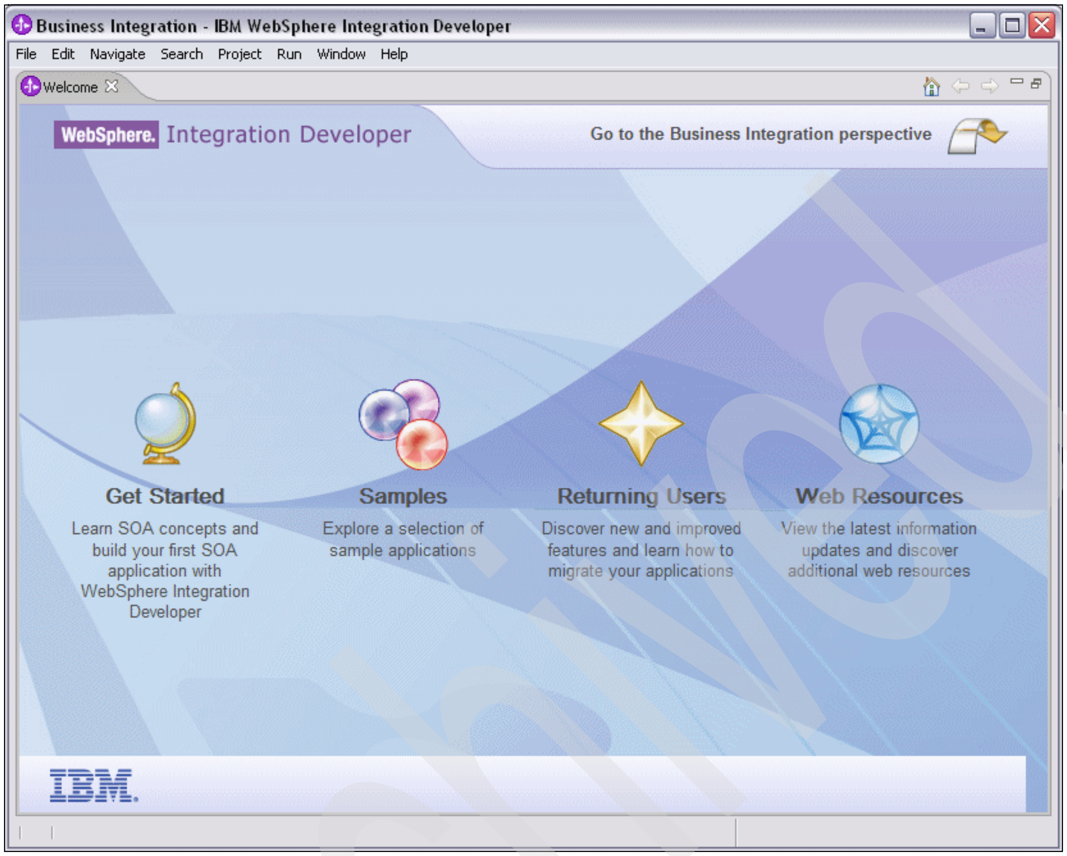
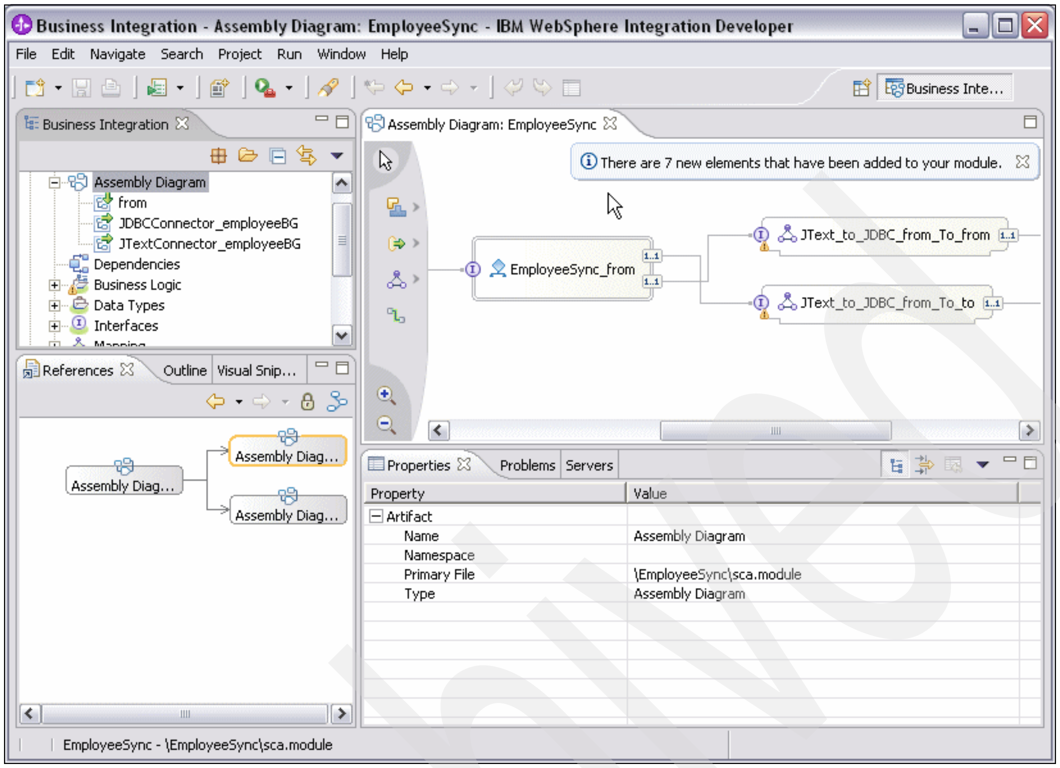

Business Integration - IBM WebSphere Integration Developer
File Edit Navigate Search Project Run Window Help
Welcome
WebSphere. Integration Developer
Go to the Business Integration perspective
Get Started
Learn SOA concepts and build your first SOA application with WebSphere Integration Developer
Samples
Explore a selection of sample applications
Returning Users
Discover new and improved features and learn how to migrate your applications
Web Resources
View the latest information updates and discover additional web resources
IBM.

Figure 5-4 Welcome page

Close the Welcome page, and click **Go to the Business Integration perspective**, located in the right-top corner, which opens the Business Integration perspective.

The workbench, as shown in Figure 5-5 on page 62, is where development and authoring of the migrated artifacts, as well as other integration modules, occurs. It offers a choice of perspectives and an array of toolbars and menu items that accomplish a variety of tasks.



There are 7 new elements that have been added to your module.

Property	Value
Artifact	
Name	Assembly Diagram
Namespace	
Primary File	\EmployeeSync\sca.module
Type	Assembly Diagram

Figure 5-5 Workbench

Editors

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

Assembly editor

Use the WebSphere Integration Developer assembly editor to build applications by assembling the SCA components. Drop the SCA components into the canvas, such as Business Processes, components, imports, and exports, specify their interfaces and bindings, and wire them together using the Assembly Diagram editor as shown in Figure 5-6 on page 63.

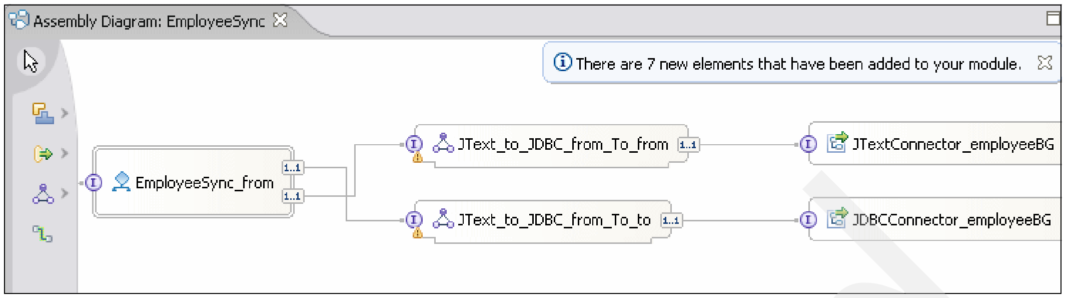
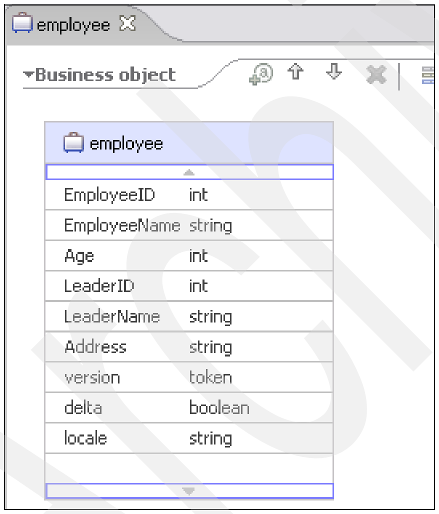

Assembly Diagram: EmployeeSync
There are 7 new elements that have been added to your module.
EmployeeSync_from
JText_to_JDBC_from_To_from
JText_to_JDBC_from_To_to
JTextConnector_employeeBG
JDBCConnector_employeeBG
1..1

Figure 5-6 Assembly editor

Business object editor

The Business object editor enables you to build and edit business objects, their attributes, and business graphs through a graphical interface, as shown in Figure 5-7. Use this editor to add, delete, reorder, and change the type of an attribute.



EmployeeID	int
EmployeeName	string
Age	int
LeaderID	int
LeaderName	string
Address	string
version	token
delta	boolean
locale	string

Figure 5-7 Business object editor

Process editor

The process editor is a graphical programming environment that allows you to visually create and manipulate business processes, as seen in Figure 5-8.

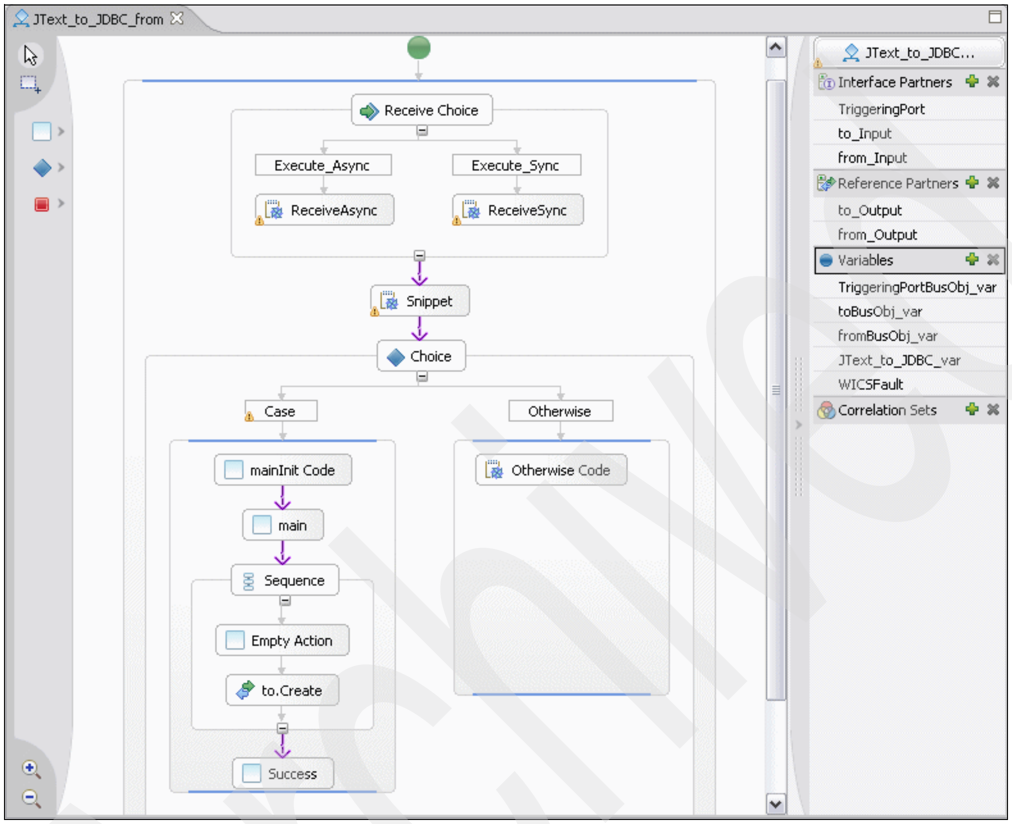


Figure 5-8   Process editor

Interface editor

Use the Interface editor (Figure 5-9 on page 65) to build Web Services Description Language (WSDL) Port Type interfaces that define some SCA components. Use this editor to add and to remove operations and to specify an operation’s inputs and outputs.

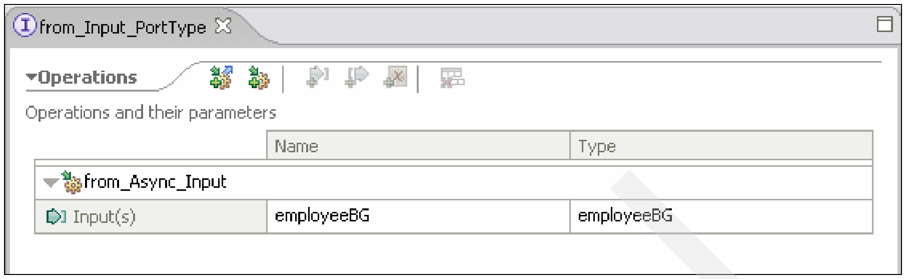


Figure 5-9   Interface editor

Interface mapping editor

Using the Interface mapping editor (Figure 5-10) you can build and edit interface maps through a graphical interface. The Interface mapping editor provides the methodology to deal with the differences between interfaces and to reconcile them for your integrated development environment.

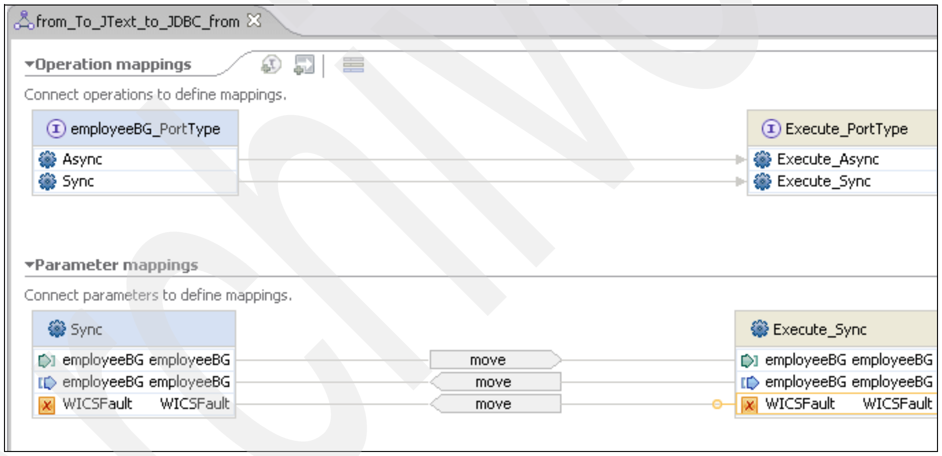


Figure 5-10   Interface mapping editor

Business object mapping editor

The business object mapping editor (Figure 5-11 on page 66) is a graphical interface that enables building and editing business object maps and their attributes. Using this editor you can define mapping of data between business objects.

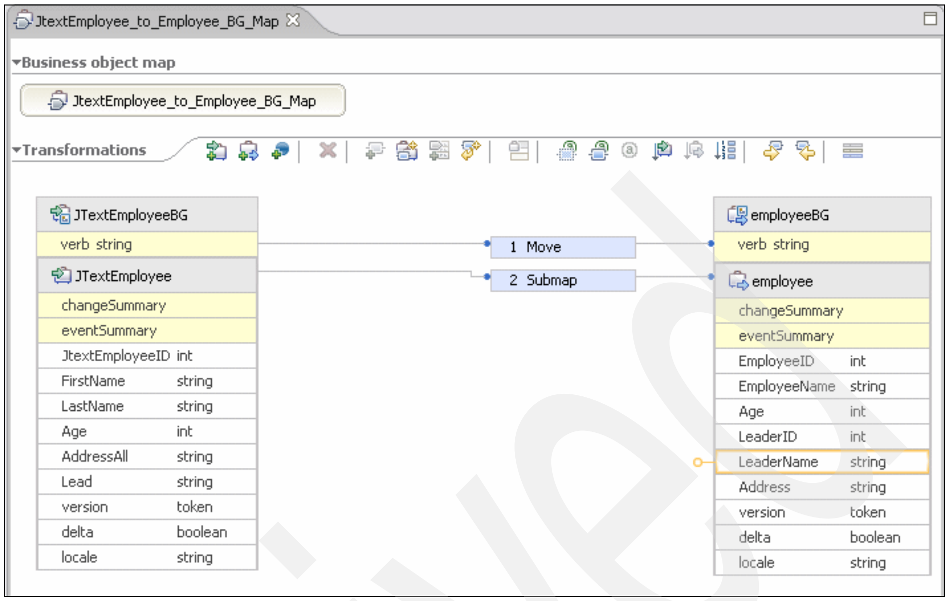


Figure 5-11 Business object mapping editor

Relationship editor

The relationship editor (Figure 5-12) in WebSphere Integration Developer enables you to build and edit relationships and their attributes through a graphical interface.

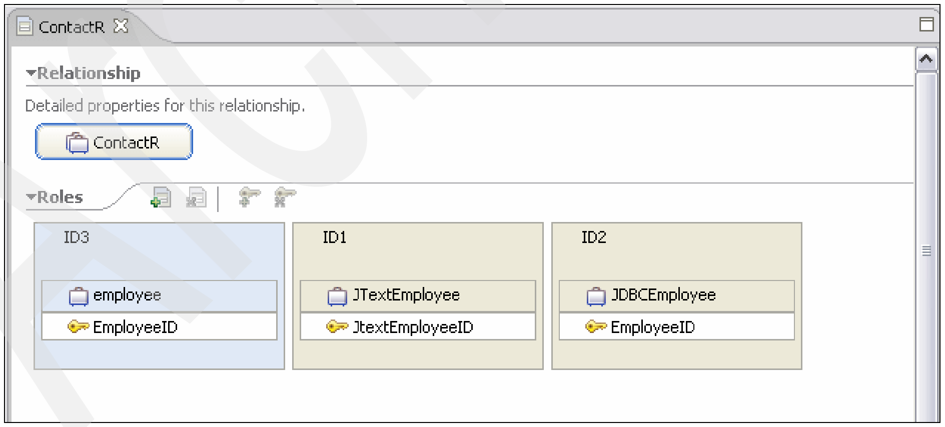
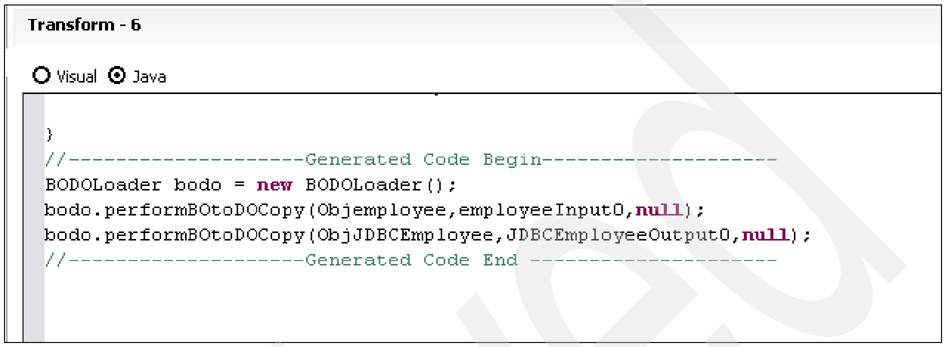


Figure 5-12 Relationship editor

Visual snippet editor

The visual snippet editor is a programming environment that you can use to graphically create and manipulate Java code. This editor provides two options to develop snippet code: Visual and Java. Figure 5-13 shows Java code in the snippet editor.



```
}
//---------------------Generated Code Begin---------------------
BODOLoader bodo = new BODOLoader();
bodo.performBOtoDOCopy(Objemployee,employeeInput0,null);
bodo.performBOtoDOCopy(ObjJDBCEmployee,JDBCEmployeeOutput0,null);
//---------------------Generated Code End ---------------------
```

Figure 5-13 Visual snippet editor

Module

Module is a Business Integration project for developing Service Component Architecture-based applications. It is the basic unit of deployment to WebSphere Process Server. A module is packaged in an EAR file and contains the following artifacts:

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

You can add dependent libraries, Java projects, and J2EE projects to a module and choose to deploy them with the module.

Shared library

Shared library is a Business Integration project for storing artifacts that are shared between multiple modules. For a module to use the resources from a library, it has to be added as a dependent to the module that is using the dependency editor as shown in Figure 5-14 on page 68.

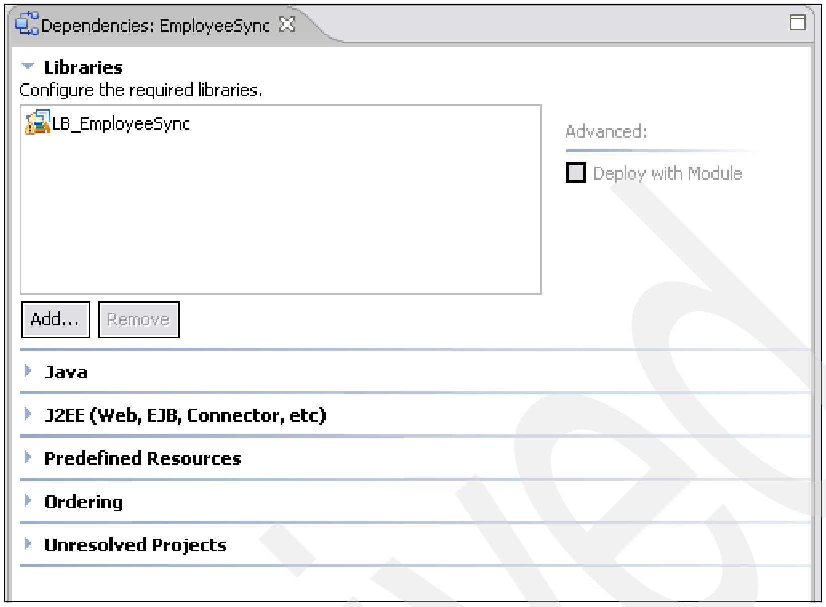


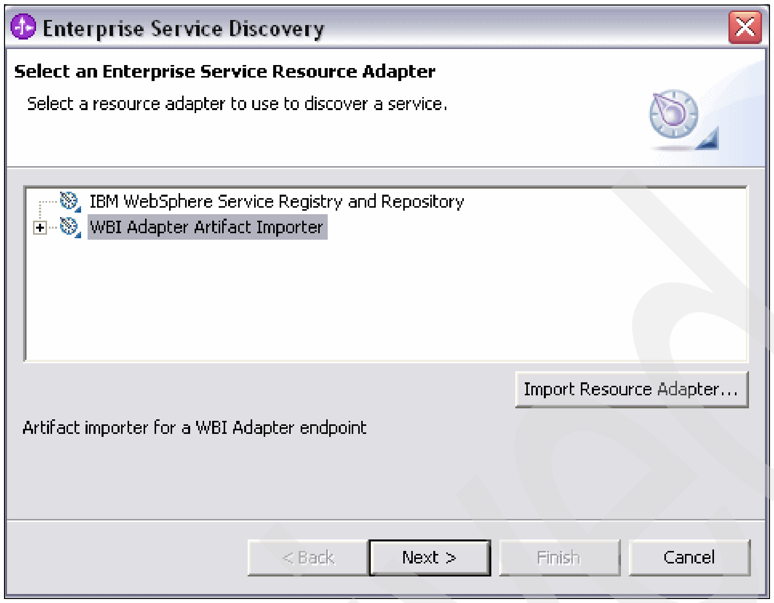
Figure 5-14 Dependency editor

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. At run time, libraries are not shared but are deployed with the module that depends on it. Also, you can add library dependencies to a library.

Enterprise service discovery wizard

Use the enterprise service discovery wizard to create Enterprise Information System (EIS) import and export components. EIS imports and exports are the means of creating services that use resource adapters to access EIS systems. Integrating applications on Enterprise Information Systems with those that are developed locally is a key feature of WebSphere Integration Developer.

Figure 5-15 on page 69 shows the ESD wizard.


Enterprise Service Discovery

Select an Enterprise Service Resource Adapter

Select a resource adapter to use to discover a service.

IBM WebSphere Service Registry and Repository
WBI Adapter Artifact Importer

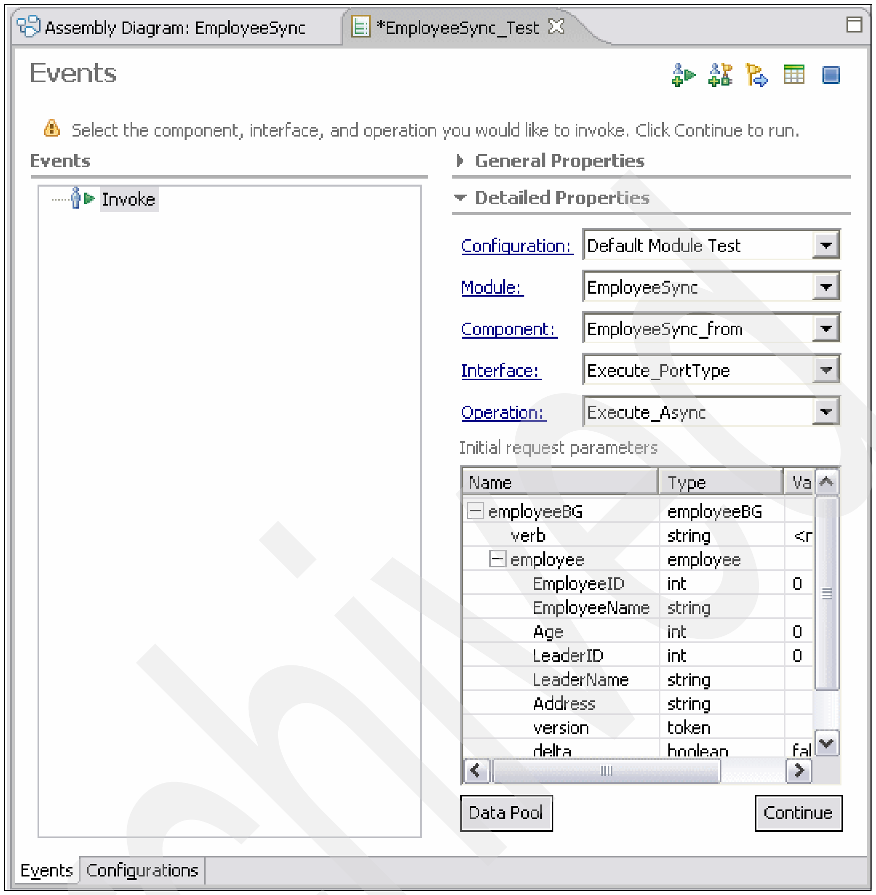
Artifact importer for a WBI Adapter endpoint

Figure 5-15 Enterprise service discovery wizard

Integration test client

The integration test client in WebSphere Integration Developer is the designated tool for testing modules and components. The test client features a sophisticated user interface to easily manage and precisely control your tests.

Figure 5-16 on page 70 shows the Integration test client.


Assembly Diagram: EmployeeSync | *EmployeeSync_Test

Events

Select the component, interface, and operation you would like to invoke. Click Continue to run.

Events: Invoke

General Properties

Detailed Properties

Configuration: Default Module Test

Module: EmployeeSync

Component: EmployeeSync_from

Interface: Execute_PortType

Operation: Execute_Async

Initial request parameters

Name	Type	Va
employeeBG	employeeBG	
verb	string	<r
employee	employee	
EmployeeID	int	0
EmployeeName	string	
Age	int	0
LeaderID	int	0
LeaderName	string	
Address	string	
version	token	
delta	boolean	fal

Figure 5-16 Integration test client

Migration wizard

Use the Migration wizard tool to migrate WebSphere Business Integration Server Foundation service project source artifacts to WebSphere Integration Developer.

Figure 5-17 on page 71 shows the most important steps in the execution of this Migration wizard.

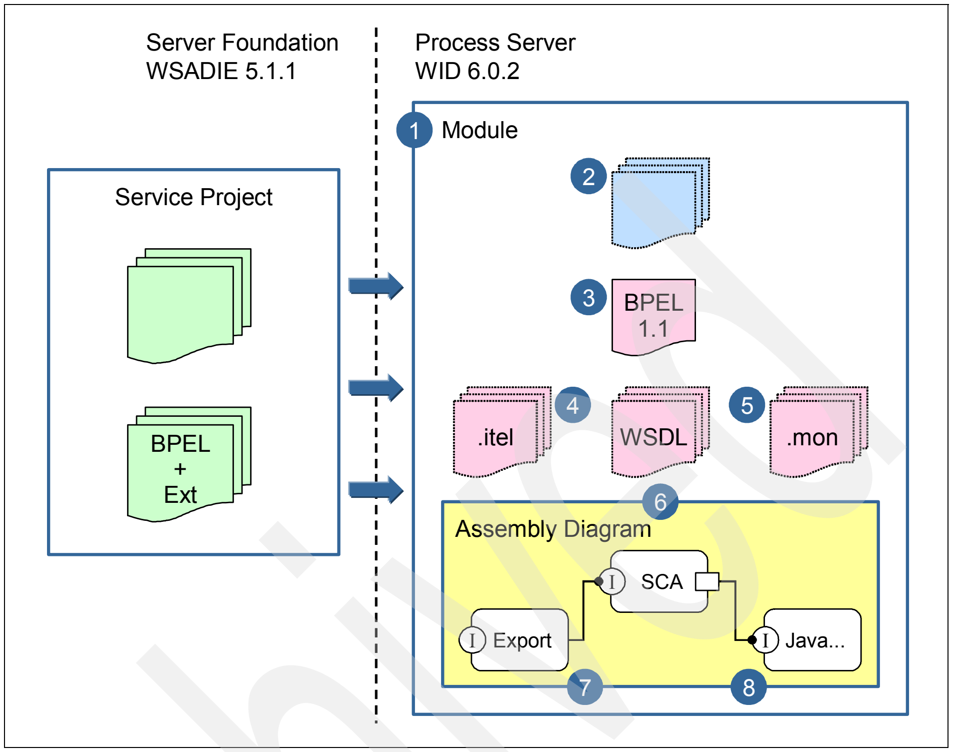


Figure 5-17 Artifact Migration wizard

The following numbered list corresponds with the numbers in Figure 5-17 to provide further explain what the Migration wizard does. The Migration wizard:

1. Creates a new business integration module and defines the classpath according to the original classpath.
2. Copies the source artifacts to the new module.
3. 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 upcoming WS-BPEL Version 2.0 specification.
4. Defines the Human Task activities using the new definition of .itel files. Cleans the BPEL file of the <wpc:staff> tags.
5. Generates a monitoring .mon file for each BPEL process to preserve the default monitoring behavior from WebSphere Studio Application Developer Integration Edition.
6. Creates an SCA component for each .bpel process.

7. Creates imports and exports depending on the deploy options chosen in WebSphere Studio Application Developer Integration Edition.
8. Wires the BPEL component to its partnerlinks (imports, exports, and Java components).

There is more information about how to use the Migration wizard in 6.4.2, “Using the Migration wizard” on page 101.

5.2.2 WebSphere Business Modeler

WebSphere Business Modeler streamlines the process of creating realistic business models by allowing you to integrate the different aspects of a complex business process in one model. WebSphere Business Modeler provides sophisticated analysis tools that let you evaluate both current and potential business processes. It also provides a direct link from business process modeling to the implementation of software services. Some functionalities of this tool are:

- 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 get a detailed, complete understanding of your process flow.
- Publish process models so that reviewers can view and comment on the models 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 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 of 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 using IBM WebSphere Business Monitor monitoring results.
- Export your process models in formats that other applications can use, such as BPEL, WSDL, XSD, FDL, and UML.

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 using the migration tool that is included in WebSphere Business Modeler V6. As both modelers are using an XMI repository, the migration is a mapping from the Version 5.1.1 XMI repository to the version 6.0.2.

To perform the migration:

1. Make a backup copy of all projects in the WebSphere Business Integration Modeler.
2. Install the last 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 some warnings are normal in the BPEL mode, but solve any error in the error view.

   The error list is an XMI file that is located 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 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 Modeler V6 project to WebSphere Integration Developer.

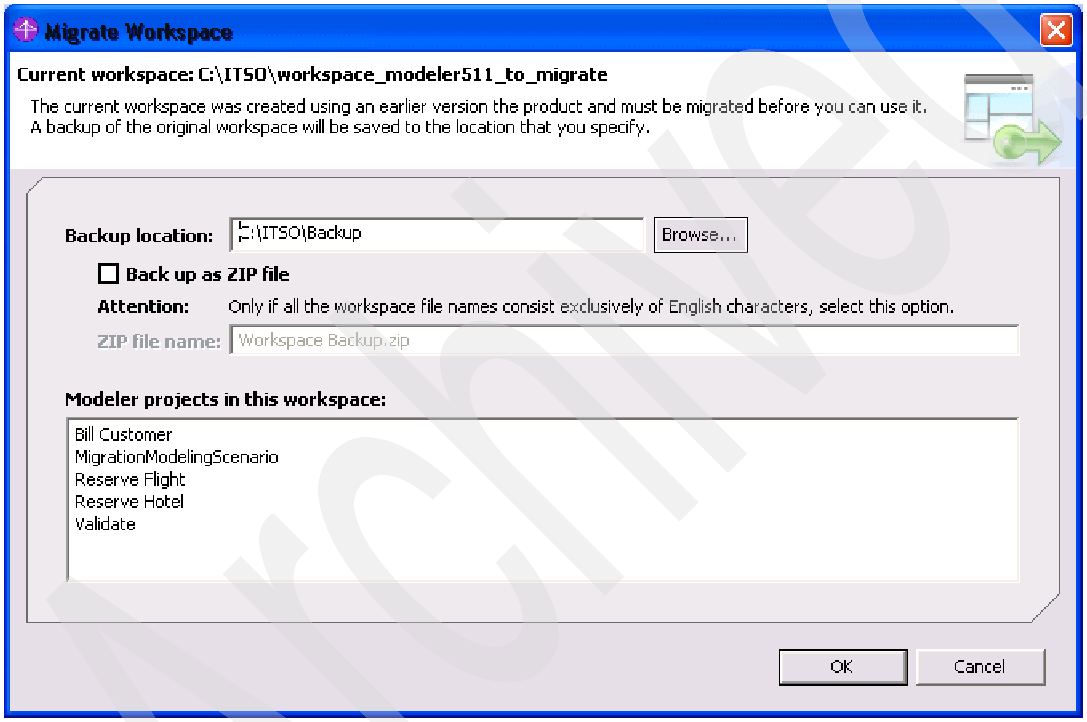
Migrating a Modeler V5 workspace

To migrate a Modeler V5 workspace:

1. Start the product WebSphere Business Modeler V6.
2. Select the location of the WebSphere Business Integration Modeler V5 workspace as the workspace. WebSphere Business Modeler opens the Migrate Workspace window.

Remember: If you selected 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: **Window** → **Preferences** → **Workbench** → **Startup and Shutdown** → **Prompt for workspace on startup**.

3. Define a location path for the workspace backup, as shown in Figure 5-18. It is optional to specify to make the backup as a zipped file. In this case, ensure that all of the files and directories are using only English characters.



Current workspace: C:\ITSO\workspace_modeler511_to_migrate

The current workspace was created using an earlier version the product and must be migrated before you can use it. A backup of the original workspace will be saved to the location that you specify.

Backup location: C:\ITSO\Backup

☐ Back up as ZIP file

Attention: Only if all the workspace file names consist exclusively of English characters, select this option.

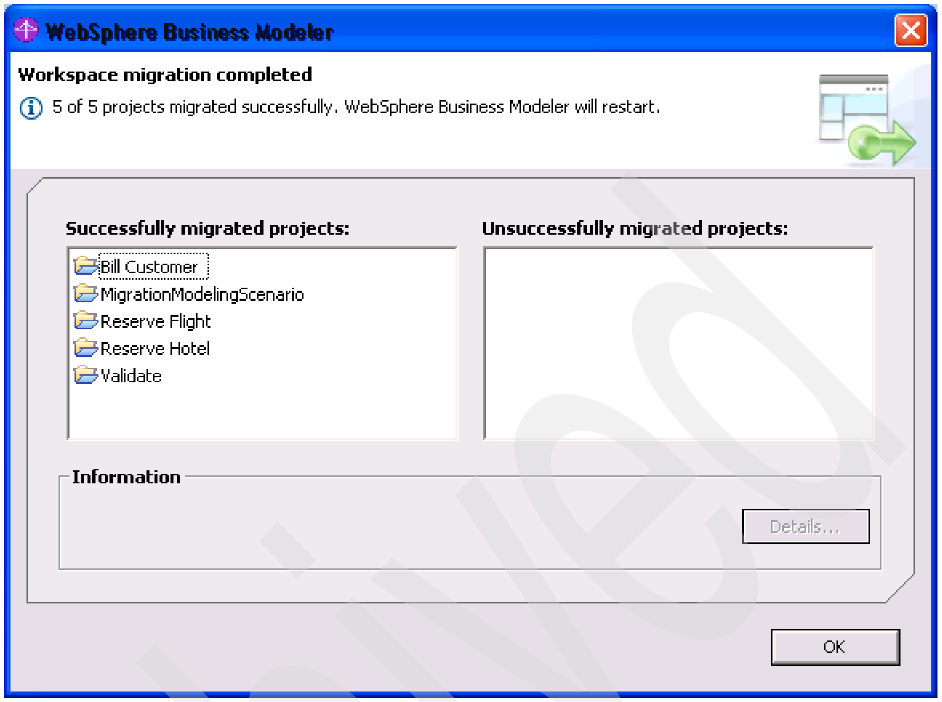
ZIP file name: Workspace Backup.zip

Modeler projects in this workspace:

Bill Customer
MigrationModelingScenario
Reserve Flight
Reserve Hotel
Validate

Figure 5-18 Backup location definition in Migrate Workspace window

The Migration Workspace window shows a summary, with a list of the projects migrated successfully and which did not migrate, as shown in Figure 5-19 on page 75.


WebSphere Business Modeler

Workspace migration completed

5 of 5 projects migrated successfully. WebSphere Business Modeler will restart.

Successfully migrated projects:

- Bill Customer
- MigrationModelingScenario
- Reserve Flight
- Reserve Hotel
- Validate

Unsuccessfully migrated projects:

Information

Figure 5-19 Migration completed

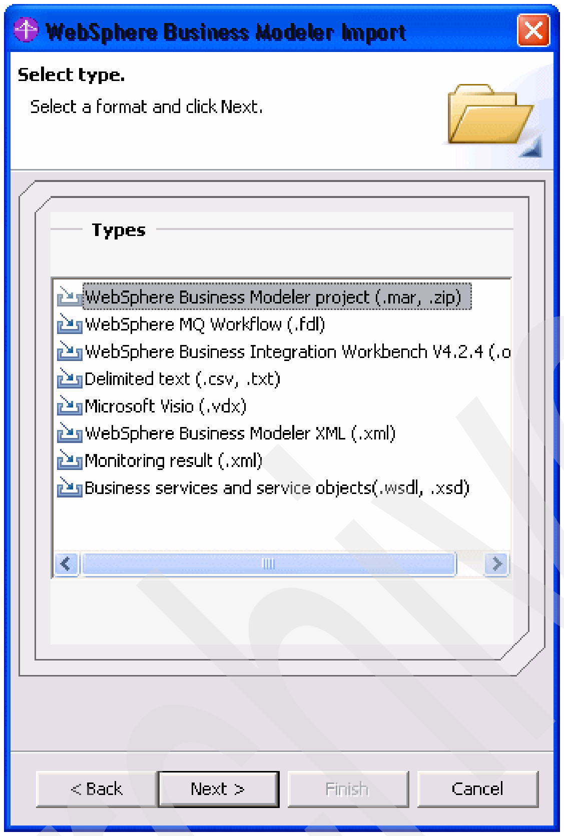
4. Review and correct eventual problems in the projects. You might need to manually adjust some configuration artifacts to get them ready to export to WebSphere Integration Developer.

Migrating Modeler V5 exported projects

To migrate Modeler V5 exported projects:

1. Export the model using WebSphere Business Integration Modeler as a **WebSphere Business Integration Modeler V5 Project**. It generates a zip file.
2. Open the importing wizard by selecting **File** → **Import** → **WebSphere Business Modeler Import.**

3. Select the format: **WebSphere Business Modeler Project**, as shown in Figure 5-20.


WebSphere Business Modeler Import
Select type.
Select a format and click Next.
Types
< Back
Next >
Finish
Cancel

WebSphere Business Modeler project (.mar, .zip)
WebSphere MQ Workflow (.fdl)
WebSphere Business Integration Workbench V4.2.4 (.o
Delimited text (.csv, .txt)
Microsoft Visio (.vdx)
WebSphere Business Modeler XML (.xml)
Monitoring result (.xml)
Business services and service objects(.wsdl, .xsd)

Figure 5-20 Modeler Import: Format selection

4. Browse for the folder that contains the zipped file. Select the project file. Choose an empty target project, or create a new Project, as shown in Figure 5-21 on page 77.

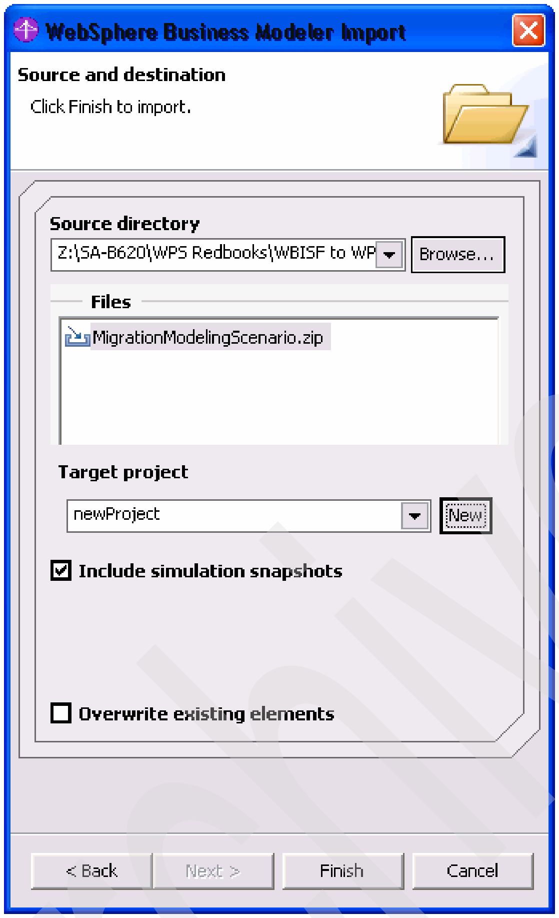

WebSphere Business Modeler Import
Source and destination
Click Finish to import.
Source directory
Z:\SA-B620\WPS Redbooks\WBISF to WP
Browse...
Files
MigrationModelingScenario.zip
Target project
newProject
New
Include simulation snapshots
Overwrite existing elements
< Back
Next >
Finish
Cancel

Figure 5-21 Modeler import: Project selection

5. Click **Finish**. The importing wizard recognizes that there is a migration from an earlier version of WebSphere Business Modeler and shows a version detection message.
6. When the importing process is completed, the wizard window shows the finished message. Click **Details** if there are any warnings or errors**.**

Migrating Modeler V5 projects using CVS

To migrate Modeler V5 projects using CVS:

1. Open the project tree view context menu (right-click). Select **Team** → **Check Out Project**.

2. Browse the project tree, and select the WebSphere Business Integration Modeler V5 project. Click **Finish.**

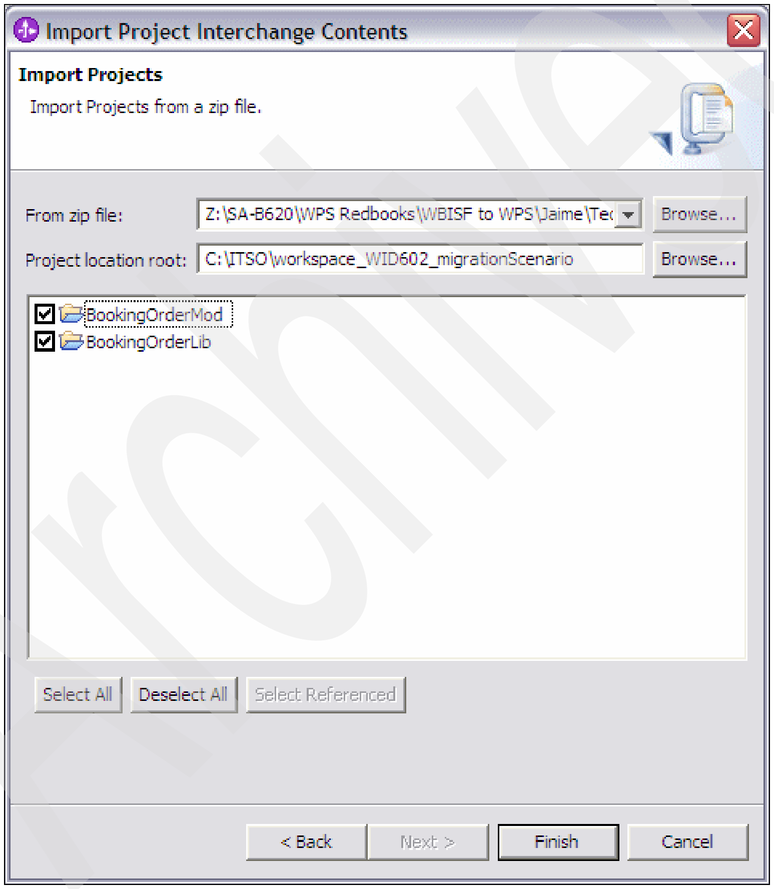
   WebSphere Business Modeler V6 makes a copy of the original project from the CVS repository and migrates that copy.

3. Review the generated project, and correct any errors that are displayed.

Exporting Modeler V6 projects to Integration Developer

Export the model using a Project Interchange file.

1. Open the WebSphere Integration Developer import wizard (Figure 5-22), and import the Modeler V6 project.



Import Project Interchange Contents

Import Projects

Import Projects from a zip file.

From zip file: Z:\SA-B620\WPS Redbooks\WBISF to WPS\Jaime\Tec

Project location root: C:\ITSO\workspace_WID602_migrationScenario

☑ BookingOrderMod
☑ BookingOrderLib

Figure 5-22 Importing project from Modeler V6

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

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

5.3 Migration approaches

A business solution that is running on WebSphere Business Integration Server Foundation usually consists of multiple applications. There might be dependencies between these applications that you must consider before you migrate a single application. See 5.3.1, “Migrating with dependencies between applications” on page 79 for more details about evaluating different migration approaches for different scenarios with different dependencies between applications.

If long running processes are used in Server Foundation applications, additional considerations must be applied. There are three approaches for handling migration of 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, a thorough assessment of the existing long running process instance environment is required. Based on that assessment, as we describe in the next section, consider one approach or a mix of several approaches.

5.3.1 Migrating with dependencies between applications

Depending on the complexity of the application to be migrated, additional considerations might be needed. When you need to migrate multiple applications, determine if these applications are inter-dependent, and if they are inter-dependent, then to what extent. The following situations might be present:

- Applications are independent of each other.

  This is the easiest case for a migration. No special considerations apply here.

- Applications are not interacting with each other but interacting 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 datastore like the remaining applications on the Server Foundation system.

- Applications invoke one another, with no circular dependencies, as shown in Figure 5-23.

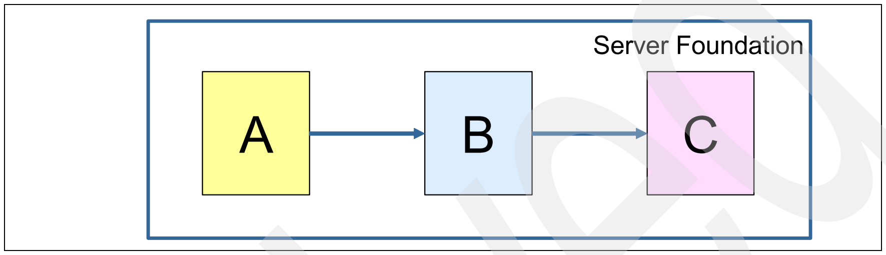


Figure 5-23 Applications invoke one another, with no circular dependencies

In this case, the common approach is to migrate one application after the other. The 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, leaving B and C on the old system until they are ready for migration (1). 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 Server Foundation. Finally, change B' from call C (3) to C' (4). See Figure 5-24 on page 81 for an illustration of this migration method. The numbers in Figure 5-24 on page 81 correspond with the numbers in this explanation.

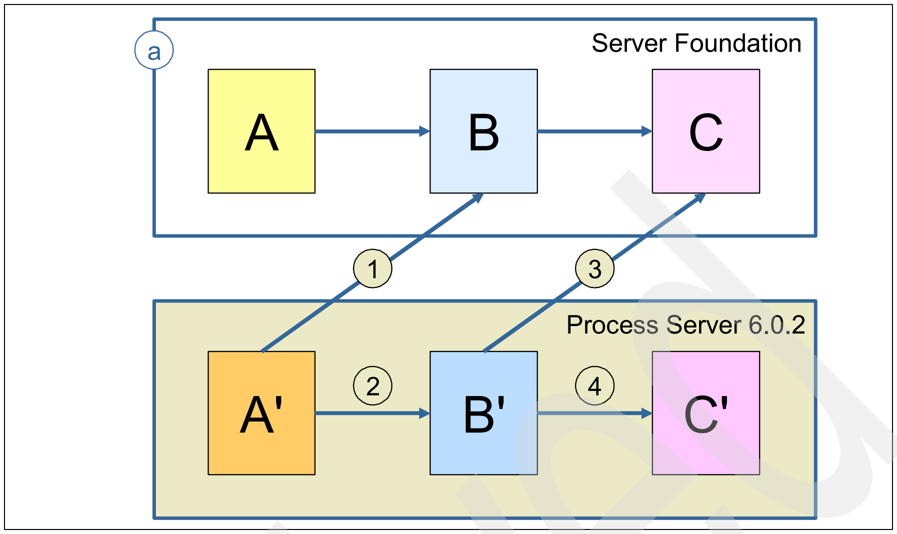


Figure 5-24 Migration starting first last

– Start migrating application C. When ready, change B to call C’ (1), followed by B, and change the invocation in A to B’ (2) and finally change the client to call A’ (3), as shown in Figure 5-25. In difference to the first approach, this approach requires changes to the Server Foundation applications to direct their calls to the migrated applications in the new environment.

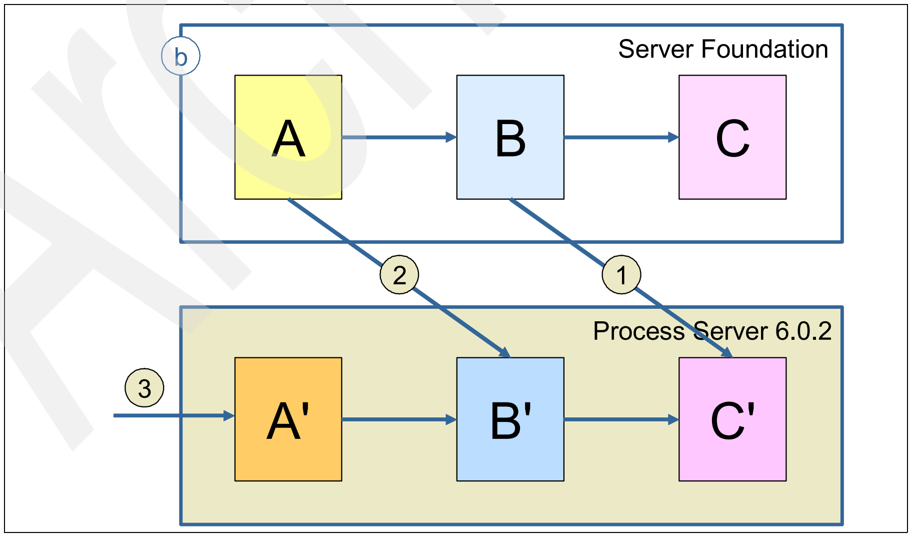


Figure 5-25 Migration starting last first

We recommend the first approach because in this approach, WebSphere Process Server does the migration work by directing the calls to the other environment. Because this is the target platform and is used in future development, it is better to spend time and resources using this product whenever possible, instead of developing on the old Server Foundation platform, which is necessary in the second approach.

If long running processes are calling one another, we recommend a migration approach that starts at the last called process until the first process in the call chain. This recommendation is opposite to 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 Server Foundation and Process Server, although it requires changes to the applications on Server Foundation so that the newly migrated processes are invoked.

- Circular dependences 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 migration approach:

  - Migrating individual applications and modifying applications that are not yet migrated to invoke the newly migrated application. You must make additional changes to the migrated applications as you keep moving applications to the Process Server. This approach actually combines the two approaches that we described earlier.
  - Migrating all applications at once.

    There are disadvantages for both approaches. The first approach implies a large maintenance and planning effort. Also, development on both platforms is required. The second approach might be risky because you are moving all functions of the business application at once. 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.

5.3.2 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 Process Server because the database schema changed between the Server Foundation and the 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 Server Foundation to Process Server; hence, take special actions for these processes. These specialities, with regards to long running processes, must be considered already during migration planning.

Assess process instances in Server Foundation environment

To choose the best migration approach for long running process instances in the Server Foundation environment, determine their average and maximum duration, which helps to get an overview of the environment to be migrated.

If available, capture the following numbers 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
- Number of active running process instances
- Average lifetime of an active process instance
- Age of oldest active process instance

There are various ways to determine the duration of processes instances:

1. The business operations knows what duration to expect for a process instance of a certain type. There also might be custom logging of values that are usable for determining the duration of a process in the application.
2. A technical evaluation of the data that 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 check the **Automatically delete process after completion** option, data can be obtained for finished processes from the BPEDB.

   If not, than 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 give a good advice. Repeat that query for start time of running instances in reasonable time frames to get an impression on how many process instances are completing in these time frames and to get confidence that the obtained numbers are applicable.

   There is also the AuditLog view within the BPEDB, which is a good source for 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. More information about using the AuditLog is in the following article:

`http://www-128.ibm.com/developerworks/websphere/library/techarticles/0503_neumann/0503_neumann.html`

We recommend that you use both approaches, the business and the technical one, that we previously described, which helps to get a complete picture on the old environment, especially if there are discrepancies observed between the results of the two approaches.

Having obtained these numbers, you know:

- How long the process instances typically take until they are completed.
  - Determines how much time is needed to finish the running process instances on the Server Foundation environment.
- What a reasonable time is, after which you can terminate a process instance.
  - It might be the case that some processes even take longer than the expected maximum time. Create a plan for what to do in that case as well, for example, take 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. So, it might be possible to terminate these process instances.

There are cases where processing a business operation requires multiple processes. Say, the business operation for loan approval consists out 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 needs to be calculated as well, 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 5-26 on page 85 shows the process durations.

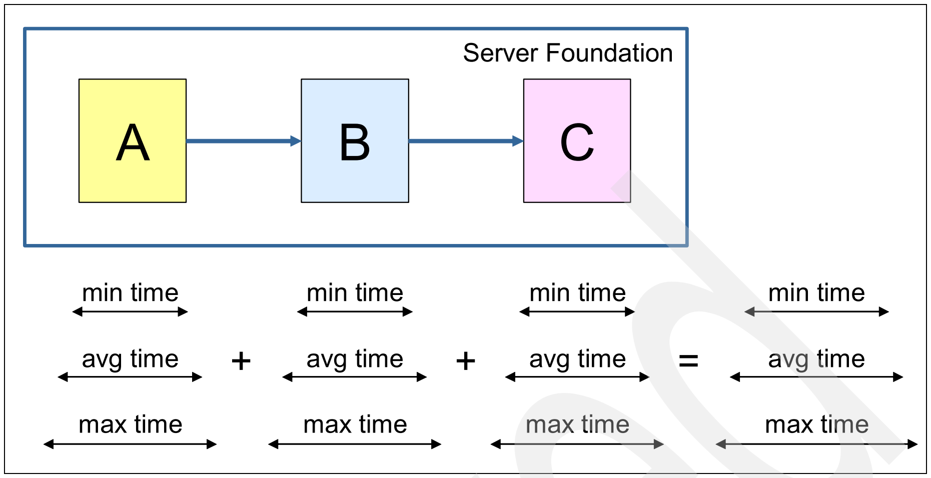


Figure 5-26 Process durations

Coexistence approach

Coexistence is the means of running both environments, Server Foundation and 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.

Figure 5-27 on page 86 illustrates the coexistence approach.

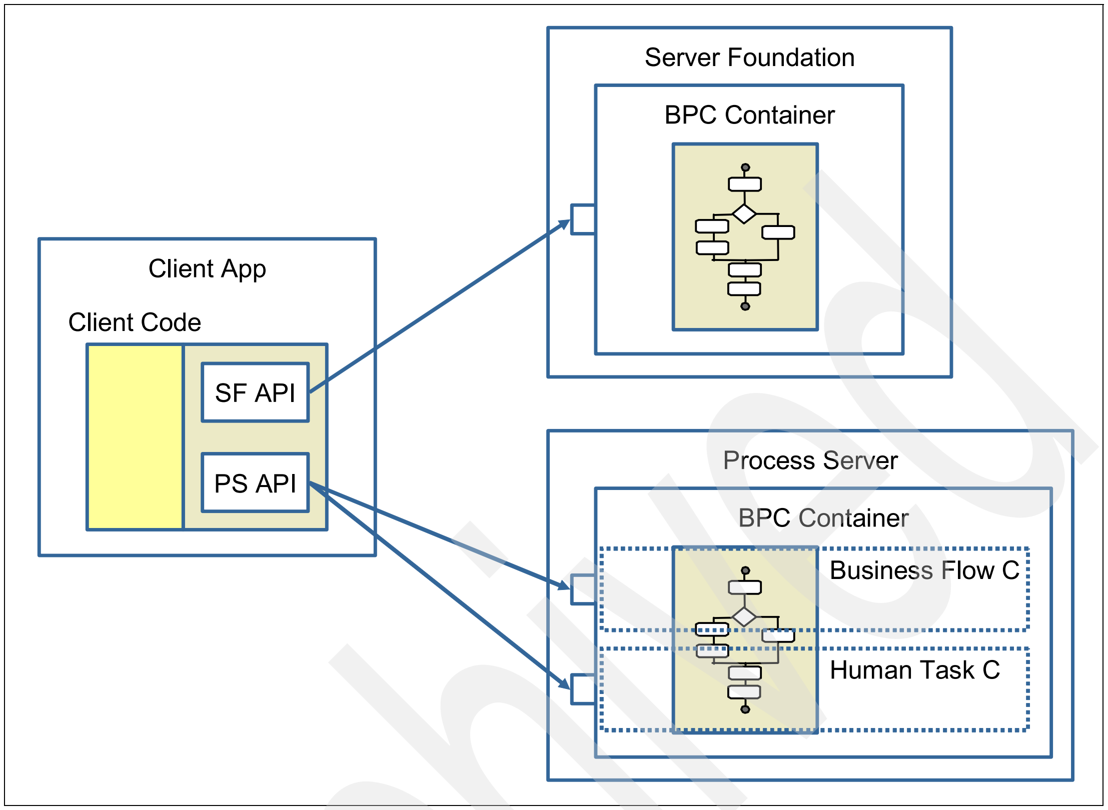


Figure 5-27 Coexistence approach

Using the coexistence approach you can build up the Process Server system, deploy and test your migrated application, test a new version of the client that talks to both systems, and, in some point-in-time, go live with both systems by changing the client to direct new process starts to the Process Server while still handling requests for the Server Foundation process instances.

The disadvantage of this approach is that there is probably the need for additional hardware 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 Server Foundation processes take until completion, which we described in section“Assess process instances in Server Foundation environment” on page 83. This defines the necessary time of running both environments in parallel. The time for which these two systems and the extra effort must be spent should be sized and part of the migration planning to make sure that the hardware and human resources are available during this phase of coexistence.

If there are process instances of several templates to be migrated, say having a loan approval process and a travel booking process that are running on the same Server Foundation environment, you must determine the order of migration. Migrate the process instances with the longest duration first, and processes with the short duration last, which minimizes the period of coexistence of Server Foundation and Process Server.

In case one business operation consists of several process instances, decide whether to migrate all of the related processes at once to the Process Server or whether to migrate them one after another.

If you migrate all of the processes at once, 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 could 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 5-28 illustrates the scenario to migrate the longest processes last.

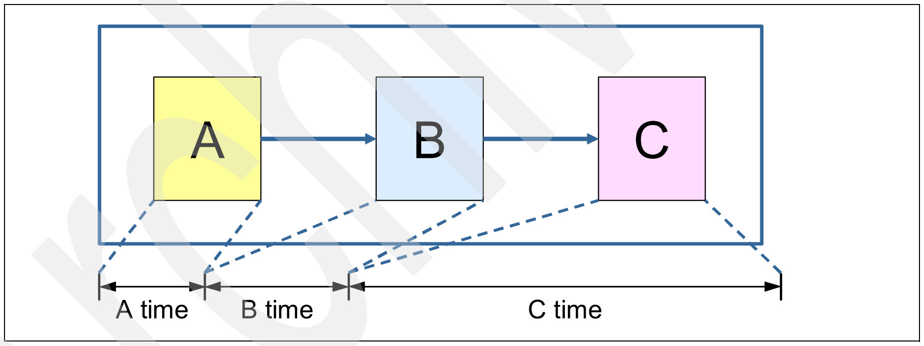


Figure 5-28 Scenario: Migrate the longest last

This approach migrates the last process in the call chain to the Process Server, followed by the preceding one, until the first process in the call chain is migrated to the Process Server. The disadvantage of this approach is that it requires you to do modifications in the remaining Server Foundation processes in the old environment to call the processes that are already migrated to the Process Server.

Adopt client applications

When the Server Foundation and Process Server environments are running in parallel, ensure that new processes are kicked off only at the new system, while users must be able to work on staff activities or human tasks on both systems.

So, the client applications that are interacting with the Server Foundation must be enhanced to cope with both systems for the time of coexistence. The Java APIs for client interaction changed. Common clients in Server Foundation use the BPE Generic API, the Process SessionBean Facade, or the BPE JMS API. As of these APIs have changed. Change the client as well to interact with the Process Server system. Details on migrating the client are in section, 6.10, “Migrating clients” on page 125.

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 needs to determine where to direct the requests on a per request basis.

Always direct a process start request for a migrated process instance to the Process Server.

You might need to send requests to running process instances to the Process Server first. If the API response indicates that the process instance was not found, the client needs to try Server Foundation. There might be cases where the client can decide upon the request message contents to which system the request should 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 just needs to know when the first order was placed on the Process Server system. It can than determine, based on the order number, which system to call. If the order number is higher than the first order in the Process Server, then call the Process Server, else call Server Foundation.

Always direct queries for human tasks to both systems until all tasks in Server Foundation are completed and it is sure that no new tasks are created during the processing of the remaining process instances. The results need to be incorporated by the client to one list to be displayed to the user.

If the period of coexistence is short, it might be affordable by end users to use two client applications instead of spending development efforts to get a common view on the Process Server and Server Foundation environments, which saves development and test time. The Server Foundation client can be phased out together with Server Foundation at the end of the period of coexistence. A decision for this approach highly demands the client functionality and the kind of end users it is exposed to.

Move at once approach

Another option of handling long running process instances is to move them at a certain point-in-time by stopping the work of the instances in the Server Foundation environment and starting respective instances in the new Process Server environment.

The advantage of this approach is that there is no additional hardware, administrative, and client development and maintenance effort 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 needs to accomplish the following steps for each process instance to migrate:

- Suspend the instance so that it stays in the same state during transition. This helps to ensure that process instances do not get finished during the time of transition and therefore might be run twice (once in the old system and once in the new environment).
- Get the input messages for the process instance and transform them into Process Server-style input messages (SDO), if applicable.
- Initiate a new process instance on the Process Server system.
- Terminate the process instance on the Server Foundation environment.

This approach is applicable to very 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. Backend systems that participate in the process need to be updated during this transition operation as well. These disadvantages might hinder going for this migration approach. Process instances for which this approach might be applicable are:

- Processes that require a short period of time until completion. Resubmitting this process instance in the new environment creates a backlog of work of at maximum this time on the new system and for the end users.
- Number of running process instances is small and enables to handling the processes manually after resubmitting on Process Server. An administrator can manually complete the human tasks that were already completed in the Server Foundation process to get the new Process Server process instance in the same state as the Server Foundation process instance was before suspending.
- The processes are structured in a way that enables them so that resubmitting does not cause conflicts. An example for such a process is a process that requires one ore multiple approval steps before executing work. The work then is all automatic, fast completing 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. So, when you suspend them and resubmit them on the new system there might be, at most, some staff activities to repeat.

Before you choose the 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 should not be repeated, in the new environment. So, the move to the at once approach is not acceptable. You want these process instances in the new environment to start off at a state in which the corresponding instance in the old environment has stopped.

Accomplishing a solution to hook up in the new process instance in the Process Server at the point that the old process that are stopped is a difficult procedure. There is additional logic in the new required process to make sure that certain activities are skipped that were done already in the Server Foundation process instance. Taking the complexity of the processes into account, it might not be reasonable to spend the effort in development of such a move over solution.

Migrating build time artifacts

In this chapter, we discuss:

- Introduction to migration of build time artifacts
- General considerations
- Preparing source artifacts for migration
- Migrating Service projects using the Migration wizard
- Manually completing the migration
- Migrating BPEL processes
- Migrating Java services
- Migrating EJB services
- Migrating Web services
- Migrating clients
- Migrating WSIFMessage API calls
- Migrating Business Rule Beans applications
- Migrating CMP/A applications
- Migrating Extended Messaging applications
- Migrating CEI applications
- Migrating WebSphere Business Integration Adapters

6.1 Introduction to migration of build time artifacts

In this chapter, we cover aspects and considerations for the migration of build time artifacts from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer.

More detailed information about this topic is in the Info Center in the following Web locations:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.602.mig.doc/welcome_wps_mig.html

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tsrcartifacts.html

In this chapter, we first focus on the capabilities that are provided through the Migration wizard, which is a tool within Integration Developer. This wizard automatically migrates Server Foundation service projects to Integration Developer modules. See the following topics for information about this:

- Preparing source artifacts for migration
- Migrating Service projects using the Migration wizard
- Manually completing the migration

The remaining sections discuss the considerations and recommendations that apply to specific Server Foundation artifacts, components and services:

- BPEL processes
- Java services
- EJB services
- Web services
- Clients
- WSIFMessage API applications
- Business Rule Beans
- Extended Messaging (CMM) applications
- Container Managed Persistence over Anything (CMP/A) applications
- Common Event Infrastructure (CEI) applications
- Adapter usage

6.2 General considerations

WebSphere Process Server inherits new and enhanced functionality compared to Server Foundation, which includes:

- Enhanced support of the upcoming WS-BPEL version 2.0 specification. The business process engine that is within Process Server is built on BPEL4WS Version 1.1 with major capabilities of the upcoming WS-BPEL Version 2.0 specification. These 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 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 Server Foundation application that is to be migrated.

> **Note:** As part of a migration project, always consider adapting a migrated Server Foundation application to use new Process Server functionality.

Before you start the migration, review the general limitations of the migration process, which are located in the WebSphere Integration Developer Info Center:

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/rlimsrcart.html`

6.2.1 General migration tasks for source artifacts

To migrate projects from WebSphere Studio Application Developer Integration Edition to WebSphere Integration Developer:

1. Prepare the source artifacts for migration. You might need to perform these actions in WebSphere Studio Application Developer Integration Edition.
2. Use the Migration wizard to automatically migrate the artifacts to the Business Integration Module project.
3. Use WebSphere Integration Developer to manually complete the migration, which involves fixing any Java code that could not 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 Process Server features, if applicable.
5. Manually migrate or change the Server Foundation artifacts where necessary.

6.3 Preparing source artifacts for migration

The first step of the source artifacts migration process consists of preparing the Integration Edition workspaces for migration. In this section, we give an overview on the most important tasks during this step.

One important part of the considerations in this step must be the changes in the project organization between Integration Edition Service projects and Integration Developer Module projects. These changes usually require a change in the Integration Edition project structure before you migrate to Integration Developer.

Details about this migration step are on the following Info Center Web page:

```
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topi
c=/com.ibm.wbit.help.migration.ui.doc/topics/tsrcartifacts.html
```

6.3.1 Project organization changes in Integration Developer

One goal of the Service Component Architecture is to separate business integration logic from implementation logic. Therefore, WebSphere Integration Developer separates the assembly role from the implementation role.

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 defines 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, which is never exposed to the business layer.

Figure 6-1illustrates the SOA layers.

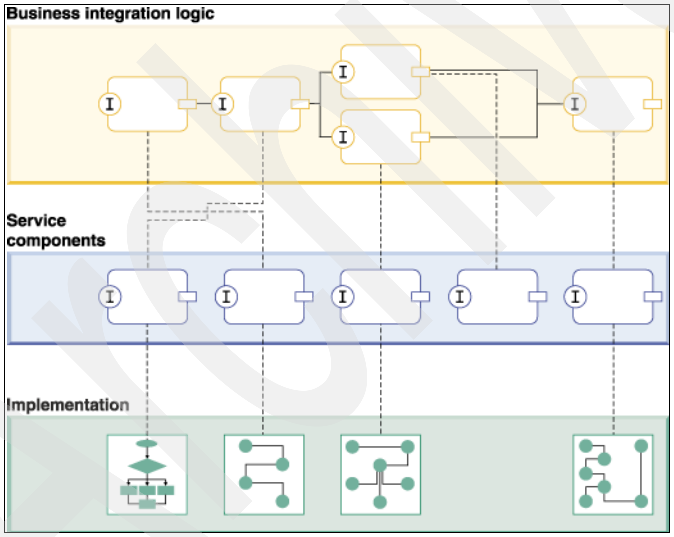


Figure 6-1 Service Component Architecture Layers

WebSphere Integration Developer offers a new way to organize SCA in separate layers. The new Assembly Editor allows you (the developer) to specify the SCA service component, how the component ties with the others, which external services it needs, and how it is functionality exposed. These concepts and functionalists differ significantly from the environment that is proposed by WebSphere Studio Application Developer Integration Edition.

In the Business Integration perspective, WebSphere Integration Developer organizes the source artifacts using two kinds of projects:

- The Business Integration Module project. This container organizes the source code using the distribution in Figure 6-2.

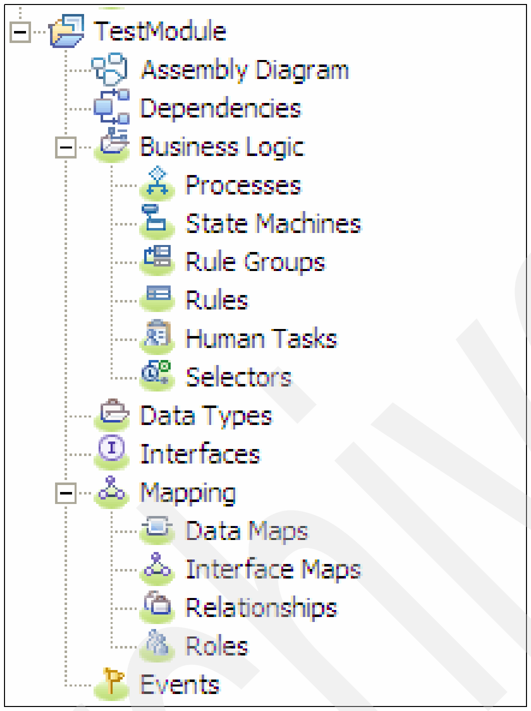


Figure 6-2 Module project

  This kind of project is some how equivalent to the Service project in WebSphere Studio Application Developer Integration Edition. Each module is published as an EAR file.

- The Business Integration Library project. This new concept allows you (the developer) to have a project with interfaces, data types, dependencies, mappings, and events without the business logic artifacts, which can be useful for sharing these definitions across different modules. Figure 6-3 on page 97 shows the Library project.

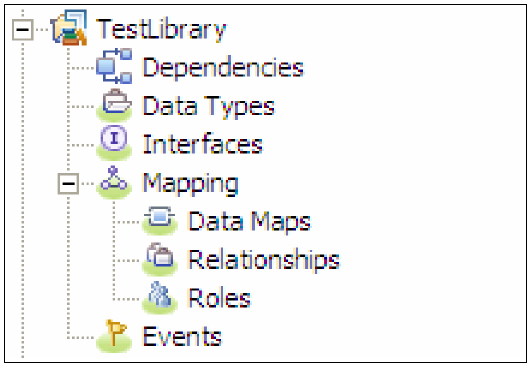


Figure 6-3 Library project

This new project organization that is within Integration Developer implies some considerations for migrating Service projects from WebSphere Studio Application Developer Integration Edition:

- Business Integration Module projects cannot share artifacts. Use the Business Integration Library projects to share artifacts.
- To share resources, such as .xsd or .wsdl files, reorganize the WebSphere Studio Application Developer Integration Edition artifacts.

  In Integration Edition Service projects, .wsdl and .xsd files from different service projects could be referenced. These dependencies should be resolved before you migrate to Integration Developer.
- The dependences of Service projects are important to the migration process. Migrate the Service projects in their dependency order by first migrating the one that is referenced by others projects.

However, designing the right granularity of WebSphere Process Server modules may 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.

6.3.2 Preparing the Integration Edition workspace

The following steps describe how to prepare the environment before you migrate the source artifacts to WebSphere Integration Developer from Integration Edition:

1. Backup the entire Integration Edition V5.1 workspace before you migrate.
2. 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:

– .wsdl and .xsd files are in the same Service project as the.bpel file. No further action is required.

– .wsdl and .xsd files are in a different Service projects.

  Prior to migration, reorganize the Integration Edition V5.1 artifacts using Integration Edition. The reason for this is that Business Integration Module projects might not share artifacts. Here are the two options for reorganizing the 5.1 artifacts:

  - Merge artifacts into one module.

    In Integration Edition, create a new Java project that holds all of he common artifacts. Place all .wsdl and .xsd files that are shared by more than one Service project into this new Java project. Add a dependency on this new Java project to all Service projects that use these common artifacts. In WebSphere Integration Developer, create a new Business Integration Library project with the same name as the Integration Edition V5.1 shared Java project before migrating any of the service projects. Manually copy the old .wsdl and .xsd files from the 5.1 shared Java project to this new Business Integration Library project folder. You must do this before you migrate the BPEL Service projects.

  - 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), create a Business Integration Library project with the same name as the Integration Edition V5.1 project, and also set up the new library project's classpath, and add the entries from the 5.1 Java project. This type of project is useful for storing shared artifacts.

3. Export the service project that should be migrated, by selecting **File** → **Export** and select **Project Interchange**.

   Figure 6-4 on page 99 shows the Export window.

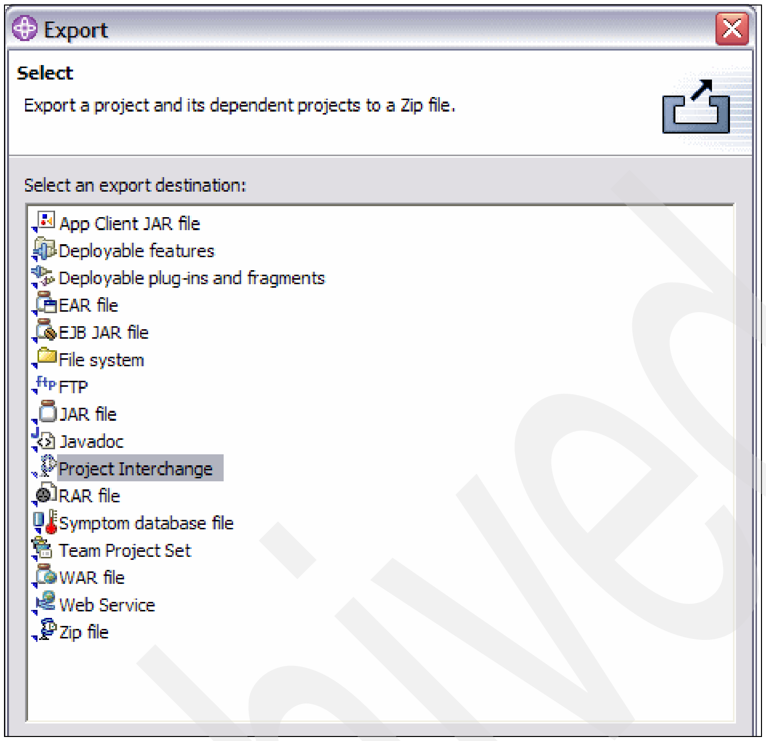


Figure 6-4 Export Project Interchange in Integration Edition V5.1

4. Unzip the exported **Project Interchange** to a temporary directory.

6.3.3 Preparing the Integration Developer workspace

The tasks in this step are detailed in the information center for WebSphere Integration Developer, and can be summarized as follows:

1. Use a new workspace.

Tip: Remember, if you selected the **Use this as the default and do not ask again** option with an old workspace, change the default value after you start the product by selecting **Window** → **Preferences** → **Workbench** → **Startup and Shutdown** → **Prompt for workspace on startup**.

2. Select **Window** → **Preferences** → **Workbench** → **Capabilities**, and review the environment that has the all capabilities enabled. The Info Center details exactly which capabilities WebSphere Integration Developer needs.

3. Review the migration section of the Rational Application Developer Information Center to determine the best way to migrate the non-WebSphere Business Integration specific projects in the workspace: Migrating from WebSphere Studio V5.1, 5.1.1, or 5.1.2.
4. Review the Web service section of the Rational Application Developer Information Center for additional background information about the Web service functionality that Rational Application Developer provides.
5. Import all of the non - Service projects from the old workspace to the new workspace by clicking **File** → **Import** → **Existing Project into Workspace**. Do not copy the deploy code that WebSphere Studio Application Developer Integration Edition generates.

   By default, WebSphere Integration Developer generates the deploy code during build time and not manually as in Integration Edition.

   Note: We do not recommend that you open WebSphere Integration Developer in an old WebSphere Studio Application Developer Integration Edition workspace that contains Service projects because you must first migrate those projects to a format that WebSphere Integration Developer can read.

6. Make sure that all projects build properly. If necessary, fix the classpath of the projects, which is described in the Info Center.

You are now ready to begin the migration process using the Integration Developer Migration wizard.

6.4 Migrating Service projects using the Migration wizard

You can migrate Integration Edition service projects with the help of a Migration wizard in WebSphere Integration Developer.

6.4.1 Overview of the Migration wizard

The Migration wizard does the following:

- Creates a new Business Integration module (the module name can be defined).
- Migrates the service project's classpath entries 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 WSDL files.
- 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.
- 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 deploy options that you choose in WebSphere Studio Application Developer Integration Edition.
- 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 Integration Edition service project. It does not migrate any application binaries.

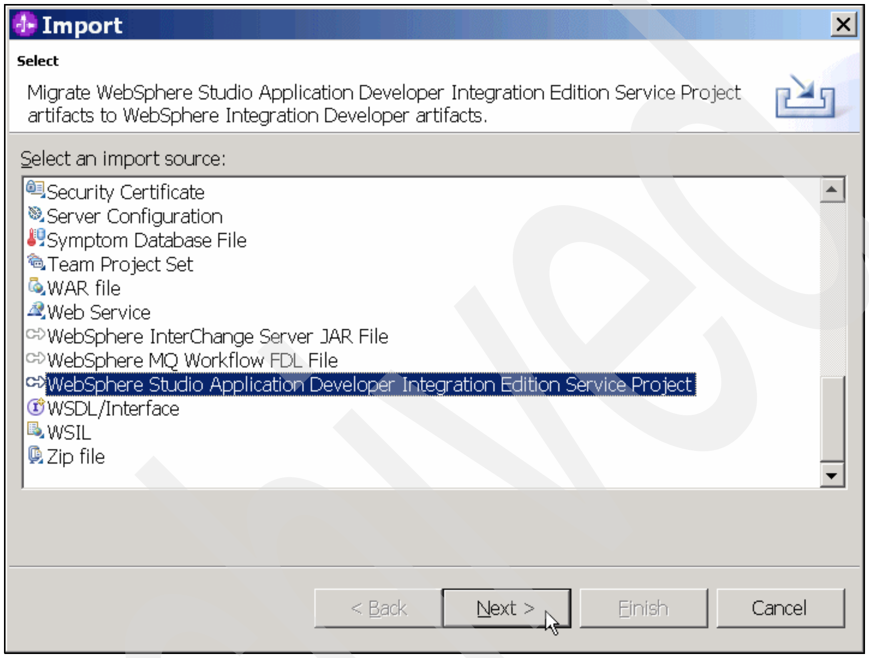
Further, it does not handle entire Integration Edition workspaces. It is meant to migrate one Integration Edition service project at a time.

6.4.2 Using the Migration wizard

Prepare the Integration Edition source artifacts *before* you use the WebSphere Integration Developer Migration wizard. For more information about this step, see section 6.3, "Preparing source artifacts for migration" on page 94.

To migrate service projects using the Migration wizard, use the following steps within WebSphere Integration Developer:

1. Invoke the wizard (6.5, “Manually completing the migration” on page 105) by selecting **File** → **Import** → **WebSphere Studio Application Developer Integration Edition Service Project**. The Migration wizard opens.



Migrate WebSphere Studio Application Developer Integration Edition Service Project artifacts to WebSphere Integration Developer artifacts.

- Security Certificate
- Server Configuration
- Symptom Database File
- Team Project Set
- WAR file
- Web Service
- WebSphere InterChange Server JAR File
- WebSphere MQ Workflow FDL File
- WebSphere Studio Application Developer Integration Edition Service Project
- WSDL/Interface
- WSIL
- Zip file

Figure 6-5 Invoke the Migration wizard within WebSphere Integration Developer V6.0.2

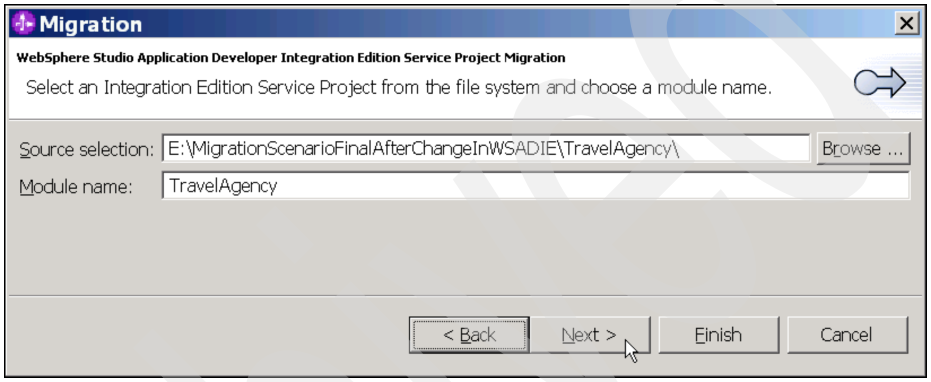
2. Enter the path for the Source Selection, or click the **Browse** button to find it. Select the **Integration Edition service project** that was already unzipped in the temporary folder in the migration preparation step, as shown in 6.6, “Migrating BPEL processes” on page 106.

   Note: You must migrate Service projects in their dependency order, for example, if a BPEL in service project A makes a process-to-process call to a BPEL in service project B, then migrate service project B before service project A; otherwise, the process-to-process call cannot be configured correctly.

   Also, enter the module name for the Business Integration module project to be created by the Migration wizard.

Note: We recommend that you choose the name of the service project as the module name because if there are other projects in the WebSphere Studio Application Developer Integration Edition workspace that are dependent on this project, you do not have to update the dependent projects' class paths after you import them into WebSphere Integration Developer.

Click **Next**.


Migration

WebSphere Studio Application Developer Integration Edition Service Project Migration

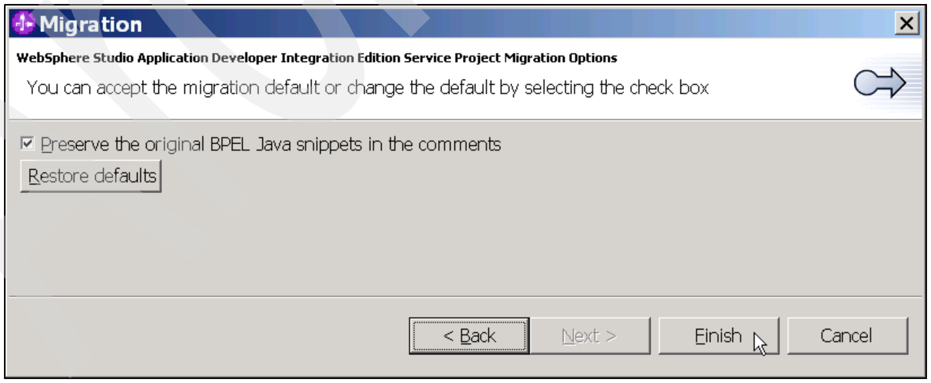
Select an Integration Edition Service Project from the file system and choose a module name.

Source selection: E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelAgency\

Module name: TravelAgency

Figure 6-6 Specify Service project to be migrated

3. Ensure that **Preserve the original BPEL Java snippets in the comments** is selected, as shown in 6.7, "Migrating Java services" on page 114. Click **Finish**. This starts the migration of the specified service project.


Migration

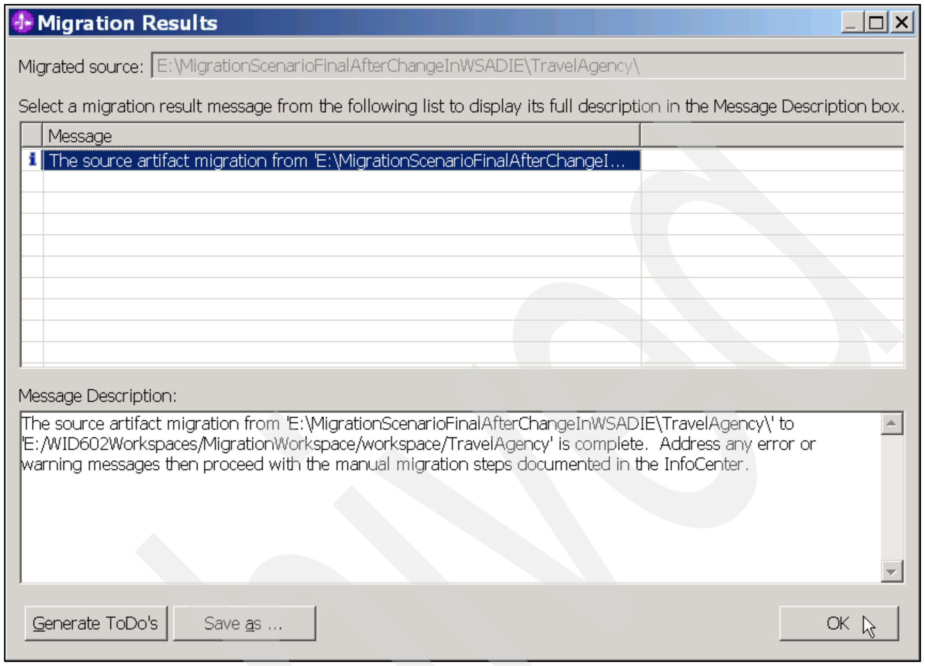
WebSphere Studio Application Developer Integration Edition Service Project Migration Options

You can accept the migration default or change the default by selecting the check box

☑ Preserve the original BPEL Java snippets in the comments

Figure 6-7 Select Preserve the original BPEL Java snippets in the comments

After the Migration wizard is finished, the results of the migration process are displayed in the Migration Results window, as shown in 6.8, “Migrating EJB services” on page 117.



E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelAgency\

Select a migration result message from the following list to display its full description in the Message Description box.

The source artifact migration from 'E:\MigrationScenarioFinalAfterChangeI...

The source artifact migration from 'E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelAgency\' to 'E:/WID602Workspaces/MigrationWorkspace/workspace/TravelAgency' is complete. Address any error or warning messages then proceed with the manual migration steps documented in the InfoCenter.

Figure 6-8 Migration Results window

4. In the Migration Results window, you can see the migration messages that were generated during the migration process. Select a message from the upper Message list to obtain more information regarding that message in the lower Message Description window. To keep all messages for future reference, click the **Generate ToDo's** button to create a list of "ToDo" tasks in the task view and click the **Save as** button to save the messages in a text file in the file system.

   A log file containing these migration messages is automatically generated to the Integration Developer workspace's .metadata folder. The log file is named ".log".

   If the Migration wizard fails, visit the following Info Center article to learn about how to deal with these failures.

   http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tmigfailsrcart.html

5. Examine each message to see if you need to take any action to immediately fix an artifact that was not fully migrated.

6. After the Migration wizard is complete, build the Business Integration module that was created, and resolve any build errors.
7. Inspect all of the migrated .bpel .wsdl files. Ensure that they are fully migrated and that you can open them the WebSphere Integration Developer BPEL Editor and Interface Editor.

After you use this Migration wizard, further manual migration steps might be required. We describe these steps and the affected artifacts in the next sections.

6.5 Manually completing the migration

After a successful artifacts migration to the new Business Integration module using the Migration wizard, you must wire together the artifacts to create an application that complies with the SCA model.

Note: Although the Migration wizard tries to successfully migrate the artifacts, you should also perform a manual verification.

To ensure that migration was correct:

- In the Business Integration perspective of the WebSphere Integration Developer, create one module for each migrated Service project.
- There must be an SCA component in the Assembly Diagram of WebSphere Integration Developer for each BPEL process in the migrated Integration Edition service project.

There are several areas of the application that might require manual rework. These can be, for example:

- The Migration wizard takes care of migrating Java snippets. Manual verification and possible modification is needed to ensure their full compliance with the SCA programming model.
- Verify and complete the wiring of components, imports, and exports in the Assembly Diagram of the migrated Business Integration module.

  Note: The migration utility attempts to do this automatically. However, we recommend manual verification.

  Projects may require some rewiring after migration to reconnect the services the way they were in Integration Edition V5.1; therefore, you must rewire all of the migrated business processes to their business partners.

For Integration Edition service projects that interacted with systems or entities that are external to the project, create an SCA import for the migrated project to access those entities as services according to the SCA programming model:

- Migrate Java services implementation. Java services code in WebSphere Business Integration Server Foundation V5.1 was the code for the BPEL Java partner. There are possible manual modifications that you must make to ensure that the Java service implementation complies with the SCA programming model, especially the SDO API.
- Migrate the client applications because the Migration wizard does not migrate client applications.

For further details about specific migration scenarios, refer to the following sections in this chapter and the WebSphere Integration Developer V6.0.2 Information Center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wbit.help.migration.ui.doc/topics/tmigsrcartman.htm

6.6 Migrating BPEL processes

You can migrate the BPEL processes with the help of the Migration wizard that is in WebSphere Integration Developer. This wizard 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. The Migration wizard also creates an SCA component in the Assembly Diagram for each .bpel process.

The following sub sections provide information and advice about additional manual migration steps that might be required.

6.6.1 Migrating BPEL Java snippets

The Java snippet API changed between Integration Edition and Process Server.

Whenever possible, the Migration wizard automatically migrates the Java snippets within a BPEL process, but there are snippets that the Migration wizard cannot fully migrate. Therefore there are required manual steps that you must perform to complete the migration.

Table 6-1 shows the changes in the BPEL Java snippet programming model and API from Business Process Choreographer V5.1 to V6. This information is useful to manually finish the migration of Java snippet code to the new API.

Table 6-1 Changes and solutions for migrating WebSphere Business Integration Server Foundation BPEL Java snippets

Change	Solution
WSIFMessage-based wrapper classes are no longer generated for wsdl message types, nor are the Java bean helper classes generated for complex schema types.	BPEL variables are directly accessed by name. The strongly-typed getters for the BPEL variables implicitly initialized the WSIFMessage wrapper object around the message parts. There is no 'wrapper' object for BPEL variables whose wsdl message definition has only a single part: in this case the BPEL variables directly represent the part (In the case where the single part is an xsd simple type, the BPEL variable is represented as the Java object wrapper type, such as java.lang.String, java.lang.Integer and so on.). BPEL variables with multi-part wsdl message definitions are handled differently: there is still a wrapper around the parts and this DataObject wrapper must be explicitly initialized in the WebSphere Process Server V6 Java snippet code, if it has not already been set by a previous operation. If any local variables from the 5.1 snippets had the same name as the BPEL variable, there may be conflicts, so try to remedy this situation if possible.
WSIFMessage objects are no longer used to represent BPEL variables.	If any custom Java classes that are invoked from the Java snippets have a WSIFMessage parameter, migrate the Java classes such that they can accept and return a DataObject.
Strongly-typed getter methods for BPEL variables are no longer available.	The variables can be directly accessed by name.
Strongly-typed setter methods for BPEL variables are no longer available.	The variables can be directly accessed by name.

Change	Solution
Weakly-typed getter methods for BPEL variables that return a WSIFMessage are no longer available.	You can access the variables directly accessed by name. For a Java snippet activity, the default access is read-write, which you can change to read-only by specifying @bpe.readOnlyVariables with the list of names of the variables in a comment in the snippet. Additionally, if a Java snippet in a condition, the variables are read-only by default, but can be made read-write by specifying `@bpe.readWriteVariables`
Weakly-typed setter methods for BPEL variables are no longer available.	You can access the variables directly accessed by name.
Weakly-typed getter methods for BPEL variables message parts are not appropriate for single-part messages and have changed for multi-part messages.	Migrate to the weakly-typed getter method for BPEL variables (DataObject's) properties. In WebSphere Process Server V6, there is no notion of using a variable for read-only access, which was the case in 5.1 for the first method above as well as the second method with `forUpdate='false'`. The variable is directly used in the WebSphere Process Server V6 snippet, and it is always able to be updated.
Weakly-typed setter methods for BPEL variables' message parts are not appropriate for single-part messages and changed for multi-part messages.	Migrate to the weakly-typed setter method for BPEL variables' (DataObject's) properties. Migrate calls to the following method: `setVariableObjectPart(String variableName, String partName, Object data)` For multi-part messages, equivalent functionality is provided by this method in WebSphere Process Server V6: `setVariableProperty(String variableName, QName propertyName,Serializable value);`
Strongly-typed getter methods for BPEL partner links are no longer available.	Migrate to the weakly-typed getter methods for BPEL partner links.

Change	Solution
Strongly-typed setter methods for BPEL partner links are no longer available.	Migrate to the weakly-typed setter methods for BPEL partner links.

Strongly-typed getter methods for BPEL correlation sets are no longer available.	**V5.1 Snippet:**
	`String corrSetPropStr =`
	`getCorrelationSetCorrSetAPropertyCustomerName(`
	`);`
	`int corrSetPropInt =`
	`getCorrelationSetCorrSetBPropertyCustomerId();`
	WebSphere Process Server V6 Snippet:
	`String corrSetPropStr = (String)`
	`getCorrelationSetProperty (“CorrSetA”, new`
	`QName(“CustomerName”));`
	`int corrSetPropInt = ((Integer)`
	`getCorrelationSetProperty (“CorrSetB”, new`
	`QName(“CustomerId”))).intValue();`

Additional parameter needed for the weakly-typed getter methods for BPEL activity custom properties.	**V5.1 Snippet:**
	`String val =`
	`getActivityCustomProperty(“propName”);`
	WebSphere Process Server V6 Snippet:
	`String val = getActivityCustomProperty`
	`(“name-of-current-activity”, “propName”);`

Additional parameter needed for the weakly-typed setter methods for BPEL activity custom properties.	**V5.1 Snippet:**
	`String newVal = “new value”;`
	`setActivityCustomProperty(“propName”, newVal);`
	WebSphere Process Server V6 Snippet:
	`String newVal = “new value”;`
	`setActivityCustomProperty(“name-of-current-act`
	`ivity”, “propName”, newVal);`

The raiseFault(QName faultQName, Serializable message) method no longer exists.	Migrate to the raiseFault(QName faultQName, String variableName) where possible; otherwise, migrate to the raiseFault(QName faultQName) method, or create a new BPEL variable for the Serializable object.

Note: If any local variables from the V5.1 Java snippets have the same name as a BPEL variable, there might be conflicts. In V6, Java snippets BPEL variables are predefined; therefore, it could come to naming clashes. Resolve this situation by renaming the relevant local variables in Integration Edition before attempting to migrate.

Note: BPEL variables whose WSDL message definition has a single-part are now directly represented by the part instead of having a wrapper around the actual data.

Variables whose message type has multiple parts, do have a DataObject wrapper around the parts (where the wrapper in Integration Edition was a WSIFMessage).

6.6.2 Migrating staff activities

Human interaction in WebSphere Process Server is performed through Human Task Manager, which provides the human task capabilities to business processes. The source artifacts that define the human tasks in Human Task Manager are either .itel or .tel files.

The Migration wizard migrates staff activities to in-line Human Task activities correctly, but note the following information:

- Check the client setting for staff activities, and you might need to perform a manual migration for missing migrated client settings.
- The Web client setting in Integration Edition is named Business Process Choreographer client setting in Integration Developer. Manually migrate missing settings.
- In WebSphere Business Integration Server Foundation, a business process could have one receive activity and multiple reply activities for the same operation. If a business process with multiple replies for the same operation exists, then ensure that if any of them has client settings that all replies for that operation have the same client settings. In WebSphere Process Server V6, only one set of client settings is supported per operation reply.
- For information about migrating custom JSPs, see the following Info Center article:

  `http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tmigclientbpcweb.html`

6.6.3 Migrating process invocation and binding

The following sections describe how to migrate the BPEL process binding and invocation style to the new SCA programming model.

Migrating a business process service invocation

This section applies to a business process that invokes another business process, where the second business process is invoked using a Web service Invocation Framework Process Binding.

To migrate a BPEL to BPEL service invocation:

1. Open the Assembly.
2. Wire references with interfaces or imports:
   - If the BPEL that is being invoked is in the same module, create a wire from the appropriate reference on the first BPEL component to the appropriate interface on the target BPEL component.
   - If the BPEL being invoked is in another module (where the other module is a migrated service project):
     i. Create an Export with SCA Binding for the second business process in its module Assembly Diagram.
     ii. Drag and drop the export from the Business Integration view, which is located under the second module, onto the open assembly editor of the first module. This creates an Import with SCA Binding in the first module.
     iii. Wire the appropriate reference on the first business process to the import that you just created in that module.
     iv. Save the Assembly Diagram.
3. Sometimes service binding information needs to be calculated at runtime and not at design time and hence called late binding and can be achieved when invoking the second business process by:
   a. Leave the first business process component's reference unwired.
   b. Open the first process in the BPEL editor and under the Reference Partners section, select the partner that corresponds to the second BPEL process to invoke using late binding.
   c. In the Properties view on the **Description** tab, enter the name of the second business process in the Process Template field.
   d. Save the business process.

Migrating EJB process bindings

You can migrate the EJB and the EJB process bindings to the recommended SCA construct.

In the Integration Edition, this binding type enabled clients to communicate with a BPEL process or other service type by invoking an EJB.

Note: This binding type was not optional for microprocesses – it was always selected because the generated EJB was used internally by the other binding types

The JNDI name of the generated EJB was automatically generated as a combination of the BPEL's name, target namespace, and valid-from time stamp, for example, Table 6-2 shows these attributes that can be found by examining the BPEL process's properties in the BPEL editor on the Description and Server content tabs:

Table 6-2 Generated namespace

Process name	**My service**
Target namespace	http://www.example.com/process87787141/
Valid from	Jan 01 2003 02:03:04

There are four options for migrating the Integration Edition process binding. The type of client(s) that access the service determine which migration option(s) to perform:

- ► Make the business processes accessible to another SCA component in the same module.
- ► Make the business processes accessible to other SCA modules and clients by export the component with SCA binding.
- ► Make the modules accessible by a non-SCA entity (for example, a JSP or a Java client) by creating a Standalone Reference.
- ► Make the business processes accessible by a Web services client by exporting the component with Web service binding.

Note: If the business process passes a reference to itself outside of its module (via a service reference), use the first option to create an export with SCA Binding for the business process. Only one business process per module can pass its service reference outside of the module because its export must be marked as the module default export. This is done by specifying *true* for the attribute called *default* of an export.

Migrating JMS process bindings

You can migrate the JMS and JMS process bindings to the recommended SCA construct.

In the Integration Edition, this binding type gives clients the ability to communicate with a BPEL process or other service type by sending a message to a message driven bean (MDB).

Note: This binding is default and mandatory for long running processes in Integration Edition. In fact, this binding type is the only binding type that is allowed for request-response interfaces of long running processes in Integration Edition.

The JNDI name that the JMS binding uses is a combination of the BPEL's name, target namespace, and valid-from time stamp.

When you select the JMS binding as the deployment type for a BPEL process, the following options are given in Integration Edition:

- JNDI Connection Factory - the default is jms*/BPECF* (this is the JNDI name of the target Business Process Choreographer queue connection factory)
- JNDI Destination Queue - the default is jms/BPEIntQueue (this is the JNDI name of the target Business Process Choreographers internal queue)
- JNDI Provider URL: Server supplied or Custom - You must enter an address. The default is *iiop://localhost:2809*

There are five options for migrating the Integration Edition JMS process binding. The type of client(s) that access the service determine which migration option(s) to perform:

- Make the business processes accessible to another component in the same module.
- Make the business processes accessible to other SCA modules and clients by exporting the component with SCA binding.
- Make the business processes accessible by a non-SCA entity (for example, a JSP or a Java client) by creating a Standalone Reference.
- Make the business processes accessible to a Web services client by exporting the component with Web service binding.
- Make the business processes accessible by a JMS client by exporting the component with JMS binding.

Note: If the business process passes a reference to itself outside of its module (through a service reference), use the first option from the previous bulleted list to create an export with SCA Binding for the business process. Only one business process per module can pass its service reference outside of the module because its export must be marked as the module default export, which you do by specifying *true* for the attribute called *default* of an export.

6.7 Migrating Java services

There are two main options for migrating Java services:

- Creating a custom Java component
- Creating a Java Web service around the Java service class.

Note: There are many new components in WebSphere Process Server that offer new functionality, such as data mapping, interface mediation, business state machines, selectors, or business rules. First determine whether one of these new component types can replace the custom Java component logic. If that is not possible, migrate the Java services.

6.7.1 Creating a custom Java component

The Java service migration technique that we recommend is to use the WebSphere Integration Developer Java component type that allows you to represent the Java service as an SCA component.

During migration, you must rewrite your custom Java code to be compatible with the SCA Java component's interface style.

To create a custom Java component from Java classes:

1. Open the Assembly Diagram, and drag and drop the WSDL interface of the Java service onto the Assembly Diagram, which creates an SCA component.
2. **Right-click** this component, and select **Generate Implementation** → **Java**.
3. Select the package where the Java implementation needs to be generated to.

This process creates a skeleton Java service that adheres to the WSDL interface of the SCA programming model, where complex types are represented by an object that is a commonj.sdo.DataObject, and simple types are represented by their Java Object equivalents.

Example 6-1 shows the relevant definitions from the Integration Edition WSDL interface.

Example 6-1 Definitions in the Integration Edition WSDL interface

```
<types>
   <schema xmlns="http://www.w3.org/2001/XMLSchema"
         attributeFormDefault="qualified"
         elementFormDefault="unqualified"
         targetNamespace="http://migr.practice.ibm.com/"
         xmlns:xsd1="http://migr.practice.ibm.com/">

         <complexType name="StockInfo">
            <all>
               <element name="index" type="int"/>
               <element name="price" type="double"/>
               <element name="symbol" nillable="true"
                         type="string"/>
         </all>
         </complexType>
   </schema>
</types>

<message name="getStockInfoRequest">
   <part name="symbol" type="xsd:string"/>
</message>
<message name="getStockInfoResponse">
   <part name="result" type="xsd1:StockInfo"/>
</message>

<operation name="getStockInfo" parameterOrder="symbol">
         <input message="tns:getStockInfoRequest"
                     name="getStockInfoRequest"/>
         <output message="tns:getStockInfoResponse"
                     name="getStockInfoResponse"/>
</operation>
```

Example 6-2 shows the Integration Edition Java methods that correspond to the WSDL interface.

Example 6-2 Java methods corresponding to the wsdl in Interface Editor

```
public StockInfo getStockInfo(String symbol)
{
      return new StockInfo();
}
```

```
public void setStockPrice(String symbol, float newPrice)
{
      // set some things
}
```

Example 6-3 shows the WebSphere Integration Developer Java methods for the same WSDL interface.

Example 6-3 Java methods for the same wsdl in WebSphere Integration Developer

```
public DataObject getStockInfo(String aString) {
      //TODO Needs to be implemented.
      return null;
}

public void setStockPrice(String symbol, Float newPrice) {
      //TODO Needs to be implemented.
}
```

As we show in Example 6-3, you must replace the *//TODO* tags with the new code. There are two options to do this:

- Move the logic from the original Java class to this class, and adapt it to use DataObjects.
  - This rework is necessary because the Java classes that are generated from WSDL definitions in Integration Edition have Web service Invocation Framework (WSIF) dependencies to eliminate.
- Create a private instance of the old Java class inside this generated Java class, and write code to:
  - Convert all parameters of the generated Java implementation class into parameters that the old Java class expects.
  - Invoke the private instance of the old Java class with the converted parameters.
  - Convert the return value of the old Java class into the return value type that was declared by the generated Java implementation method.
  - We recommend this option for consumption scenarios where the new WebSphere Process Server style Java components must consume the WSIF service proxies.

After you complete one of the options from the previous list, rewire the Java component. Rewire the Java component's interface:

- If this service is invoked by a business process in the same module, create a wire from the appropriate business process reference to this Java component's interface.
- If this service is invoked by a business process in another module, create an Export with SCA Binding, and from the other module, drag and drop this export onto that module's Assembly Editor to create the corresponding Import with SCA Binding. Wire the appropriate business process reference to that Import.
- If this service was published in Integration Edition to expose it externally, see the section 6.9, "Migrating Web services" on page 123 for instructions about how to republish the service.

6.7.2 Creating a Java Web service

Creating a Java Web service is an alternative option to consider. The Rational Application Developer Web services tooling allows you to create a Web service around a Java class. For more information about this option, see the following Info Center article:

```
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topi
c=/com.ibm.wbit.help.migration.ui.doc/topics/toutjavaopt2.html
```

6.8 Migrating EJB services

In this section, we describe how to migrate the original EJB service so that it is accessible by SCA components and imports in WebSphere Process Server.

If the Integration Edition Service project was dependent on another EJB, EJB client, or Java project, import those existing projects using the **File** → **Import** → **Existing Project into Workspace** wizard. This was usually the case when an EJB was referenced from a Service project.

If any .wsdl or .xsd files that are referenced from the service project exist in another type of project, create a new Business Integration Library with the same name as the old non-Service project, and copy all of those artifacts to the library.

There are two main options when migrating an EJB service:

- Creating an SCA import with Stateless Session Bean Binding
- Creating an EJB Web service

6.8.1 Creating an SCA import with Stateless Session Bean Binding

We recommend this migration technique for migrating an EJB service. Here, you create an SCA import with Stateless Session Bean Binding type, which allows you to invoke an EJB from an SCA component or export.

More information about integrating EJBs with WebSphere Process Server is in the following article:

http://www.ibm.com/developerworks/websphere/library/techarticles/0602_xu2/0602_xu2.html

Further, a Java component is needed to convert between the SCA Java and WSDL interface style and the existing EJB interface style.

This Java component should use the WSDL interface of the EJB service as the interface. This is for the scenario where a BPEL process is calling the EJB service. An SCA reference of a BPEL process must be WSDL-typed. Because the SCA import with Stateless Session Bean Binding uses a Java-style interface, the Java component in between is also used to mediate between these two SCA-interface styles.

Note: Even though the migration tool automatically handles this, any changes that made after migration to the interfaces and data types (business objects) that are involved in the EJB interface requires that you manually update the conversion code that we mentioned here. Errors may be displayed in WebSphere Integration Developer depending on the type of change made.

To create the Java component that mediates between the SCA programming model and the EJB:

1. Open the Assembly Diagram.
2. Drag and drop the WSDL interface of the Integration Edition EJB service onto the Assembly Diagram, which creates an SCA component. Right-click this component, and select **Generate Implementation** → **Java**. Select the package to where the Java implementation needs to be generated.

This creates a skeleton Java service that adheres to the WSDL interface of the SCA programming model, where complex types are represented by an object that is a commonj.sdo.DataObject, and simple types are represented by their Java Object equivalents.

Example 6-4 on page 119 shows the relevant definitions from the Integration Edition EJB service WSDL interface.

Example 6-4 Definitions in the Integration Edition WSDL interface

```
<types>
   <schema xmlns="http://www.w3.org/2001/XMLSchema"
         attributeFormDefault="qualified"
         elementFormDefault="unqualified"
         targetNamespace="http://migr.practice.ibm.com/"
         xmlns:xsd1="http://migr.practice.ibm.com/">
         <complexType name="StockInfo">
            <all>
               <element name="index" type="int"/>
               <element name="price" type="double"/>
               <element name="symbol" nillable="true"
                         type="string"/>
         </all>
         </complexType>
   </schema>
</types>
<message name="getStockInfoRequest">
   <part name="symbol" type="xsd:string"/>
</message>
<message name="getStockInfoResponse">
   <part name="result" type="xsd1:StockInfo"/>
</message>
<operation name="getStockInfo" parameterOrder="symbol">
         <input message="tns:getStockInfoRequest"
                     name="getStockInfoRequest"/>
         <output message="tns:getStockInfoResponse"
                     name="getStockInfoResponse"/>
</operation>
```

Example 6-5 shows the Integration Edition 5.1 Java methods.

Example 6-5 Integration Edition WSDL Java code

```
public StockInfo getStockInfo(String symbol)
{
      return new StockInfo();
}

public void setStockPrice(String symbol, float newPrice)
{
      // set some things
}
```

Example 6-6 shows the WebSphere Integration Developer V6.0 Java methods for the same WSDL within the created Java component.

Example 6-6 WebSphere Process Server WSDL Java code

```
public DataObject getStockInfo(String aString) {
   //TODO Needs to be implemented.
      return null;
}

public void setStockPrice(String symbol, Float newPrice) {
      //TODO Needs to be implemented.
}
```

To replace the *//TODO* sections with the correct code, first create a reference from this Java component to the actual EJB so that it can access the EJB according to the SCA programming model. Therefore the SCA import with Stateless Session Bean Binding is created next:

1. Keep the Assembly Editor open, and switch to the J2EE perspective. Locate the EJB project that contains the EJB service.
2. Expand its Deployment Descriptor: *project-name* item, and locate the EJB. Drag and drop it onto the Assembly Editor. If you are warned about project dependencies that need to be updated, select the **Open the module dependency editor** option, and click **OK**.
3. Under the J2EE section, ensure that the EJB project is listed, and if it is not, add it by clicking the **Add** button.
4. Save the module dependencies, and close that editor. Select the new SCA import that was created in the Assembly Editor. Click the Descriptions tab, and go to the Properties view to change the import's name and the display name to something more meaningful. On the Binding tab, the import type is automatically set to **Stateless Session Bean Binding** and the JNDI name of the EJB is already set appropriately.
5. In the Assembly Editor, select the Wire tool from the palette.
6. Click the Java component, and release the mouse.
7. Click the Stateless Session Bean Import, and release the mouse.
8. When you are asked **A matching reference will be created on the source node. Do you want to continue?**, click **OK**, which creates a wire between the two components.
9. In the Assembly Editor, click the Details tab, go to the Properties, and select the Java component. Expand References, and select the reference to the EJB that was just created. Update the reference's name if the generated

name is not very descriptive or appropriate. Remember the name of this reference for future use.

10. Save the Assembly Diagram.

The SCA programming model must be used to invoke the EJB from the Java component. Follow these steps to write the code that invokes the EJB through the Stateless Session Bean Import from the generated Java class:

1. Create a private variable (whose type is that of the remote EJB interface), as shown in Example 6-7.

Example 6-7 Remote EJB interface private variable

```
private YourEJBInterface ejbService = null;
```

2. If there are complex types in the EJB interface, also create a private variable for the BOFactory, as shown in Example 6-8.

Example 6-8 Creating a private variable for BOFactory

```
private BOFactory boFactory = (BOFactory)
   ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo
   /BOFactory");
```

3. In the constructor of the Java class, use the SCA APIs to resolve the EJB reference (remember to fill in the name of the EJB reference that you wrote down a few steps back), and set the private variable equal to this reference, as shown in Example 6-9.

Example 6-9 Resolving the EJB reference using SCA API

```
// Locate the EJB service
this.ejbService = (YourEJBInterface)
   ServiceManager.INSTANCE.locateService("name-of-your-ejb-reference");
```

4. For each *//TODO* in the generated Java class, do the following:
   a. Convert all of the parameters into the parameter types that the EJB expects.
   b. Invoke the appropriate method on the EJB reference using the SCA programming model, which sends the converted parameters.
   c. Convert the return value of the EJB into the return value type that the generated Java implementation method declares.

   Example 6-10 on page 122 shows the Java component implementation.

Example 6-10 Java component implementation

```
/**
* Method generated to support the implementing wsdl port type named
* "interface.MyBean".
*/
public DataObject getStockInfo(String aString) {
   DataObject boImpl = null;
   try {
      // invoke the EJB method
      StockInfo stockInfo = this.ejbService.getStockInfo(aString);

      // formulate the SCA data object to return.
      boImpl = (DataObject)
            this.boFactory.createByClass(StockInfo.class);

      // manually convert all data from the EJB return type into the
      // SCA data object to return
      boImpl.setInt("index", stockInfo.getIndex());
      boImpl.setString("symbol", stockInfo.getSymbol());
      boImpl.setDouble("price", stockInfo.getPrice());
} catch (RemoteException e) {
      e.printStackTrace();
   }
   return boImpl;
}

/**
* Method generated to support the implementing wsdl port type named
* "interface.MyBean".
*/
public void setStockPrice(String symbol, Float newPrice) {
   try {
      this.ejbService.setStockPrice(symbol, newPrice.floatValue());
} catch (RemoteException e) {
      e.printStackTrace();
   }
}
```

6.8.2 Creating an EJB Web service

Creating an EJB Web service is an alternative option by considering the Rational Application Developer Web services tooling that allows you to create a Web service around an EJB. For more information about this option, see the following article:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/toutejbopt2.html

6.9 Migrating Web services

For Web service support, WebSphere Business Integration Server Foundation V5.1 uses the Java API for XML-based Remote Procedure Call (JAX-RPC) and Web services Invocation Framework criterion to address different combinations of Web services bindings and implementations.

These two criterion provide standard APIs for invoking Web services from Java, although JAX-RPC does not specify how services can be invoked over anything other than SOAP over HTTP. Multi-protocol JAX-RPC adds this support from WebSphere Application Server V5.1.1.

Both Multi-protocol JAX-RPC and Web service Invocation Framework are supported by the WebSphere Application Server. Web service Invocation Framework is deprecated from V5.1.1; instead, use SCA invocation framework.

In the following section, we describe how to migrate Web services to be called from an SCA environment, such as an SCA component that represents a BPEL process. We also explain the differences between Server Foundation and Process Server with regards to Web services security.

6.9.1 Migrating a Web service (SOAP/JMS) or (SOAP/HTTP)

To migrate a SOAP/JMS or SOAP/HTTP service project for an outbound service migration, follow these steps:

1. Import the Integration Edition Service project using the Migration wizard, which results in the creation of a Business Integration module with the WSDL messages, port types, bindings, and services generated in Integration Developer.

   Note: If the Web service that this application invokes is also a WebSphere Studio Application Developer Integration Edition Web service that got migrated, there might have been updates to that Web service during the migration. If this is the case, use those migrated WSDL files here.

2. In the Business Integration module, open the Assembly Editor, which is going to invoke the Web service.

3. Open the Assembly Editor of the migrated Business Integration module. Ensure that the WSDL interface, binding, and service definitions are present in the migrated module or in a library on which the migrated module is dependent.
4. Expand the Web service Ports logical category. Drag and drop the port that corresponds to the service to be invoked onto the Assembly Editor of the Business Integration module, which is going to invoke the Web service.
5. Choose to create an Import with Web service Binding.
6. After you create the import, select it in the Assembly Editor, and go to the Properties view. Under the Binding tab, you can see the port and service that the import is bound to.
7. Save the Assembly Diagram.

After you complete this, wire the import using the following steps:

1. If this service is invoked by a business process in the same module, create a wire from the appropriate business process reference to this import.
2. If this service is invoked by a business process in another module, create an Export with SCA Binding, and from the other module, drag and drop this export onto that module’s Assembly Editor to create the corresponding Import with SCA Binding. Wire the appropriate business process reference to that import.
3. Save the Assembly Diagram.

6.9.2 Migrating Web service Security

WebSphere Application Server Versions 5.0.2, 5.1, and 5.1.1 support the following specifications:

- Web services security: SOAP Message Security Draft 13 (formerly Web services security Core Specification)
- Web services security: Username Token Profile Draft 2

In April 2004, the Web service security specification (officially called Web services Security: SOAP Message Security Version 1.0) became the Version 1.0 OASIS standard. Also, the Username token and X.509 token profiles are Version 1.0 specifications.

WebSphere Application Server V6 supports the following Web services security specifications from OASIS:

- Web services security: SOAP Message Security 1.0 specification
- Web services security: Username Token 1.0 Profile
- Web services security: X.509 Token 1.0 Profile

There is a major difference between the two security profiles, which leads to the situation that both the client and the Web service provider should use the security profile V1.0 specification and also run in WebSphere Application Server V6.

Both the Web service consumer and the provider that deploy and run on WebSphere Application Server V6 must use the WS-Security Version 1.0 profile because WebSphere Application Server V6 does not support the Draft 13 security profile for a Web service provider or even for a client that runs on Server Foundation V5.1 environment.

Manually reconfigure Web service security in WebSphere Process Server through the deployment descriptors to be able to use Web service security.

6.10 Migrating clients

Migrating client applications may need to be required because of the differences in the programming models of Server Foundation and Process Server. Any required code changes are manual. The migration tooling does not modify Server Foundation client applications in order to interact with Process Server.

In Server Foundation, the most important APIs to take into consideration for a migration are the Business Process Choreographer APIs exposed as stateless session EJB and as generic JMS service.

Process Server provides several APIs to interact with components and applications. The most important for the migration considerations are:

- The SCA API
- The Business Process Choreographer APIs:
  - Business Flow Manager (BFM) API
  - Human Task Manager (HTM) API

In WebSphere Process Server V6.0.2, there is no generic JMS API for Business Process Choreographer.

There are several new options for client development. Clients can make use of the SCA API to interact with SCA Standalone References. They can also interact through several bindings, such as JMS or Web Services, with SCA Exports. Further, clients can make use of the Business Process Choreographer APIs to interact with processes and human tasks. Here, there is also the new option to use Web Services calls to communicate with the Business Process Choreographer.

One decision to make during the migrating of client code from Server Foundation to Process Server is therefore, whether to adopt the client to one of the new options that Process Server provides. In the following sections, we describe some of the migration options.

For further detailed discussion on the tasks involved with migrating client applications from Server Foundation, refer to the Process Server information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tmigclient.html

6.10.1 Migrating to clients using SCA programming model

This option to migrate the Server Foundation client code is to adopt the client to the new 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 using the assembly editor in the natural environment of WebSphere Process Server. Figure 6-9 illustrates the client migration options using the SCA programming model.

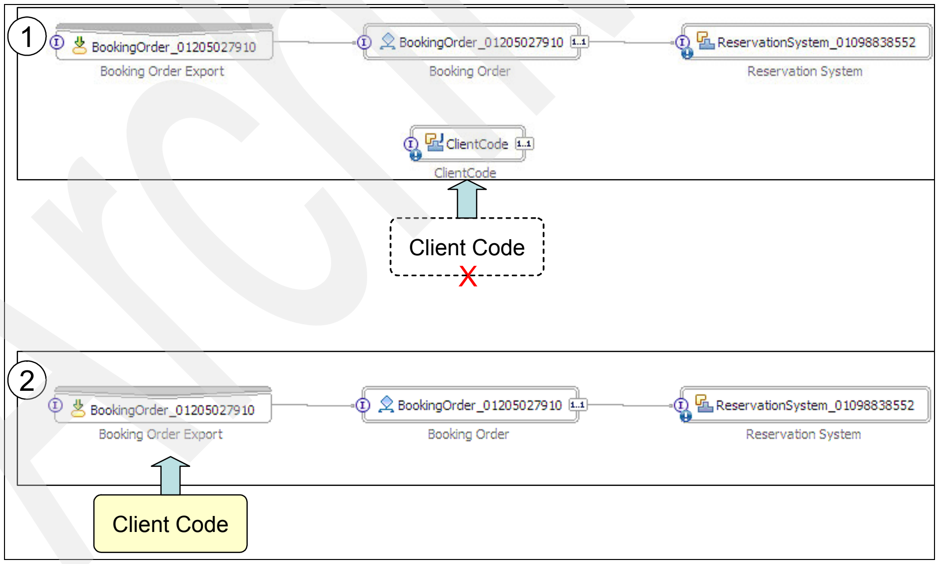


Figure 6-9 Client migration options using SCA programming model

For elements outside of the SCA module, invoke the process component or any other SCA component using SCA Exports.

Example 6-11 is an example of calling a process.

Example 6-11 Calling a process

```
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.createByElement("http://data/BookingOrder","Input");
   input.set("message", "data to start the process");
   DataObject out = (DataObject) service.invoke("Input", input);
} catch (Exception ex){
   ...
}
```

This case is appropriate for the clients that need to interact with the process using only the input and output that is defined in the export of this process. The client, in this case, does not require other data of the process or to know the life cycle of the process.

For further information about this topic, visit the following Web site:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tmigclientejb.html

6.10.2 Migrating to clients using Process Choreographer API

In Server Foundation, the Business Process Choreographer API offers a rich set of methods to interact with Business Process Choreographer processes and tasks.

Figure 6-10 on page 128 illustrates clients using the Business Process Choreographer API.

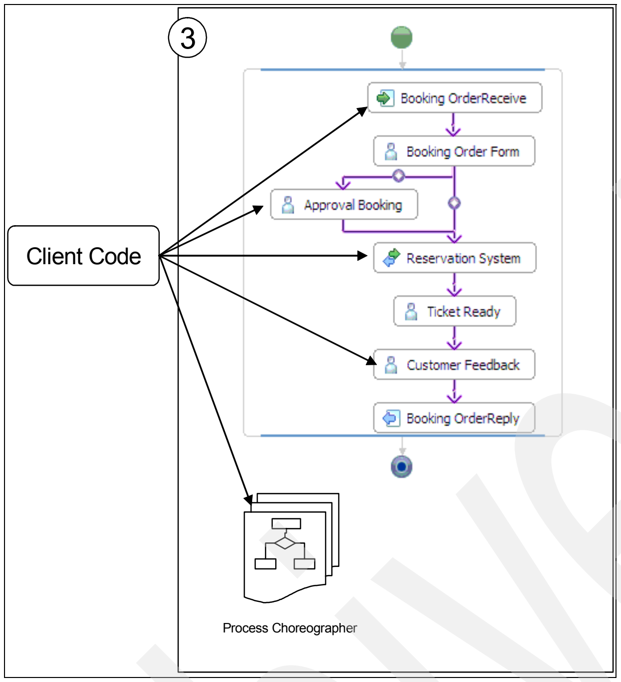


Figure 6-10 Clients using the Business Process Choreographer API

In Server Foundation, the Business Process Choreographer provides this API functionality through a stateless session EJB or through using JMS invocations.

In WebSphere Process Server V6.0.2, the API is published as two EJBs or using invocations through Web services. Further, the functionality was enhanced and changed in some instances.

Note: Many well known method names in Server Foundation V5.1 BPC API are maintained; however, some of the methods used in V5.1 are deprecated in Process Server V6.0.2. Therefore we recommend that you verify the migrated client code for the usage of these deprecated methods and replace them accordingly.

Functionality enhancements

In both Server Foundation and Process Server, the API functionality allows you to:

- 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). Using this PIID in the method *getProcessInstance(PIID),* an instance of the class *ProcessInstanceData* 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, kind of the activity (invoke, assign, reply, receive and so on.), owner, and the input and output message types.

Process Server provides enhanced support for client interactions with human tasks. The most important methods for interacting with the Human Task Manager are:

- callTask
- cancelClaim
- claim
- complete
- createFaultMessage
- createInputMessage
- createOutputMessage
- createWorkItem
- delete
- deleteWorkItem
- getCustomProperty and setCustomProperty
- getDocumentation
- getFaultNames
- getFaultMessage and setFaultMessage
- getInputMessage
- getOutputMessage and setOutputMessage
- getRoleInfo
- getTask
- resume
- startTask
- startTaskAsSubtask
- suspend

- terminate
- transferWorkItem
- updateInactiveTask
- updateTask

Like in Server Foundation, methods that can be called depend on the authorization of the person that uses the client code and the actual 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. Ask the Business Process Choreographer for processes using the query() method or for process templates using the queryProcessTemplates() method. Both methods require as arguments:

- Select clause: Describes the query result, that is, the names of the table views and columns of the query to be returned. Each entry should be of the form VIEW.COLUMN, where the views are:
  - ACTIVITY
  - ACTIVITY_ATTRIBUTE
  - EVENT
  - PROCESS_TEMPLATE
  - PROCESS_INSTANCE
  - PROCESS_ATTRIBUTE
  - WORK_ITEM
- Where clause: It is the search condition to be applied to the query.
- Order-by clause.
- Maximum number of query results.
- Time zone for the constants in the where clause. When its value is null UTC is assumed.

Example 6-12 shows a query for all of the activities.

Example 6-12 Query example

```
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));
}
```

The queries in 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* and a JMS API to invoke a subset of the EJB API functionality.

The JMS API is not present in WebSphere Process Server V6.0.2. There are several ways to migrate clients using this API. See the section "Migrating JMS process bindings" on page 112 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 a remote interface. Example 6-13 shows the JNDI name of this interface.

Example 6-13 Process Choreographer EJB interface in Server Foundation

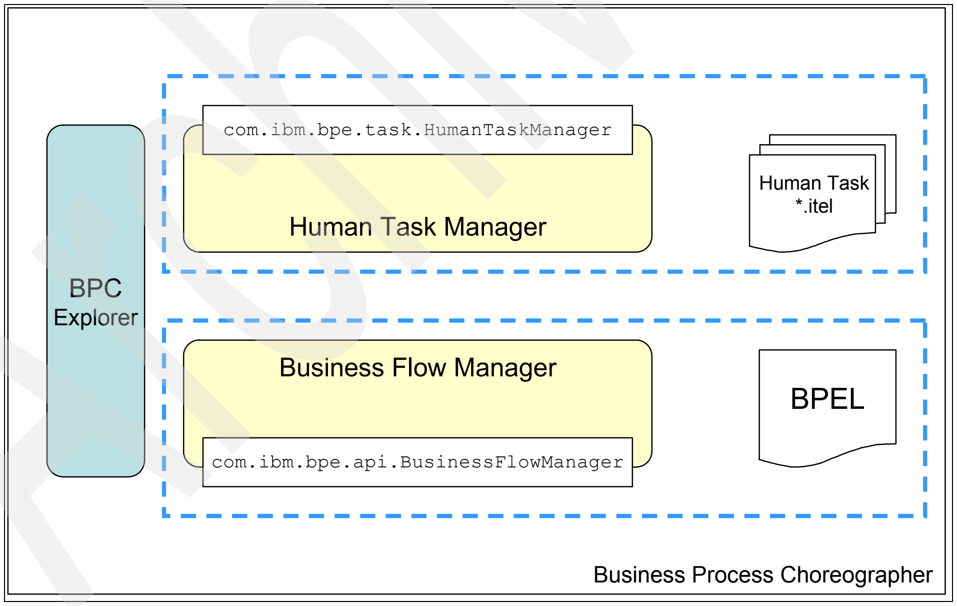
```
JNDI Name: com/ibm/bpe/api/BusinessProcessHome
Interface: com.ibm.bpe.api.BusinessProcess
```

In order to instantiate and send a message as input to a process, the code in Server Foundation is similar to Example 6-14.

Example 6-14 Example initiating a process in Server Foundation

```
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. 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 6-11.



com.ibm.bpe.task.HumanTaskManager
Human Task Manager
Human Task
*.itel
BPC
Explorer
Business Flow Manager
com.ibm.bpe.api.BusinessFlowManager
BPEL
Business Process Choreographer

Figure 6-11 Business Process Choreographer API in Process Server

Example 6-15 on page 133 shows the JNDI names of these interfaces.

Example 6-15 Process Choreographer EJB interfaces in Process Server

```
BusinessFlowManager EJB
JNDI Name: com/ibm/bpe/api/BusinessFlowManagerHome
Interface: com.ibm.bpe.api.BusinessFlowManager

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 Server Foundation.

Web services API in Process Server

To invoke the functionality of the Business Flow Manager and the Human Task Manager, Process Server offers two separate Web services interfaces (WSDL port types). Using the Web services API, the client could be a non-java component, such as .NET.

Message changes

Many of the BusinessProcess interface methods in Server Foundation and in the BusinessFlowManager and HumanTaskManager interface methods in Process Server require a ClientObjectWrapper as an input parameter. The content of this ClientObjectWrapper object has significantly changed from Server Foundation to Process Server.

The data in the process is managed by Server Foundation using the Web service Information Framework. It 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. 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 Server Foundation, the content of such a ClientObjectWrapper object has to be an instance of *WSIFMessage*. Example 6-16 shows the messages in Server Foundation.

Example 6-16 Messages in 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 the Service Data Object (SDO) in Process Server, the Web service Invocation Framework is no longer used. The messages in the process are now Business Objects that adopt the SDO API. The ClientObjectWrapper instance in Process Server requires a DataObject as content instead of a WSIFMessage. Example 6-17 shows messages in the Process Server.

Example 6-17 Messages in Process Server

```
   //defining data
   BOFactory bof =
(BOFactory)serviceMgr.locateService("com/ibm/websphere/bo/BOFactory");
   DataObject input =
bof.createByElement("http://data/BookingOrder","Input");
   input.set("message", "data to start the process");
   DataObject out = (DataObject) service.invoke("Input", input);
```

Client applications that make use of this API need to change the way they construct their ClientObjectWrapper instances when migrating to Process Server.

6.10.3 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 5.3, “Migration approaches” on page 79. Each project has different constraints, which define the approach of the migration:

- Coexistence and different clients for Server Foundation and Process Server system: In this case, each process uses different client code. The Server Foundation client code remains working changeless. New code is required for Process Server.

- Coexistence and one client for both, Server Foundation and Process Server system: In this case, the client code acts as a façade for the two systems. Develop new client code using the old code as code fragments to rewrite the new version.
- Move at once: In this case, write new client code. For some tasks, also consider that the client can use a non-java programming language, such as .NET.

6.11 Migrating WSIFMessage API calls

The data and invocation model in WebSphere Process Server differs from the programming model used in Server Foundation.

While in Server Foundation, the data model was based on the Web Service Invocation Framework (WSIF), while in Process Server, it is based on Service Data Objects (SDO). The Web service Invocation Framework-based invocation model that is within Server Foundation is based on the Service Component Architecture (SCA) in Process Server.

Web Service Invocation Framework is not supported in WebSphere Process Server V6, which implies the need for migration between these two programming models.

Service Data Objects (SDOs) are used as 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 being exchanged between Service Component Architecture services.

A Business Object is a set of attributes that represent a business entity, such as Employee, an action on the data, such as Create, Read, Update, and Delete (CRUD operations), and instructions for processing the data. Components of the integration application use Business Objects to exchange information and to trigger actions.

SDO Business Objects are the primary data abstraction for the Service Component Architecture (SCA), for example, Figure 6-12 on page 136 illustrates how SCA provides the framework to define service components and compose these services into integrated application, and it further shows that business objects represent the data that flows between each service.

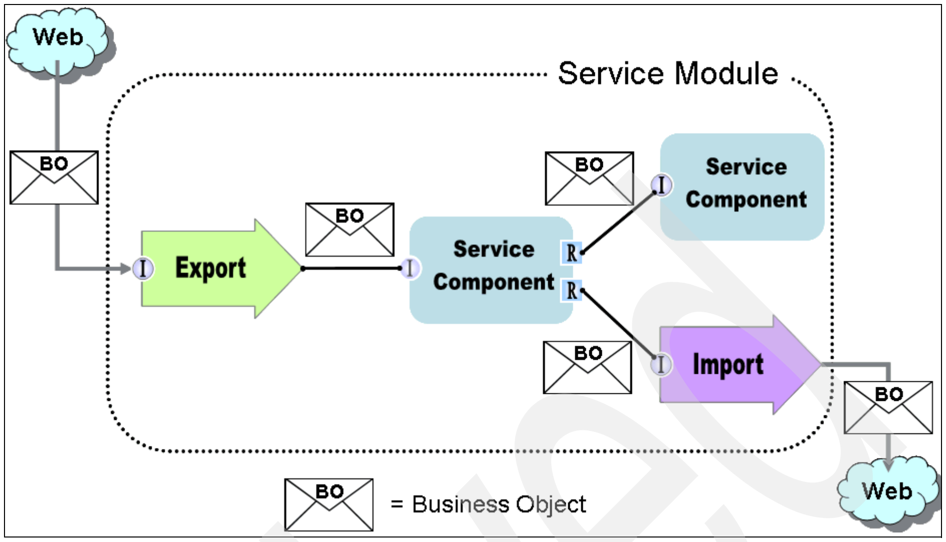


Figure 6-12 Business Objects for Service Component Architecture

In this section, we show how to migrate from the Web Service Invocation Framework programming model to the new SCA programming model where the data flowing through the application is stored in Service Data Objects.

Java snippets within 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, but there are snippets that the Migration wizard cannot fully migrate, meaning manual steps are required to complete the migration.

The following table details 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 Service Data Objects, and no strongly-typed interface is generated.

Table 6-3 on page 137 shows the changes and solutions for migrating WSIFMessage API calls to SDO APIs.

Table 6-3 Changes and Solutions for migrating WSIFMessage API calls to SDO APIs

Change	Solution
WSIFMessage-based wrapper classes are no longer generated for wsdl message types. The Java bean 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 commonj.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 Java beans should not be used.	Use weakly-typed SDO API to get the DataObject properties.
Strongly-typed setter methods for BPEL variables' message parts are no longer available.	Use weakly-typed SDO API to set the DataObject properties.
Weakly-typed getter methods for WSIFMessage properties should no longer be used.	Use weakly-typed SDO API to set the DataObject properties.
Weakly-typed setter methods for WSIFMessage properties should no longer be used.	Use weakly-typed SDO API to set the DataObject properties.
All WSIFMessage API calls should be migrated to the SDO API where possible.	Migrate the call to an equivalent SDO API call where possible. Redesign logic if not possible.

6.12 Migrating Business Rule Beans applications

There is no Integration Developer tooling support for the Server Foundation Business Rule Beans. However, if the Business Rule Bean artifacts are compiled in Integration Developer, the WebSphere Integration Developer documentation to install those deprecated features over the top of the embedded WebSphere Process Server test server must be followed:

http://www-1.ibm.com/support/docview.wss?uid=swg21217937

Then, add the appropriate deprecated jar files to the project classpath as external jars.

In general, use the new Business Rule tooling that is available in WebSphere Integration Developer to recreate the business rules according to the Process Server business rules specification.

6.13 Migrating Container Managed Persistence over Anything applications

For Server Foundation EJB projects with Container Managed Persistence over Anything (CMP/A), migrate the deployment descriptor files after the Server Foundation project is imported into the new Integration Developer workspace.

See the WebSphere Integration Developer Information center for more information:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tejbproj.html

6.14 Migrating Extended Messaging applications

The Extended Messaging (CMM) feature of Server Foundation is deprecated in Process Server, but your applications can still use this feature in Process Server.

To run Extended Messaging applications from Server Foundation on Process Server, perform a source artifact migration as described in the WebSphere Integration Developer Infocenter:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tejbproj.html

Basically, you must migrate the deployment descriptor entries.

Furthermore, you must perform a custom installation of WebSphere Process Server, as described in 7.6, “Migrating Extended Messaging” on page 166.

There is no Integration Developer tooling support for the Server Foundation Extended Messaging. However, if the Extended Messaging artifacts must be compiled and run in Integration Developer, the WebSphere Integration Developer documentation to install those deprecated features over top of the embedded WebSphere Process Server test server must be followed:

`http://www-1.ibm.com/support/docview.wss?uid=swg21217937`

6.15 Migrating CEI applications

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

CEI is also a part of WebSphere Process Server. The APIs of CEI stayed the same, in general. So applications that use these APIs in Server Foundation need no change during a migration to Process Server.

6.15.1 Business Process Choreographer events

In Server Foundation, predefined events could be emitted from business processes. Based on the *business relevant* value of activities, processes, links, and variables, events were emitted to the Common Event Infrastructure.

In WebSphere Process Server, the event emission is based on a **.mon* file, which allows a more fine-grained choice on which events per activity, process, or variable should 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 runtime.

The business process events that are emitted in Server Foundation are still available in Process Server; however, they have slightly changed and their contents were enriched.

Therefore, custom applications that 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 Process Server.

For more information about the changes in the events, see the Infocenter description of the event contents for Server Foundation:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.websphere.wbifz.doc/info/wbifz/workflow/tasks/tmonitor.html`

For more information about the changes in the events, see the Infocenter description of the event contents for Process Server:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.mon.doc/doc/bpc/cmonitor_process.html

6.16 Migrating WebSphere Business Integration Adapters

WebSphere Business Integration Adapters consist of a collection of software APIs that provide native communication with the back end Enterprise Information Systems (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. While WebSphere Business Integration Adapters are capable of communicating 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 that allow customers and business partners to leverage their investment and continued use. New J2EE Connector Architecture (JCA) 1.5 WebSphere Adapters get staged over time.

If the Client is a WebSphere Business Integration Adapter, it is needed to use the Enterprise Service Discovery tooling in WebSphere Integration Developer to create the Import with JMS Binding by reading the back end and creates 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 6-13 on page 141 shows the WebSphere Business Integration Adapters support in WebSphere Process Server V6.

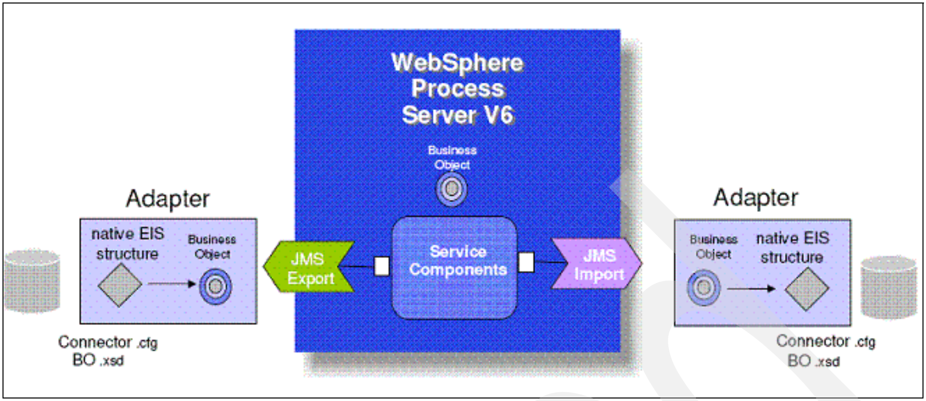


Figure 6-13 WBI Adapters architecture in WebSphere Process Server

The integration of these adapters is done using the same components and programming model as other SCA integration applications.

The center of Figure 6-13 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 Java Message Service (JMS) export. The business integration application can invoke other services outside of the SCA module through the use of a JMS import. The adapters communicate with the back end systems using the application specific data structure to business object and 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. WebSphere Process Server default messaging support is Service Integration Technologies messaging using *MQClientLink* is automatically configured the first time an Enterprise Application (EAR) that contains WebSphere Integration Adapter artifacts is installed. Destinations 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 ada binding, the incoming business object is converted from WebSphere Business Integration Format to the WebSphere Process Server business object or business graph format. The business object is then passed on to the SCA component. Imports receive request business object or business graph from an SCA component, convert the object from the WebSphere Process Server format to the WebSphere Business Integration business object format, and write the message to the JMS request destination queue.

The following list is a big picture for the Import/Export functions:

- Export
  - Listen for incoming events on the JMS delivery destination (*InboundDelivery*) or the synchronous request destination (*indoundRequest*)
  - Convert WebSphere Business Integration business object format to WebSphere Process Server business object or business graph format
  - Pass business object or business graph to the SCA component
  - Receive response/exception from the SCA component
  - Return response/exception to the synchronous response queue
- Import
  - Receive request business object or business graph from the SCA component
  - Convert business object or business graph to WebSphere Business Integration format
  - Write the message to the JMS request destination
  - Receive response/exception message from WebSphere Business Integration adapter on the response destination
  - Check the message for exceptions
  - Convert the response message to the business object or business graph
  - Return the response to the calling SCA component or throw the exception

6.16.1 Development tools for WebSphere Business Integration Adapters

Development tools consist of two major tools:

- WebSphere Business Integration Adapter and System Manager
- WebSphere Integration Developer

WebSphere Business Integration Adapter and System Manager are still used when working with adapters in WebSphere Process Server V6 to:

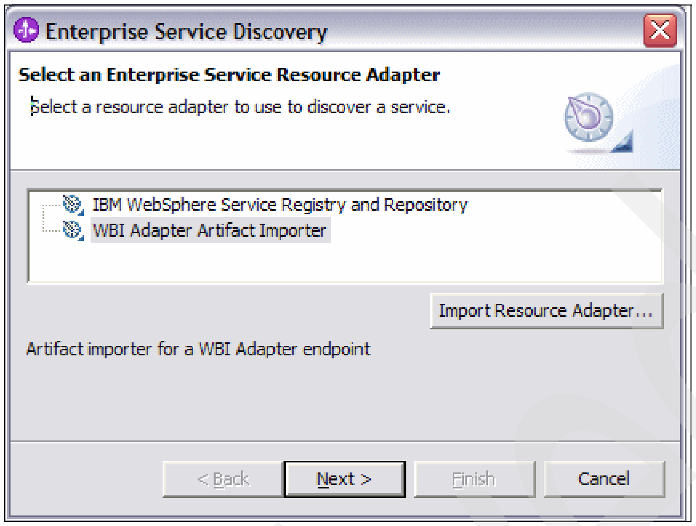
- Create Business Objects
  - Business Object Designer
  - Object Discovery Agent (ODA)
- Create Connector Configuration
  - Connector Configuration Editor
- Test
  - Visual Test Connector

WebSphere Integration Developer tools are used to create the necessary artifacts for integration with the WebSphere Process Server. Using WebSphere Integration Developer and the enterprise service discovery wizard, discover existing WebSphere Business Integration business objects and connector configuration, and generate the SCA artifacts, such as import and export files, wsdl interfaces and business objects, or business graphs in a format that is compatible with WebSphere Process Server. After all of the artifacts are created, components get assembled into the integration solution, which results in an EAR file that can be exported and installed on the WebSphere Process Server.

6.16.2 Migration steps

To migrate:

1. From WebSphere Integration Developer V6.0.2, click **File** → **New** → **Enterprise Service Discovery**.
2. Select WebSphere Business Integration Adapter Artifact Importer, as shown in Figure 6-14 on page 144.


Enterprise Service Discovery

Select an Enterprise Service Resource Adapter

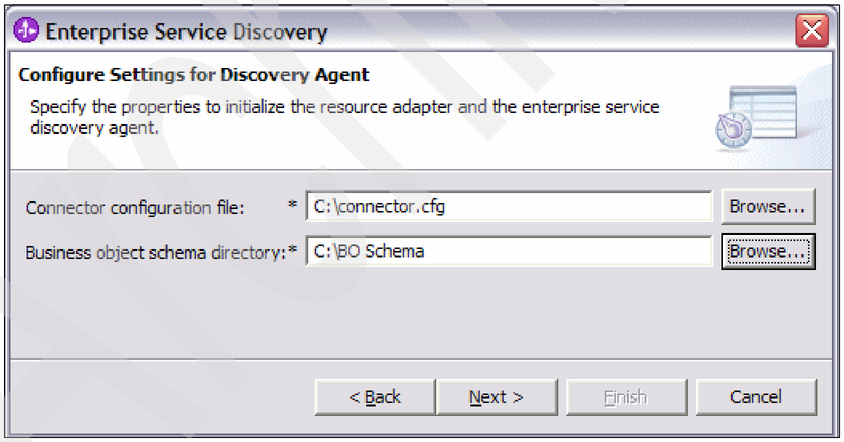
Select a resource adapter to use to discover a service.

- IBM WebSphere Service Registry and Repository
- WBI Adapter Artifact Importer

Artifact importer for a WBI Adapter endpoint

Figure 6-14 WebSphere Business Integration Adapter Artifact Importer

3. Configure the setting for the discovery agent, as shown in Figure 6-15:
   a. Browse to an existing connector configuration file.
   b. Browse to an existing business object schema directory.


Enterprise Service Discovery

Configure Settings for Discovery Agent

Specify the properties to initialize the resource adapter and the enterprise service discovery agent.

Connector configuration file: *	C:\connector.cfg
Business object schema directory:*	C:\BO Schema

Figure 6-15 Specifying initialization properties

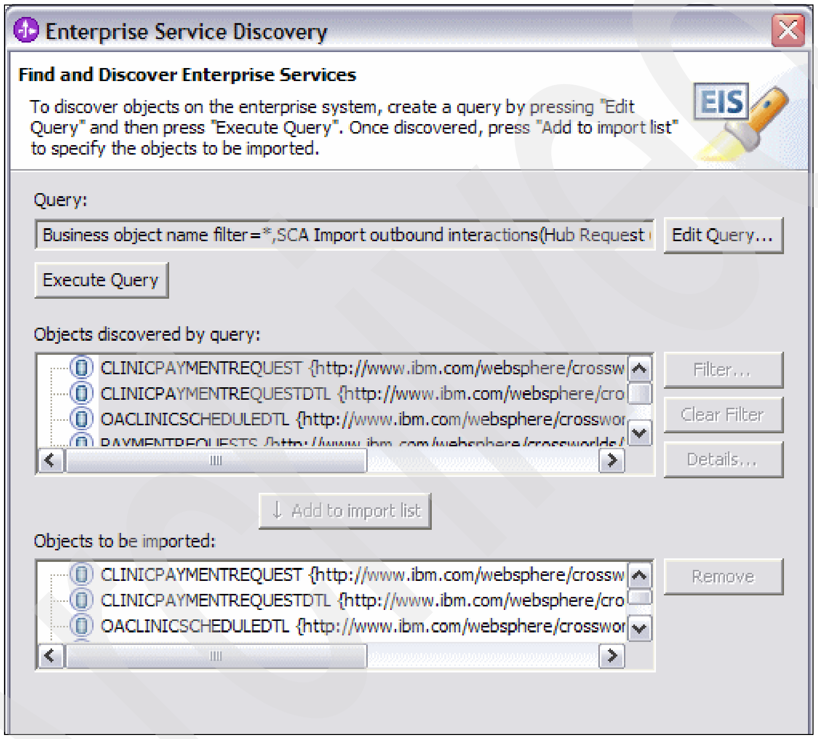
4. Find and discover enterprise services:
   a. After you specify the configuration file and the location of the business object schema, the next step is to select which business objects to be included in the application. Select **Edit Query** to provide options to filter the resulting list of business objects that are discovered and the

interaction mode in which they get used, such as outbound or inbound, request/response, or no way, and this determines the appropriate wsdl interfaces that get 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.

Figure 6-16 shows the Find and Discovery Enterprises Services window.


Enterprise Service Discovery

Find and Discover Enterprise Services

To discover objects on the enterprise system, create a query by pressing "Edit Query" and then press "Execute Query". Once discovered, press "Add to import list" to specify the objects to be imported.

Query:

Business object name filter=*,SCA Import outbound interactions(Hub Request

Objects discovered by query:

CLINICPAYMENTREQUEST {http://www.ibm.com/websphere/crossw
CLINICPAYMENTREQUESTDTL {http://www.ibm.com/websphere/cro
OACLINICSCHEDULEDTL {http://www.ibm.com/websphere/crosswor
PAYMENTREQUESTS {http://www.ibm.com/websphere/crossworlds/

Objects to be imported:

CLINICPAYMENTREQUEST {http://www.ibm.com/websphere/crossw
CLINICPAYMENTREQUESTDTL {http://www.ibm.com/websphere/cro
OACLINICSCHEDULEDTL {http://www.ibm.com/websphere/crosswor

Figure 6-16 Find and Discover Enterprise Services

5. 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 b business object. In many scenarios that involve WebSphere Business Integration Adapters, the business graph is the preferred format; however, this depends on the specific application requirements.

   b. Select the target project and Folder name.

c. Click **Finish**.

6.16.3 Artifacts generated

Table 6-4 lists the artifacts that are generated from the adapters creation process. If **generate a business graph** is selected, the **.xsd* and **BG.xsd* is available in the business integration module.

Table 6-4 Artifacts generated

Artifact	Description
*.xsd	Business object xsd file
*BG.xsd	Business graph of the supported business objects. These are the migrated versions of the original xsd files.
*ApplicationName*DeliveryHub.export	SCA export file that specifies the export for Adapter Delivery to Hub communication.
ApplicationNameRequestHub.export	SCA export file that specifies the export for Adapter Request to Hub communication.
ApplicationNameAgent.import	SCA import file that specifies the import for Hub to Adapter communication.
ApplicationName.wsdl	The wsdl file identifies the operations and messages that are used.
ApplicationNameWsadmin.wbia	The file used as input to automatically configure MQClientLink and JMS queues to allow communication between the server and the adapter framework.

7

Migrating runtime components

In this chapter, we discuss the following topics:

- Introduction to migration of runtime components
- General considerations
- Process Server topology
- Migrating Business Process Choreographer
- Migrating Business Rule Beans
- Migrating Extended Messaging
- Migrating the Common Event Infrastructure

7.1 Introduction to migration of runtime components

In this chapter, we cover aspects and considerations for the migration of the runtime environment from WebSphere Business Integration Server Foundation to WebSphere Process Server.

Because WebSphere Process Server is based on WebSphere Application Server V6, we recommend that you follow the migration procedures and principles for migrating to this platform. See the *WebSphere Application Server V6 Migration Guide*, SG24-6369 for detail information.

There are considerations and recommendations that apply specifically to the following Server Foundation components and services:

- Business Process Choreographer
- Business Rule Beans
- Extended Messaging (CMM)
- Common Event Infrastructure (CEI)

We describe these components in this chapter, as well as general considerations for a migration to WebSphere Process Server.

Migrating the runtime environment is a completely manual process. In this chapter, we also focus on the differences between Server Foundation and Process Server and information that is necessary as starting point for considerations regarding the runtime migration.

7.2 General considerations

WebSphere Business Integration Server Foundation and WebSphere Process Server are based on WebSphere Application Server.

In the following section, we provide some considerations that correspond to the relationship of Application Server and Process Server.

7.2.1 Product upgrade and configuration migration

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

Also, there is currently no way to preserve the configuration information of a Server Foundation runtime environment for the new Process Server environment. It is not possible to run a network deployment environment with a

Deployment Manager that manages both Server Foundation and Process Server nodes, servers, and clusters.

Build the deployment topology for the target Process Server environment from scratch.

7.3 Process Server topology

In this section, we discuss some important components and changes in Process Server with regards to migration from Server Foundation, which includes the new messaging infrastructure that Application Server provides and the databases that are used in Process Server in comparison to the databases that are used in Server Foundation.

For a complete list of system requirements for using Process Server, refer to the following Web site:

http://www-1.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&uid=swg27006205

7.3.1 Messaging in Process Server

Because Process Server is based on a newer version of WebSphere Application Server, one important change in the Application Server that is relevant to Process Server is that with WebSphere Application Server V6, the *service integration bus* was introduced.

Service integration bus

The service integration bus is a JMS 1.1 compliant JMS provider for reliable message transport and has the capability of intermediary logic to adapt message flow intelligently in the network.

Service integration bus capabilities 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

  Application servers or clusters that were added to the bus.

- Messaging engines

  The application server or cluster component 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 engine(s) and determines on which cluster member(s) the messaging engine(s) runs.

- Destinations

  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 JDBC™ database, to hold information, such as messages, subscription information, and transaction states. Messaging engines can share a database, each using its own set of tables.

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:

- 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 Process Server

Process Server has at least two, at most four buses.

The Service Component Architecture requires 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.

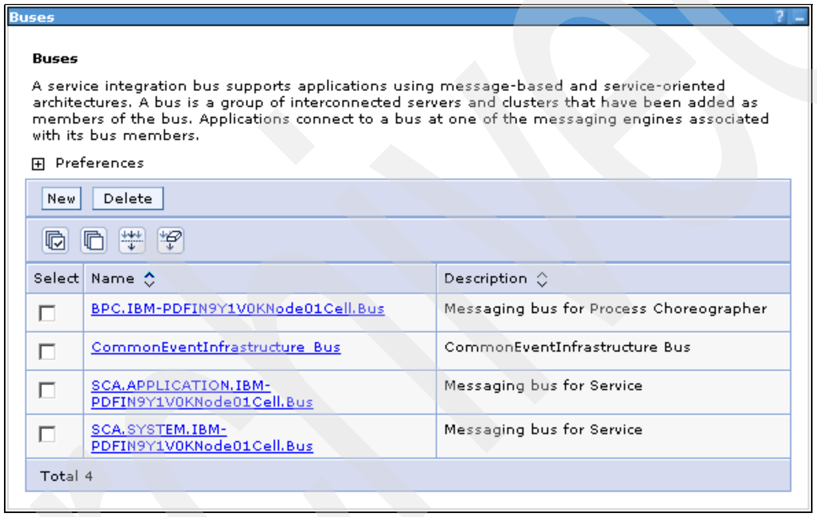
The Business Process Choreographer component of Process Server, consisting of the Business Flow Manager (BFM) and the Human Task Manager (HTM), requires one additional bus to execute business process and human task components within SCA modules:

- 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:

- CommonEventInfrastructure bus: This bus is used for transmitting common base events, asynchronously, to the Common Event Infrastructure server.

In Figure 7-1, the administrative console of a standalone Process Server is displayed, where all four buses are configured.


Buses

Buses

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

⊞ Preferences

Select	Name	Description
☐	BPC.IBM-PDFIN9Y1V0KNode01Cell.Bus	Messaging bus for Process Choreographer
☐	CommonEventInfrastructure_Bus	CommonEventInfrastructure Bus
☐	SCA.APPLICATION.IBM-PDFIN9Y1V0KNode01Cell.Bus	Messaging bus for Service
☐	SCA.SYSTEM.IBM-PDFIN9Y1V0KNode01Cell.Bus	Messaging bus for Service

Total 4

Figure 7-1 Buses that the Process Server uses

7.3.2 Databases required by Process Server

When you use the Business Process Choreographer, Server Foundation requires one database for storing business process templates and instance data.

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

Table 7-1 on page 152 lists the components of WebSphere Process Server that require database tables, the default database names, and where the tables associated with these components are stored. The database names are configurable.

Table 7-1 Databases within the Process Server

Database (default name)	Description	Components using the database
BPEDB	► Business Process Choreographer database. ► Stores template and instance data of business processes and human tasks. ► Required for executing business processes and human tasks.	► Business Processes ► Human Tasks ► Business State Machines
CEIDB	► Common Event Infrastructure database. ► Stores common base events.	Common Event Infrastructure
WPRCSDB	► Common database. ► Acts as a repository by various components. ► Required.	► Relationships ► Recovery ► Application scheduler ► Selectors ► Business rules
EsbLogMedDB	Messages processed by mediation modules can be logged in this enterprise service bus logger mediation database.	Mediation modules
Messaging engines data store	The messaging engine database is used by the message engines required for running Process Server. Every messaging engine has its own schema.	Service integration buses messages are stored here.

See the Process Server Info Center for more information about database specifications:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.602.ins.doc/doc/cins_db_specs.html

7.3.3 Selecting a target topology

Selecting the target topology for Process Server is an important step, which you must do early in a migration project.

Both Server Foundation and Process Server are based on the Application Server and use data stores and messaging to execute. Although this looks similar from a high level, there are major differences between the topologies of a Process Server environment and a Server Foundation environment. These differences are mainly caused by the introduction and use of the new service integration bus in Process Server, especially when it comes to environments that are built for scalability or high availability.

In Server Foundation, usually WebSphere MQ was used as a JMS provider. In Process Server, the service integration bus messaging must be used. There is a great variety of set up scenarios for Process Server, which depend on the non-functional requirements for the environment, such as scalability or high availability and the kind of components that are used in applications to be run on the environment.

Refer to the following articles for or more details and recommendations about a Process Server environment for your specific requirements:

- *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413

  `http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SG24-7413`

- *Basic steps for clustering WebSphere Process Server*

  `http://www.ibm.com/developerworks/websphere/library/techarticles/0704_chilanti/0704_chilanti.html`

- *Selecting your deployment pattern*

  `http://www-128.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html`

- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688

  `http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SG246688`

7.4 Migrating Business Process Choreographer

Business Process Choreographer gives you the ability to execute business processes within Server Foundation and Process Server.

In Server Foundation, one single component within Business Process Choreographer handled the business process navigation and supporting staff activities.

In Process Server, Process Choreographer was divided into two components, the Business Flow Manager and the Human Task Manager, which were enriched with new functionality. Basically, the focus on people interaction in Process Server was largely enhanced within the new Human Task Manager. The Business Flow Manager component handles the navigation of business processes as shown in Figure 7-2 on page 154.

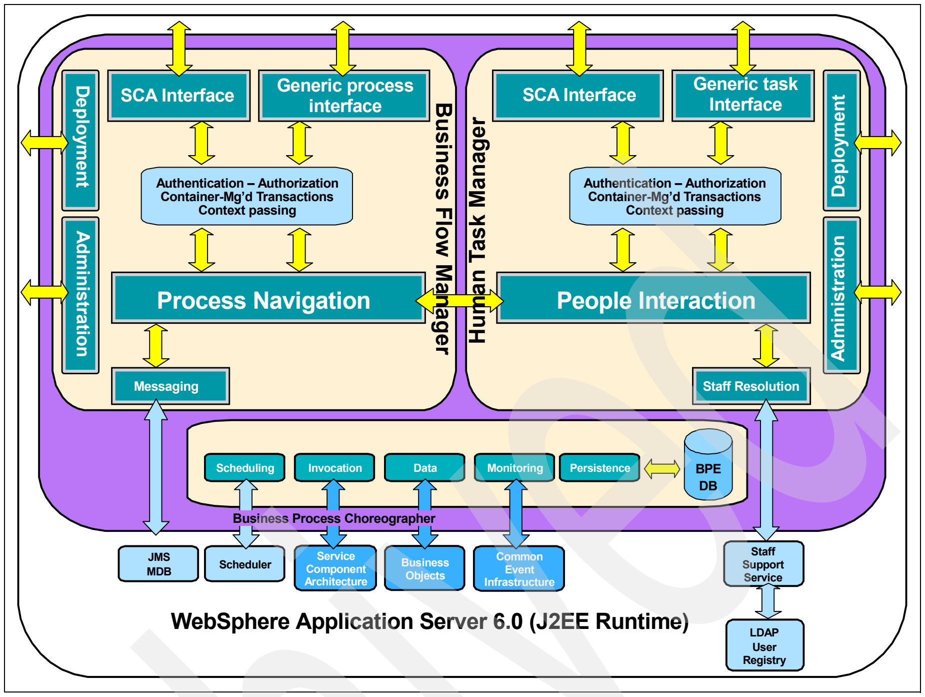


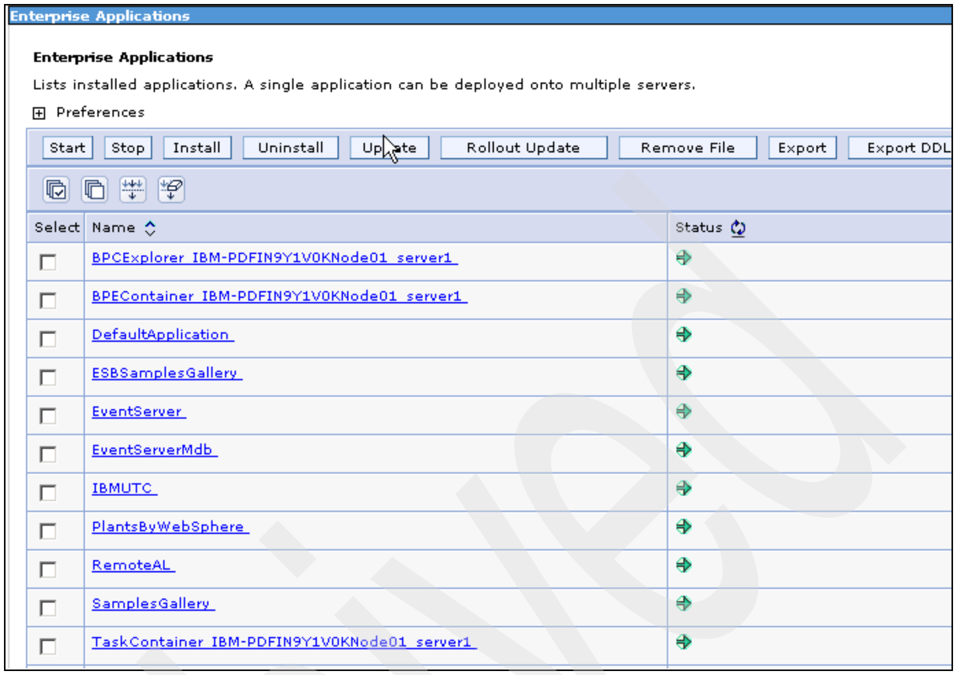
Figure 7-2 Componentized Business Process Choreographer within Process Server

For detailed information about Business Process Choreographer, visit the following Web site:

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

7.4.1 Business Process Choreographer applications

The division of Business Process Choreographer into two sub components is obvious in the runtime environment. There is one component that handles the business process navigation and the staff processing in Server Foundation. In Process Server, there is one component that functions as a container for executing business processes and another component that functions as a container for processing different kinds of human interaction. This change is reflected in the Process Server administrative console. Figure 7-3 on page 155 shows the Process Choreographer applications in a standalone Process Server environment.


Enterprise Applications

Enterprise Applications

Lists installed applications. A single application can be deployed onto multiple servers.

⊞ Preferences

Select	Name	Status
☐	BPCExplorer_IBM-PDFIN9Y1V0KNode01_server1	➔
☐	BPEContainer_IBM-PDFIN9Y1V0KNode01_server1	➔
☐	DefaultApplication	➔
☐	ESBSamplesGallery	➔
☐	EventServer	➔
☐	EventServerMdb	➔
☐	IBMUTC	➔
☐	PlantsByWebSphere	➔
☐	RemoteAL	➔
☐	SamplesGallery	➔
☐	TaskContainer_IBM-PDFIN9Y1V0KNode01_server1	➔

Figure 7-3 Admin console: Process Choreographer applications within Process Server

The BPEContainer_*nodename_servername* application represents the Business Flow Manager in the Process Server.

The equivalent to the Business Process Choreographer Web Client application in Server Foundation is now the Business Process Choreographer Explorer, which is represented by the BPCExplorer_*nodename_servername* application. This replaces the BPEWebClient_*nodename_servername* application in Server Foundation.

The division into two sub components started already in Server Foundation. Specifically with the service pack 1 of Server Foundation V5.1.1, the human task manager was introduced, which is why, in some Server Foundation systems, a HumanTaskManager_*nodename_servername* application exists, which represents the Human Task Manager in Server Foundation. This application is replaced in the Process Server by the TaskContainer_*nodename_servername* application.

In Server Foundation and in Process Server, *security roles* and *runAs roles* are associated with these applications. The same roles apply in both systems, but Process Server introduces additional and new roles.

Table 7-2 gives a general overview of the roles that are available in Process Choreographer with Server Foundation and Process Server:

Table 7-2 Security roles available in Business Process Choreographer (BPC) in both products

BPC application	Role	In Server Foundation	In Process Server
BPEContainer	► BPESystemAdministrator ► BPESystemMonitor ► WebClientUser ► JMSAPIUser	X X X	X X X X
BPEWebClient / BPCExplorer	WebClientUser	X	X
TaskContainer	► TaskSystemAdministrator ► TaskSystemMonitor ► EscalationUser	X (starting with V5.1.1.1)	X X X

All of the roles that were available in Server Foundation are also available in Process Server. When you migrate to Process Server, the security must take the new roles into account to assign the users to these roles.

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

7.4.2 Business Process Choreographer messaging resources

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

- BPEIntQueue
- BPEApiQueue
- BPERetQueue
- BPEHldQueue

In Process Server, the BPEApiQueue is not used anymore because the JMS API that is provided with Server Foundation is not supported in Process Server.

All other queues are also required in Process Choreographer within Process Server. Also, there are additional queues for the new human task container. The following queues are needed by Business Process Choreographer within Process Server:

- BPEIntQueue
- BPERetQueue
- BPEHldQueue

- HTMIntQueue
- HTMHldQueue

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

Default JMS provider

This is the service integration bus that we mentioned in Service integration bus. A Business Process Choreographer bus must be present when using 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 that we listed in the previous section, are 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. Documentation and scripts supporting this steps for a queue manager or a WebSphere MQ Cluster are available in the Info Center:

```
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topi
c=/com.ibm.wsps.ins.doc/doc/bpc/t2creq.html
```

We recommend that you avoid using WebSphere MQ messaging provider and prefer service integration bus for Business Process Choreographer in Process Server because of the following reasons:

- Although WebSphere MQ is supported for the Business Process Choreographer in 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 a relatively complex process.
- WebSphere Process Server utilizes 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 has to include both the service integration bus and WebSphere MQ as messaging providers, which is an unnecessary complication.

7.4.3 Business Process Choreographer database resources

Business Process Choreographer requires a relational database to execute business processes and human tasks. The Business Process Choreographer database (BPEDB) is required in both products, Server Foundation and Process Server.

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 BPEDB from the Server Foundation environment on the new Process Server environment. It is also not possible to move any kind of template or instance data from a Server Foundation BPEDB to a Process Server BPEDB.

The following information applies to both products. In a clustered Business Process Choreographer setup, one database serves all 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 Process Choreographer.

More information about creating the BPEDB in WebSphere Process Server is in the Info Center:

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/bpc/t2data.html`

7.4.4 Business Process Choreographer administration

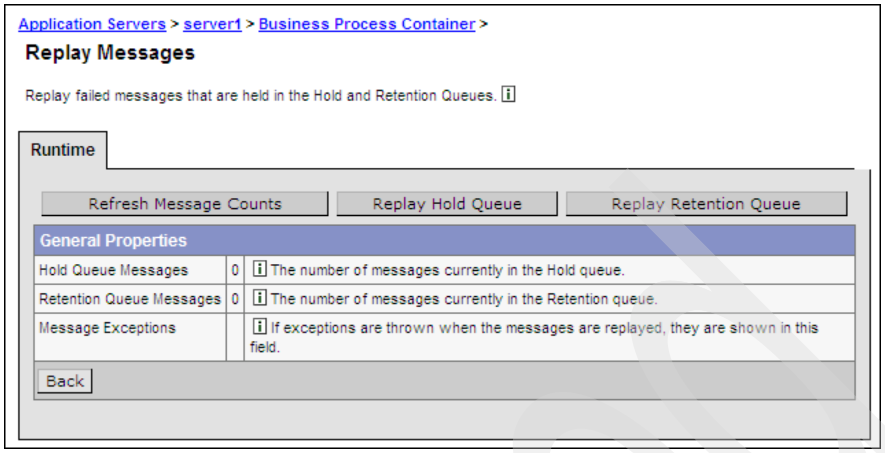
Administering the Business Process Choreographer remains the same for Server Foundation and Process Server. However, in Process Server the administering capabilities were enhanced by means of additional functionality in the administration console and a variety of new scripts.

Administering instance data, which you could do in the BPE Web Client in Server Foundation, can now be achieved using the BPC Explorer. This client application that is provided with Process Server provides the same functionality as the BPE Web Client and was enhanced to support human tasks. Further, it has better query functions and can visualize business processes and their current state.

Administration in Server Foundation

As shown in Figure 7-4 on page 159, use the administration console for Server Foundation, and select **Application servers** → ***server1*** → **Business Process Container** to access the administrative functions for the Business Process Container. The main functions are:

- Business Process Container Install Wizard
- Query and replay failed messages that are in the retention or hold queue


Application Servers > server1 > Business Process Container >

Replay Messages

Replay failed messages that are held in the Hold and Retention Queues.

Runtime

Refresh Message Counts | Replay Hold Queue | Replay Retention Queue

General Properties		
Hold Queue Messages	0	The number of messages currently in the Hold queue.
Retention Queue Messages	0	The number of messages currently in the Retention queue.
Message Exceptions		If exceptions are thrown when the messages are replayed, they are shown in this field.

Back

Figure 7-4 Replay messages runtime function in Server Foundation console

Scripts were provided for other administrative tasks, such as:

- Deleting audit log entries.
- Refreshing staff entries that are cached in the database.
- Removing unused staff entries that are cached in the database.
- Query and replay failed messages via scripts.

Further, on an application level, you could start and stop process templates.

Administration in Process Server

The capabilities of managing the Process Choreographer Containers in Process Server are enhanced. Within the Process Server administration console (Figure 7-5 on page 160), under **Application servers** → ***server1*** → **Container Settings**, are the administrative functions to manage the runtime and configuration environment for the Task Container and the Business Process Container.

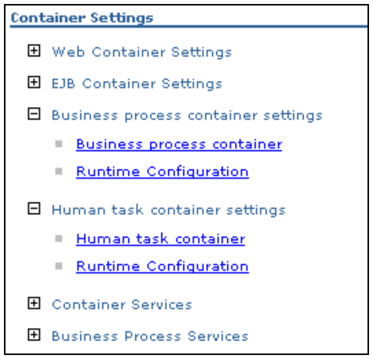


Figure 7-5 Configuration entry point for Business process and Human task container

When you make changes in the runtime configuration you do not have to restart a server to make the changes effective. After a server restart, the runtime changes are discarded, unless they were saved to a configuration as well.

Business Process Container

The capabilities for administering the Business Process Container in Process Server are:

- Business Process Container installation wizard.
- Query and replay failed messages that are in the retention or hold queue.
- Change the retry limit and the retention queue message limit.

  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 fed back into normal processing automatically. If the messages count on the retention queue reaches the retention queue message limit, the processing continues in quiece mode. Look for more information about this in the Process Server Info Center:

  `http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.602.bpc.doc/doc/bpc/c5replay.html`

  After the maximum number of retries, failed messages are put into the hold queue where an administrator can replay them.
- Enable or disable event emission to the Common Event Infrastructure or the AuditLog for all business processes that are running in the container.

Scripts are provided for administrative tasks:

- Deleting audit log entries.
- Deleting process templates that are no longer valid.

- Deleting completed process instances.
- Deleting data from the observer database.
- Querying and replaying failed messages.

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

Human Task Container

The most important capabilities for administering the human task container in Process Server are:

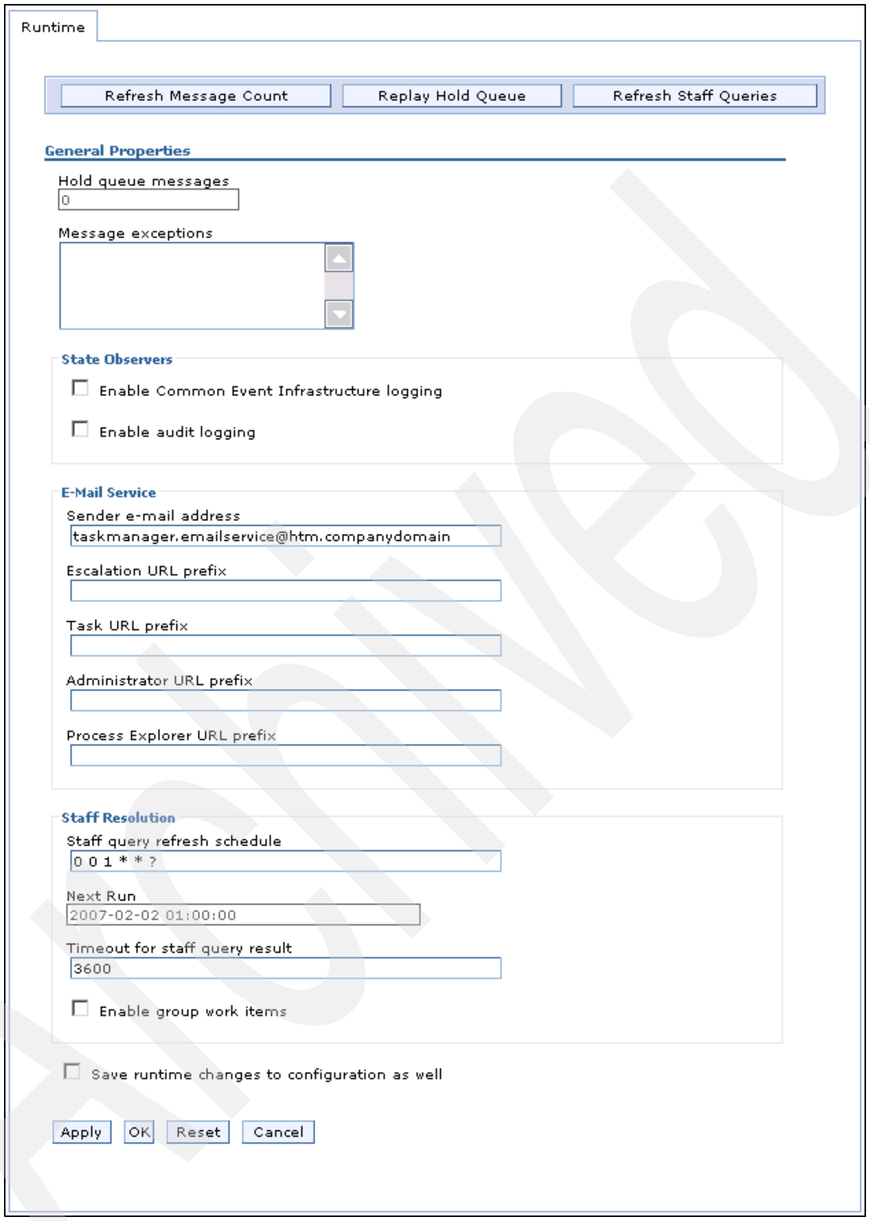
- Human task container installation wizard.
- Query and replay failed messages that are in the hold queue.

  For the task container, if a message cannot be processed, it is immediately stored in the task container's hold queue from where an administrator can replay it.
- Refresh staff query results using a refresh daemon.
- Enable or disable event emission to the Common Event Infrastructure or the AuditLog for all human tasks that are running in the container.

Scripts are provided for administrative tasks:

- Deleting audit log entries.
- Deleting task templates that are no longer valid.
- Querying and replaying failed messages.
- Refreshing staff query results.
- Removing unused staff query results.

Further, on an application basis, you can start and stop task templates. Figure 7-6 on page 162 shows the Process Server runtime configuration panel for the human task container.


Runtime

General Properties

Hold queue messages: 0

Message exceptions:

State Observers

☐ Enable Common Event Infrastructure logging

☐ Enable audit logging

E-Mail Service

Sender e-mail address: taskmanager.emailservice@htm.companydomain

Escalation URL prefix:

Task URL prefix:

Administrator URL prefix:

Process Explorer URL prefix:

Staff Resolution

Staff query refresh schedule: 0 0 1 * * ?

Next Run: 2007-02-02 01:00:00

Timeout for staff query result: 3600

☐ Enable group work items

☐ Save runtime changes to configuration as well

Figure 7-6 Process Server runtime configuration panel for human task container

7.4.5 Staff support service and staff-plugin configuration

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 using a staff plugin that is specific to the directory service, as shown in Figure 7-7.

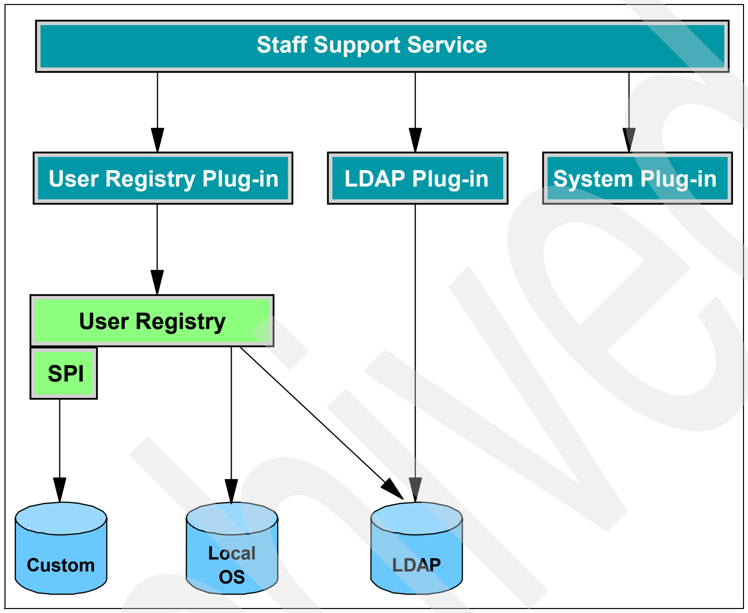


Figure 7-7 Staff resolution plug-ins in Server Foundation and Process Server

In general, the staff resolution architecture has not changed between Server Foundation and Process Server.

Each of the staff plugins is associated with at least one configuration. In particular, a configuration specifies an XSL transformation 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 was extended. Additional predefined staff verbs in WebSphere Process Server V6.0.2 are:

- Users by user ID without Named Users: Assigns users, given their user ID except the ones that are explicitly excluded.
- Group Members without Named Users: Assigns all users of a group except the ones that are explicitly excluded.

- Group Members without Filtered Users: Assigns all users of a group except the ones that are found using a search filter.
- Group: Assigns group members, but as a group work item. Also supports WebSphere security dynamic groups.

Already in 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 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 to the company's LDAP schema. These customized files can be moved from the Server Foundation environment to the new Process Server environment. However, extensively test their proper functioning in the new environment.

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

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/bpc/c4staff.html

7.4.6 Microflows and long running processes

Special considerations may apply if microflows or long running processes are part of applications to be migrated from Server Foundation to Process Server.

We discuss approaches to migrate long running processes in 5.3, "Migration approaches" on page 79.

From a runtime environment perspective, make sure that no more new process instances for migrated processes can be started in the old environment. You can do this by stopping the process templates, as described in the Server Foundation Info Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.wasee.doc/info/ee/workflow/tasks/t4adft_bpel.html

Any failures during the execution of process instances, which were migrated, must be cleaned up in the old environment to ensure that no outstanding work is left on Server Foundation.

If your processes are using the Compensation feature, clean up any failed compensations on Server Foundation to finish all business-related work on the old environment. You can repair the compensation actions that are in error in the Business Process Choreographer Web Client. See the Server Foundation Info Center for more information about this action:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=
/com.ibm.wasee.doc/info/ee/wfclient/tasks/t7comp.html
```

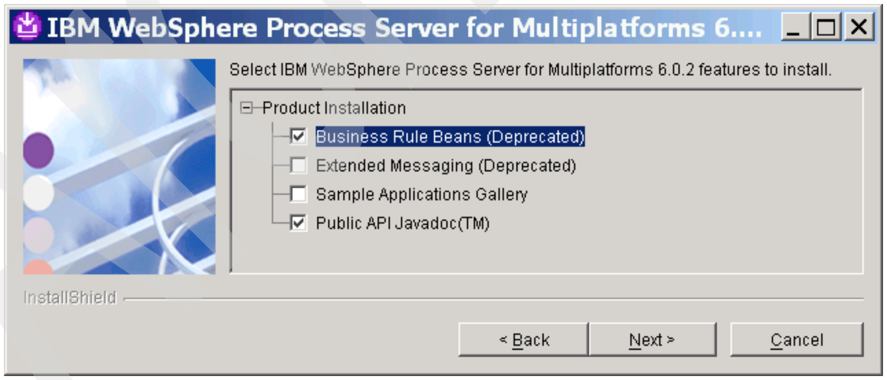
If long running process instances appear to be hanging, look at whether there are messages in the BPERetQueue or BPEHldQueue. Replay these messages and make sure that no messages that concern the migrated processes are in theses queues after replay. More information about replaying failed messages is in the Server Foundation Infocenter:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=
/com.ibm.wasee.doc/info/ee/workflow/concepts/c7replay.html
```

7.5 Migrating Business Rule Beans

Business Rule Beans are a deprecated feature in Process Server. You can still use them, but we recommend that you use the new Business Rules capabilities that are provided in WebSphere Process Server.

To run business rule beans applications from WebSphere Business Integration Server Foundation, perform a custom installation of Process Server. During the custom installation, enable the installation of the business rule beans components, as shown in Figure 7-8.


IBM WebSphere Process Server for Multiplatforms 6....

Select IBM WebSphere Process Server for Multiplatforms 6.0.2 features to install.

Product Installation
- [✓] Business Rule Beans (Deprecated)
- [ ] Extended Messaging (Deprecated)
- [ ] Sample Applications Gallery
- [✓] Public API Javadoc(TM)

Figure 7-8 Selecting Business Rules Beans during Process Server installation

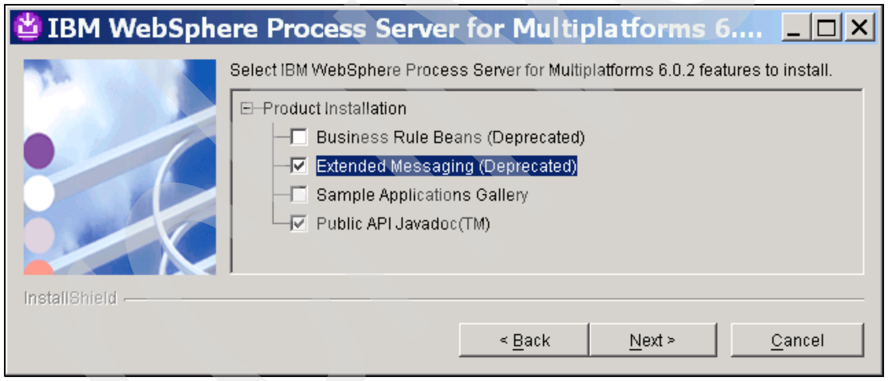
Process Server supports binary compatibility for business rule beans, which means that you can install directly to Process Server your ear file that you used in Server Foundation without going through a source artifact migration before hand.

7.6 Migrating Extended Messaging

The Extended Messaging (CMM) feature of Server Foundation is deprecated in Process Server, but your applications can still use this feature in Process Server. However, you must make changes in the source artifacts as described in 6.14, “Migrating Extended Messaging applications” on page 138.

To enable Process Server to run the migrated application, perform a custom installation of WebSphere Process Server. During the custom installation, enable the installation of the deprecated Extended Messaging feature, as shown in Figure 7-9.

Make sure that this feature is enabled when you run the Extended Messaging applications in the Process Server environment.


IBM WebSphere Process Server for Multiplatforms 6....

Select IBM WebSphere Process Server for Multiplatforms 6.0.2 features to install.

Product Installation
- [ ] Business Rule Beans (Deprecated)
- [✓] Extended Messaging (Deprecated)
- [ ] Sample Applications Gallery
- [✓] Public API Javadoc(TM)

Figure 7-9 Selecting Extended Messaging during Process Server installation

7.7 Migrating the Common Event Infrastructure

The Common Event Infrastructure (CEI) 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 (CBE) from different event sources and provides them to different event consumer applications.

Figure 7-10 on page 167 illustrates the capabilities of CEI.

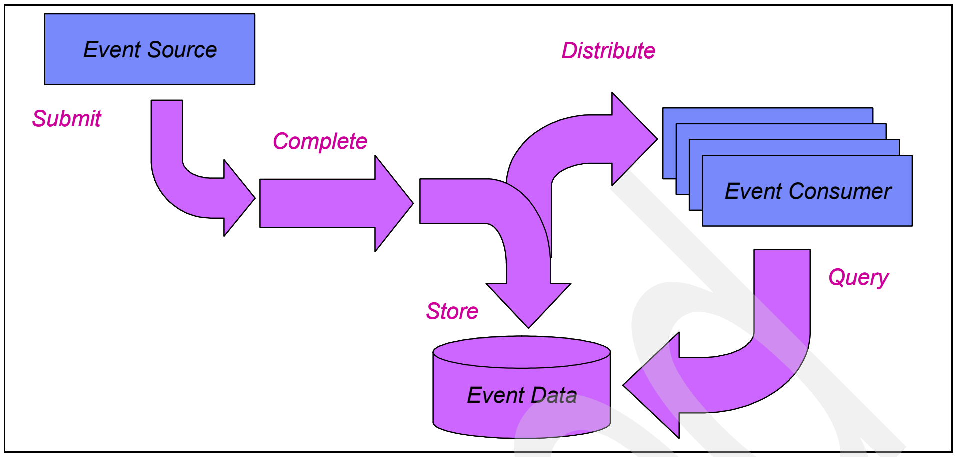


Figure 7-10 Capabilities of Common Event Infrastructure

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

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

Figure 7-11 on page 168 illustrates the architecture of the CEI.

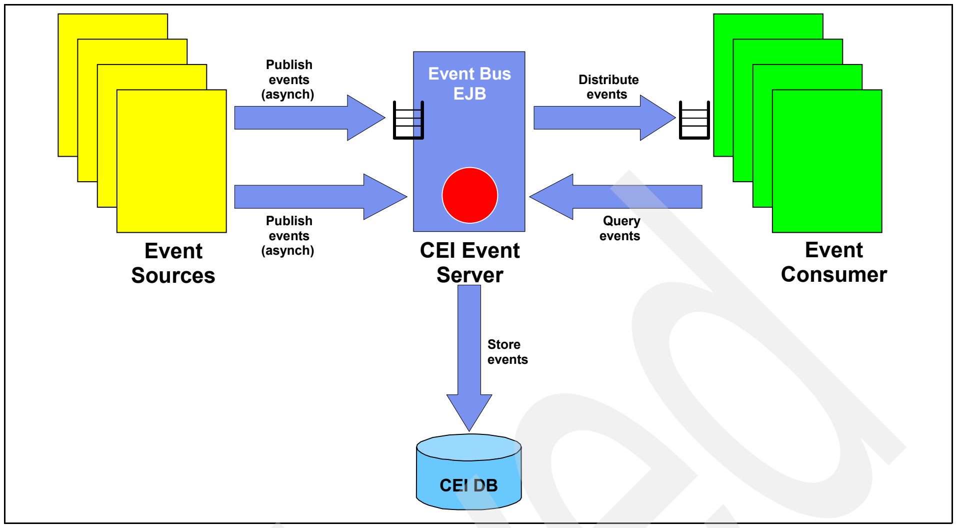


Figure 7-11 Architecture of the Common Event Infrastructure

CEI is also a part of WebSphere Process Server. The APIs of CEI stayed the same. So applications that used these APIs in Server Foundation need no change during a migration to Process Server.

However, it is not possible to preserve instance data from Server Foundation. The content of the Server Foundation CEI database cannot be moved to a Process Server CEI database. It is also not possible to preserve the configuration data for the CEI setup.

8

Best practices

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

We discuss the following topics:

- 8.1, “Planning” on page 170
- 8.2, “Artifacts preparation” on page 170
- 8.3, “Post migration” on page 179
- 8.4, “Non-functional considerations” on page 182

8.1 Planning

The following list summarizes the best practices for planning:

- Document the system, process design before migration 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 Process Server production environment and migrate processes to this environment.
- Run production pilot in parallel with your Server Foundation environment.

  Pilot impacts only a limited number of users and helps to fine-tune your application in the production environment. Pilot allows for seamless rollback.
- Perform exhaustive functional, integration, and performance tests.

  This approach helps to avoid many production problems.
- Do not forget to back up your environment, and be prepared to do a rollback. In case of serious problems, rollback to your original environment.
- Assess non functional metrics of your old environment, and define a new Service Level Agreement (SLA).

  The Service Level Agreement 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 non functional requirements and the 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.

8.2 Artifacts preparation

In this section, we discuss the artifact preparation best practices for a migration project.

8.2.1 WSADIE artifact preparation

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

- Follow the recommendations for Web Services Description Language (WSDL) files in 8.2, “Artifacts preparation” on page 170.
- Follow the naming conventions that are defined for the project. We provide some recommendations in 8.2.8, “Naming convention” on page 175.
- The Migration wizard in WebSphere Integration Developer cannot manage a whole workspace like the modeler Migration wizard. Define each project to migrate. Prepare it with a rebuild and cleaning process. Leave only the artifacts that you really need to migrate.
- Change the namespaces to avoid name conflicts. Follow the recommendations for XSD files in 8.2, “Artifacts preparation” on page 170.
- Assure that the reference of all artifacts is correct. Define the shared components.
- Follow the recommendations for BPEL files in 8.2, “Artifacts preparation” on page 170.
- Do not migrate the generated code.

8.2.2 Modeler artifact preparation

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

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

8.2.3 Schema XSD files

The following list summarizes the best practices for the schemas:

- In WebSphere Process Server V6, two different WSDL/XSD definitions that have the same name and target namespace are not allowed, even in Server Foundation this situation is possible.
- When the XSDs with the same name and namespaces are referring to the same entity, leave only one definition, and delete the copies. Clean and rebuild the project. Solve the reference problems pointing to the remaining type.

- If the XSD refers 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 conflicts in the namespace, introduce the imports in import chaining. Only one import for each namespace per WSDL file.
- Be specific. Where possible, avoid xsd complexTypes 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 8-1 is an example of an anonymous reference.

Table 8-1 Anonymous reference example

Anonymous reference to avoid	**Complete definition with name and namespace**

`<schema`
`attributeFormDefault="qualified"`
`elementFormDefault="unqualified"`
`targetNamespace="http://booking.b`
`pe.samples.websphere.ibm.com"`
`xmlns="http://www.w3.org/2001/XML`
`Schema"`
`xmlns:tns="http://booking.itso.ib`
`m.com">`
`<element name="CustomerElement">`
`<complexType>`
`<all>`
`<element name="id"`
`type="string"/>`
`<element name="name"`
`type="string"/>`
`<element name="address"`
`type="string"/>`
`</all>`
`</complexType>`
`</element>`
`</schema>`
	`<schema`
	`attributeFormDefault="qualified"`
	`elementFormDefault="unqualified"`
	`targetNamespace="http://booking.b`
	`pe.samples.websphere.ibm.com"`
	`xmlns="http://www.w3.org/2001/XML`
	`Schema"`
	`xmlns:tns="http://booking.itso.ib`
	`m.com">`
	`<complexType name="Customer">`
	`<all>`
	`<element name="id"`
	`type="string"/>`
	`<element name="name"`
	`type="string"/>`
	`<element name="address"`
	`type="string"/>`
	`</all>`
	`</complexType>`
	`<element name="CustomerElement"`
	`type="tns:Customer"/>`
	`</schema>`

- In WebSphere Process Server V6, two different wsdl/xsd definitions that have the same name and target namespace are not allowed. Another possibility to this situation is in the generated artifacts:
  - For any wsdl and xsd that gets generated from an EJB or Java bean, 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 -namespace **`Java2WSDL`** command line option to ensure that the namespace of the generated files is unique for the given class.

8.2.4 WSDL files and Web services

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

- Try to follow the Web Services Interoperability specification (WS-I).
- Define only one part per wsdl message. This decision obeys the WS-I, and it is the WebSphere Process Server V6 preferred style.
- When possible, use the Document/literal style for the WSDL file. This is the preferred style in WebSphere Process Server V6.
- 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 using IBM Web services (as opposed to Apache SOAP/HTTP) because IBM Web services are directly supported in WebSphere Process Server V6 and Apache Web services are not.
- There are two ways to organize wsdl and xsd files in Integration Edition V5.1 to minimize the amount of reorganizing during migration. In WebSphere Process Server V6, shared artifacts, such as wsdl and xsd files, must be located in Business Integration projects (Business Integration modules and libraries) in order to be referenced by a 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 Business Integration Library with the same name as the 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/xsd files that a service project references in the service project itself. Integration Edition service projects IDs get migrated to a Business Integration module in WebSphere Integration Developer and a module cannot have dependencies on other modules (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 complexTypes 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/EJB skeleton that gets generated from the WSDL PortTypes/Messages is dependent on Web service Invocation Framework 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 SCA platform. Also avoid creating message types whose high-level element is an array type (maxOccurs attribute is greater than one) because this type of interface is not natively supported in the SCA programming model.

8.2.5 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 the a 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 Integration Edition V5.1 (It is a must to use the wsdl or XML editor to modify the XSLT that are 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 Web service Invocation Framework metadata APIs if possible.

8.2.6 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. This is not supported in WebSphere Process Server. Avoid using the

Business Process Choreographer (BPC) Generic Messaging API (Generic 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

- Prefer to invoke the generic Choreographer API as opposed to invoking the generated session beans that are specific to a particular version of a process (including the valid-from string).
- In WebSphere Process Server, each operation replay has only one client setting. If you have a business process with multiple replies for the same operation, ensure that if any of them has client settings that all replies for that operation have the same client setting.

8.2.7 Modeling migration best practices

The following list summarizes the best practices for the modeling migration:

- Use the last fix pack available for each modeler tool. In some un-fixed versions, the error view can show errors that are not synchronized with the model.
- It is faster to migrate the whole workspace than it is to migrate each project.
- Before you migrate a workspace, delete the not used projects.
- Do not use special characters.
- Open, check, and clean each model before the migration process.
- 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 disc space *before* you start a workspace migration. WebSphere Business Modeler makes an additional backup of the workspace.

8.2.8 Naming convention

The following list presents some recommendations about name conventions to follow in your project:

- 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. Try to install the products in a folder with short path and do not use names that are too long.
- Do not use acronyms and abbreviations unless the abbreviation is widely used.

- Follow the java naming convention. First letter of a word and first letter of each internal word capitalized.
- For the artifacts, use English characters. These names generate code artifacts and sometimes they are mapped removing spaces and changing the capitalization.
- Be consistent in naming the artifacts. It could be useful to use a suffix in each artifact type, for example:
  - Libraries: *Name* + “Lib”
  - Interfaces: *Name* + “Interface”
  - SCA components:
    - Human Task: *Name* + “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:

  ```
  http://modulename/interfacename/project
  ```

8.2.9 Modeler naming convention

Some recommendations for modeler naming:

- Be consistent. A model project can use many activities. Define the convention for elements, and follow this convention in all the activities, business items, input, and outputs. Also, use a convention for the metrics and KPIs (not covered in this book).
- Avoid non-English characters. Use the first letter of each word in upper case. Do not use very long activity names.
- Some default value names of modeler are not the best name for some entities. Consider the possibility of changing each element name and defining a more appropriated name, for example, the default “Input Criteria” and “Output Criteria” name used in each activity is used to name the operations and the methods in Java implementation.
- The names of the generated artifacts are a translation of the names in modeler. Try the translated names, and follow the naming convention. Table 8-2 on page 177 shows some translations that are performed in the code generation.

Table 8-2 Example name mapping from Modeler to WSADIE

Artifact in Business Modeler		**Artifact in WSADIE**		**Recommend**
Type	Example	Type	Example	
Catalog name	`Business items/data`	translated to package name	`Businessitems.data`	Do not use several levels, for example, com.ibm.itso.project It works for Server Foundation, but not to migrate to Process Server
Business item name	`Booking order`	Class name	`BookingOrder`	The translator suppresses the spaces. User uppercase in the first letter of each word.
Process catalog name	`data`	Process folder	`\data\`	Starts with lowercase.

Process name	`Booking order`	BPEL and WSDL name	`BookingOrder.bpel`
			`BookingOrder.component`
			`BookingOrder.wsdl`
			`BookingOrderInterface.wsdl`
				User uppercase in the first letter of each word.

Activity not Human Task	`Reservation System`	Service name	`ReservationSystemPT.java`
			`ReservationSystemPTJavaBinding.wsdl`
			`ReservationSystemPTJavaService.wsdl`
				Use uppercase in the first letter of each word.

Input and Output logic	`Input Criteria`
	`Output Criteria`
		Operation in the Interface	`Input Criteria`
			`Output Criteria`
				Consider defining a verb as a name.

Input and Output logic	`Input Criteria`
	`Output Criteria`
		Method Name in Java service implementation	`SendReservationSystem_InputCriteria`
			`InputCriteriaMessageMessage.java`
				Consider defining a verb as a name.

The mapping to WebSphere Integration Developer from modeler artifacts is different, as shown in Table 8-3 on page 178.

Table 8-3 Example name mapping to WebSphere Integration Developer

Artifact in Business Modeler		**Artifact in WSADIE**		**Recommend**
Type	Example	Type	Example	
Catalog name	`Business items/data`	A folder in the Library project	`businessitems/data`	Do not use several levels.
Catalog name		The XSD file	`data.xsd`	
Catalog name	`Business items/data`	The namespace definition	`<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" … xmlns:tns="http://ProjectName/Businessitems/data "  …/>`	
Business item name	`Customer`	elements in the xsd file	`<xsd:complexType name="Customer">`	Starts with lowercase.
Process catalog name	`data`	folders	`data`	Starts with lowercase. Use uppercase in the first letter of each word.

Process name	`Booking order`	Folder Name and BPEL artifacts	`data/BookingOrder`
			`BookingOrder.bpel`
			`BookingOrder_01205027910.component`
			`BookingOrder_01205027910.export`
				Starts with lowercase. Use uppercase in the first letter of each word.

Human Task Activity	`Booking Order Form`	Inline tel file (`itel file`)	`BookingOrderForm_1872960898.itel`	Starts with lowercase. Use uppercase in the first letter of each word.
Java Service Activity	`Reservation System`	java file	`ReservationSystem_01098838552Impl.java`	Starts with lowercase. Use uppercase in the first letter of each word.

Input and Output logic	`Input Criteria`
	`Output Criteria`
		Operation name in BPEL	`<bpws:invoke name="BookingOrderForm_InputCriteria" operation="InputCriteria" …>`	Consider defining a verb as the name.

Artifact in Business Modeler		Artifact in WSADIE		Recommend
Type	Example	Type	Example	
Input and Output logic	`Input Criteria Output Criteria`	WSDL Message Name	`in BookingOrderInterface.wsdl <wsdl:portType name="BookingOrder"> <wsdl:operation name="InputCriteria"> <wsdl:input message="tns:InputCriteria" name="InputCriteriaRequest"/> <wsdl:output message="tns:OutputCriteria" name="InputCriteriaResponse"/ > </wsdl:operation> </wsdl:portType>`	
Input and Output logic	`Input Criteria Output Criteria`	Method in Java service	`public DataObject InputCriteria(DataObject reservationInput)`	

8.3 Post migration

In the following sections, we explain some best practices for post migration steps of migrated artifacts.

8.3.1 Refactoring processes

Consider refactoring and tuning business process after migration because the original process is not always optimal for WebSphere Process Server.

The following list summarizes the best practices for refactoring processes:

- Consider optimizing the generated 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 as SCA components and deploy them together with the process.
- Consider refactoring the process to utilize the Process Server Business Rules engine.

  If the process is using if-then rules' logic and they can be externalized, and modifying processes in run time without redeployment is required, then the rule engine can simplify process complexity.
- Consider the refactoring process to utilize the 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 very linear, use a business process; however, if it is primarily event-driven and contains cyclical patterns, use a state machine. Both editors and the languages which they are modeled are equally valid.
- Be as specific in staff assignments as possible.

  The more users you assigned to a task, the more you need to update the work lists.
- Choose between inline and standalone 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 either be implemented directly in the process using a human task activity or as a property of an invoke, pick, receive, event handler, or on message activity. Model tasks as inline task 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 a stand-alone task if any of the following conditions are present:
    - Information from the business process are not necessary.
    - The task provides just another service.

- Choose between microflows and macroflows. Choose the microflow process by default because it is simpler and is executed much faster than macroflow. Choose microflow instead of macroflow if some of the following is valid:
  - Process requires more than one transaction
  - Process need to stop at any point and wait for external input, either in the form of an event or a human task
  - 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.

8.3.2 Further development

The following list summarizes the best practices for further development and customization for WebSphere Process Server artifacts:

- Use WebSphere Integration Developer to manually complete the migration. This involves fixing any Java code that could not be automatically migrated, and verifying the wiring of the migrated artifacts.
- Use the development tooling to edit integration artifacts.

  For creating, configuring, and modifying 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 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 J2SE environment.
  - In EJB, as always, do not perform any functions that may 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 being 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 may be serialized at runtime 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 for creating and modifying 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.

8.4 Non-functional considerations

Tune your process after migration. Use the following hints:

- 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 get 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 modelling a business process with a single transaction, consider making it a microflow. Microflows have great performance and run very quickly in the runtime environment.

- Disable monitoring and tracing whenever possible.

  Monitoring and tracing is 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 process server database.

  WebSphere Process Server is installed by default using Cloudscape; 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 macroflows (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 datasource's cache, which results 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 under an activation specification of the Platform Messaging Component SPI Resource Adapter. Its custom property, maxConcurrency, under the 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 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.

  Refer to 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 maxed out, it sometimes is caused by this setting being too low, so increase it.

– Configure the Object Request Broker (ORB) thread pool.

  This configuration parameter is relevant if the cluster is driven by a driver node through the SCA synchronous binding. Due to the interaction between synchronous SCA, workload manager, and the ORB, the ORB thread pool size on the cluster nodes needs to be configured to maximize the clustering throughput.

  The rule-of-thumb is to use the same number of ORB threads on all application nodes, and have 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 6-node cluster should be 20.

– Configure relationship and BPE 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.

9

Technical scenarios

In this chapter, we discuss the following:

- 9.1, “Overview of scenarios” on page 186
- 9.2, “Hardware and software environment setup” on page 186
- Two working technical oceanariums that demonstrate migration of various artifacts:
  - 9.3, “Travel Operations scenario” on page 188
  - 9.4, “Travel Booking scenario” on page 246

9.1 Overview of scenarios

In this chapter, we discuss the technical scenarios that we use in this. These scenarios demonstrate the simple migration of WebSphere Business Integration Server Foundation solutions using the migration support that WebSphere Process Server provides. We also show testing and verification of the migrated application.

In the first scenario, we show you how to migrate a WebSphere Business Integration Server Foundation solution starting from artifacts in WebSphere Studio Application Developer Integration Edition.

In the second scenario, we focus on how to migrate starting from artifacts in WebSphere Business Integration Modeler.

You can download all of the material that we reference. See Appendix A, “Additional material” on page 285 for more information.

9.2 Hardware and software environment setup

In this section, we describe the hardware and software that are used in a test environment for the scenarios that we demonstrate in this book. It represents the hardware system, which meets the minimum recommended requirements. The hardware and software environment setup are sufficient for the technical scenarios that we used in this book and does not represent an actual production environment.

Figure 9-1 on page 187 shows the physical topology that we used in the technical scenarios.

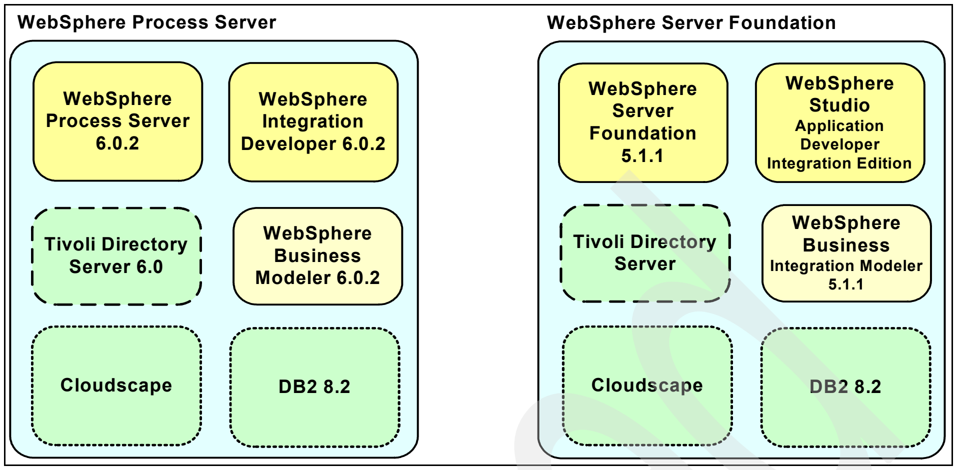

WebSphere Process Server
WebSphere Process Server 6.0.2
WebSphere Integration Developer 6.0.2
Tivoli Directory Server 6.0
WebSphere Business Modeler 6.0.2
Cloudscape
DB2 8.2
WebSphere Server Foundation
WebSphere Server Foundation 5.1.1
WebSphere Studio Application Developer Integration Edition
Tivoli Directory Server
WebSphere Business Integration Modeler 5.1.1
Cloudscape
DB2 8.2

Figure 9-1 Physical topology

Table 9-1 shows the hardware configuration that we used.

Table 9-1 Hardware environment

Hardware	Machine
Machine type	IBM NetVista
CPU	Intel® Pentium® 4 1.80 GHz
Memory	1.5GB
Hard disk	40GB
Operating System	Windows XP Professional Service Pack 2

Table 9-2 on page 188 shows the list of products that we installed.

Table 9-2 Software versions used in the scenarios

Software	Version
WebSphere Process Server	6.0.2
WebSphere Business Integration Server Foundation	5.1.1
WebSphere Integration Developer	6.0.2
WebSphere Studio Application Developer Integration Edition	5.1.1
WebSphere Business Modeler	6.0.2
WebSphere Business Integration Modeler	5.1.1
DB2® Universal Database™	8.2
Microsoft® Windows XP Professional	SP 2

9.3 Travel Operations scenario

The Travel Operations scenario discusses 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 Integration Edition, is migrated to an Integration Developer project and then deployed and run in WebSphere Process Server.

Figure 9-2 on page 189 shows the migration process.

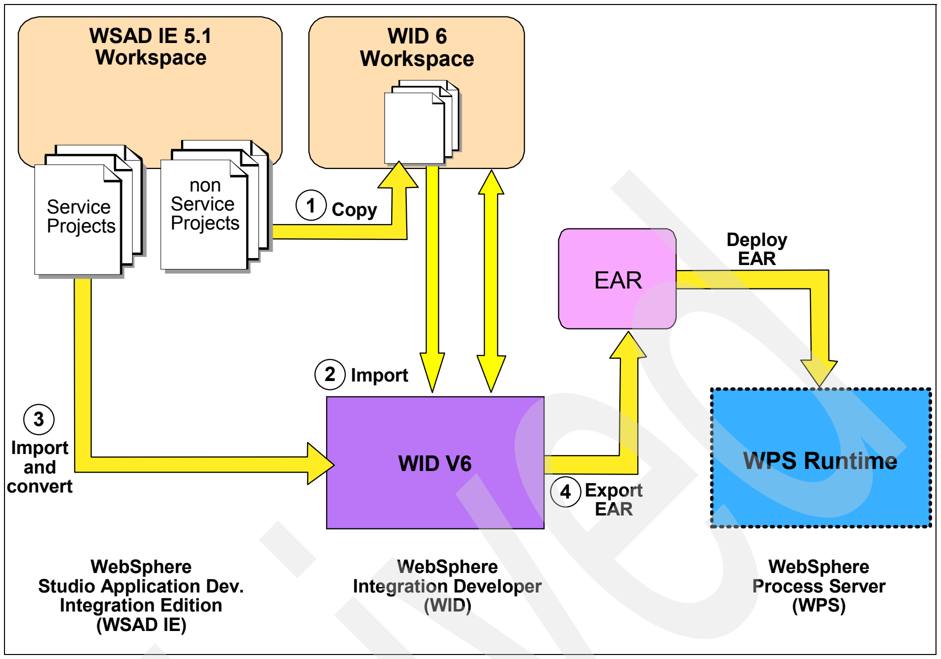


Figure 9-2 Migration process to WebSphere Process Server

9.3.1 Overview

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.

The approval request is reviewed within a staff activity. If approved, the Travel Operations process calls another process, the TravelAgency process, to book the trip. If not approved, than it calls an EJB service to send an email to the employee informing about the disapproval.

The TravelAgency process invokes two Java services, one to reserve a hotel, one to reserve a flight and sends a booking confirmation email to the employee.

9.3.2 Preparation

In this section, we discuss the initial steps that must be completed to migrate the Travel Operations scenario to WebSphere Process Server.

Examine the Integration Edition projects

The WebSphere Studio Application Developer Integration Edition Project Interchange file *WSADIEV511_PI_StartingPointForTechnicalScenario1.zip* is in the attached zip files. Import the artifacts into WebSphere Studio Application Developer Integration Edition.

The Server Foundation application to be migrated consists of two service projects, one Java project and an enterprise application:

- TravelOperations service project inherits:
  - One long running process
  - One staff activity
  - EJB process binding to TravelAgency process
  - EJB binding to EmailSender Session Bean
- TravelAgency service project inherits:
  - One microflow
  - Two Java services
  - Java bindings to these Java services
  - EJB binding to EmailSender Session Bean
- EmailSender EJB project:
  - Stateless Session Bean that sends an email when invoked

> **Note:** In this example, a Session Bean is going to send emails. Therefore, have a proper SMTP server available. In this scenario, the host *relay.de.ibm.com* is set in the TravelOperationsProcess and TravelAgencyProcess. To send an email when you recreate the scenario, make sure set a valid host name in the processes in your environment.
>
> If you do not use a valid host, the scenario still runs through properly; however, no email is received.

- EmailSenderClient project:
  - Java project functioning as EJB client for EmailSender EJB project
- EmailSenderEAR EAR project:
  - Enterprise Application project containing EmailSender and EmailSenderClient projects

- CommonArtifacts Java project:
  - This Java project inherits the .wsdl files for the EmailSender Session Bean. These files are used by the processes in projects TravelAgency and TravelOperations to call the EmailSender bean.

Figure 9-3 shows the module structure in WebSphere Studio Application Developer Integration Edition.

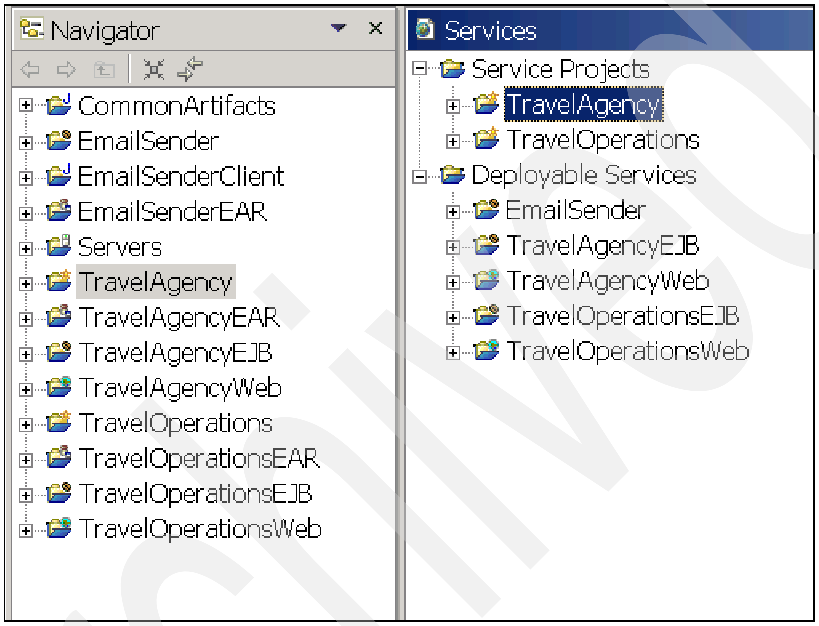
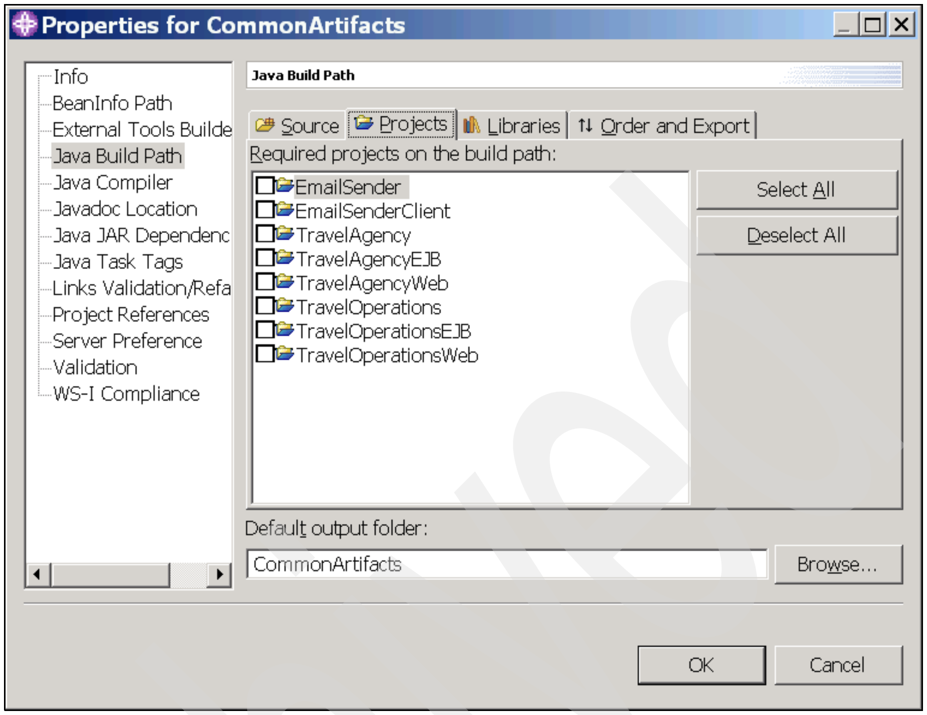


Figure 9-3 Services and Navigator views on the projects to migrate

Examine dependencies

In preparation for the migration, clarify all of the dependencies between the projects to find the sequence in which the projects are migrated to Integration Developer:

1. In WebSphere Studio Application Developer Integration Edition, go to the **Navigator** view, and right-click project CommonArtifacts.
2. Click **Properties** → **Project References** or **Java Build Path**. Here you see the dependencies for project CommonArtifacts, as shown in Figure 9-4 on page 192. The CommonArtifacts project does not have any dependencies on other projects.



Required projects on the build path:

- [ ] EmailSender
- [ ] EmailSenderClient
- [ ] TravelAgency
- [ ] TravelAgencyEJB
- [ ] TravelAgencyWeb
- [ ] TravelOperations
- [ ] TravelOperationsEJB
- [ ] TravelOperationsWeb

Default output folder: CommonArtifacts

Figure 9-4 Java Build Path of CommonArtifacts project

3. Repeat this step for the other projects, and create a list of dependencies. For the service projects, it is sufficient to only look at the service project itself because the corresponding EJB and EAR projects are generated.

Fill in the dependencies as shown in Table 9-3.

Table 9-3 Project dependencies

Project name	**Required projects**
CommonArtifacts	
EmailSenderClient	
EmailSender	EmailSenderClient
EmailSenderEAR	EmailSenderClient EmailSender
TravelAgency	CommonArtifacts EmailSenderClient

Project name	Required projects
TravelOperations	CommonArtifacts EmailSenderClient TravelAgency

Starting with projects that have no dependencies, going to the ones with plenty of dependencies is the order in which the projects forming the Integration Edition application must be migrated.

The table shows that it does not matter whether you begin with the EmailSender, EmailSenderEJB, and EmailSenderEAR projects or with the CommonArtifacts project. However, to migrate TravelAgency and TravelOperations, they must both be present in Integration Developer.

Table 9-3 on page 192 also shows that the service project TravelOperations has a dependency on another service project, the TravelAgency project. You must resolve dependencies between service projects before migration to Integration Developer.

Resolve dependencies between service projects

To fully migrate the .bpel files within a service project, you must ensure that all of the .wsdl and .xsd files that the .bpel files reference can be resolved in a business integration project in the new 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, refer to 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 you are migrating, reorganize the 5.1 artifacts using WebSphere Studio Application Developer Integration Edition prior to migration because Business Integration module projects cannot share artifacts.

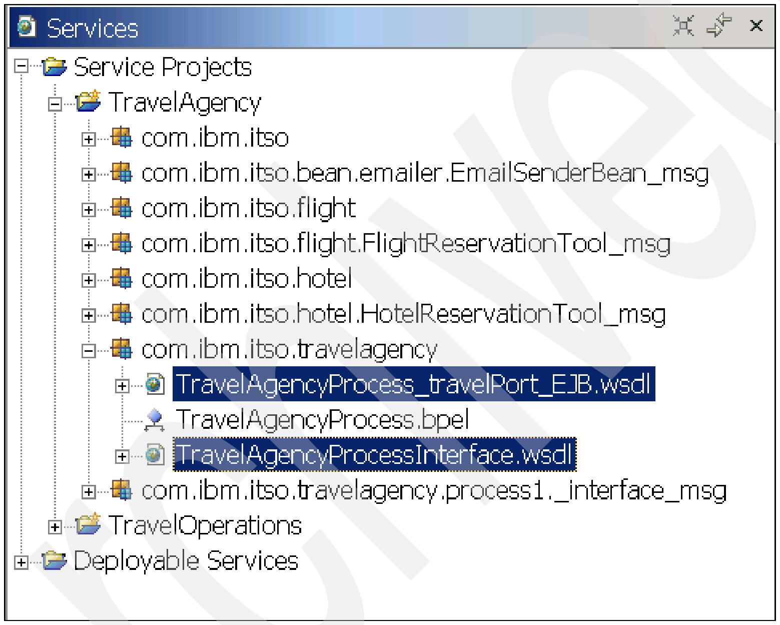
  In this scenario, the TravelOperationsProcess from the TravelOperations service project refers to the .wsdl file of the TravelAgencyProcess in the TravelAgency service project, which you must reorganize.

  As an example, you select the option of reorganizing the 5.1 artifacts in WebSphere Studio Application Developer Integration Edition by placing all of the common .xsd and .wsdl artifacts in a Java project and adding a dependency on this Java project to all service projects that use these common artifacts. In WebSphere Integration Developer, create a new

Business Integration Library project with the same name as the 5.1 shared Java project before you migrate any of the service projects.

There is already the Java project CommonArtifacts, which store the .wsdl files for the EmailSender EJB service. This project resolves the dependency between the TravelOperations and the TravelAgency service projects. There are two .wsdl files within the TravelAgency project on which the TravelOperations project is dependent. This is the interface of the TravelAgencyProcess, which the TravelOperationsProcess calls through EJB binding:

- TravelAgencyProcessInterface.wsdl
- TravelAgencyProcess_travelPort_EJB.wsdl



```
Services
Service Projects
  TravelAgency
    com.ibm.itso
    com.ibm.itso.bean.emailer.EmailSenderBean_msg
    com.ibm.itso.flight
    com.ibm.itso.flight.FlightReservationTool_msg
    com.ibm.itso.hotel
    com.ibm.itso.hotel.HotelReservationTool_msg
    com.ibm.itso.travelagency
      TravelAgencyProcess_travelPort_EJB.wsdl
      TravelAgencyProcess.bpel
      TravelAgencyProcessInterface.wsdl
    com.ibm.itso.travelagency.process1._interface_msg
  TravelOperations
Deployable Services
```

Figure 9-5 .wsdl files to move in the CommonArtifact Java project

You must move the two files 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 project TravelAgency.
3. Mark the files TravelAgencyProcessInterface.wsdl and TravelAgencyProcess_travelPort_EJB.wsdl. Right-click these files and choose **Move**.

4. In the **Folder Selection** window, browse to the travelagency folder within the CommonArtifacts project, as shown in Figure 9-6. Ensure, that in the Select the destination field, **CommonArtifacts/com/ibm/itso/travelagency** is entered. Click **OK**.

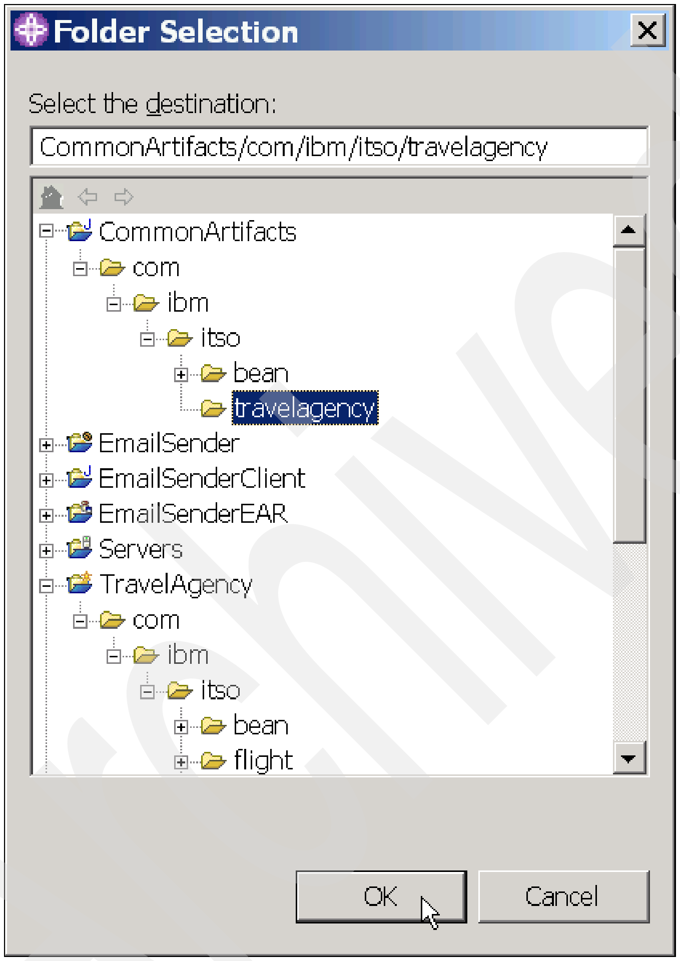


Figure 9-6 Selection dialog for moving the .wsdl files to the CommonArtifact project

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. This is 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 open the tab **Projects**. Clear the TravelAgency project.

   c. Go to **Project References**, and clear the TravelAgency project here too.

   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:

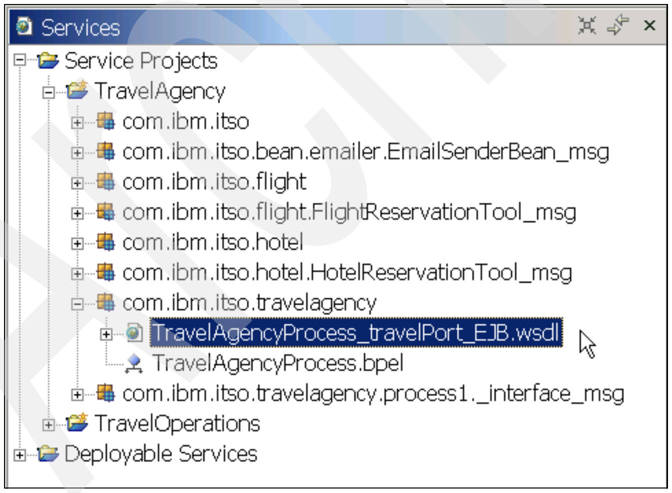
1. Generate Deploy Code for the TravelAgency project.

   a. Open the **Services** view, and go to TravelAgencyProcess.bpel file in TravelAgency project.

   b. Right-click this file, and select **Enterprise Services** → **Generate Deploy Code**.

   c. Ensure that the **Referenced Partners** are assigned, and click **OK**. This starts the generation of deployment code.

2. After the Generate Deploy Code wizard finishes, explore the TravelAgency project. Notice, that the TravelAgencyProcess_travelPort_EJB.wsdl was regenerated, as shown in Figure 9-7.



Services
Service Projects
TravelAgency
com.ibm.itso
com.ibm.itso.bean.emailer.EmailSenderBean_msg
com.ibm.itso.flight
com.ibm.itso.flight.FlightReservationTool_msg
com.ibm.itso.hotel
com.ibm.itso.hotel.HotelReservationTool_msg
com.ibm.itso.travelagency
TravelAgencyProcess_travelPort_EJB.wsdl
TravelAgencyProcess.bpel
com.ibm.itso.travelagency.process1._interface_msg
TravelOperations
Deployable Services

Figure 9-7 Regenerated TravelAgencyProcess_travelPort_EJB.wsdl file

   a. Delete this file from the TravelAgency project. Right-click this file, and select **Delete**. Click **Yes** in the Confirmation dialog.

3. Rebuild all projects by choosing **Project** → **Rebuild all** from the menu.
4. Generate Deploy Code for the TravelOperations project.

   Open the **Services** view and go to TravelOperationsProcess.bpel file in TravelOperations project. Right-click this file and select **Enterprise Services** → **Generate Deploy Code**.

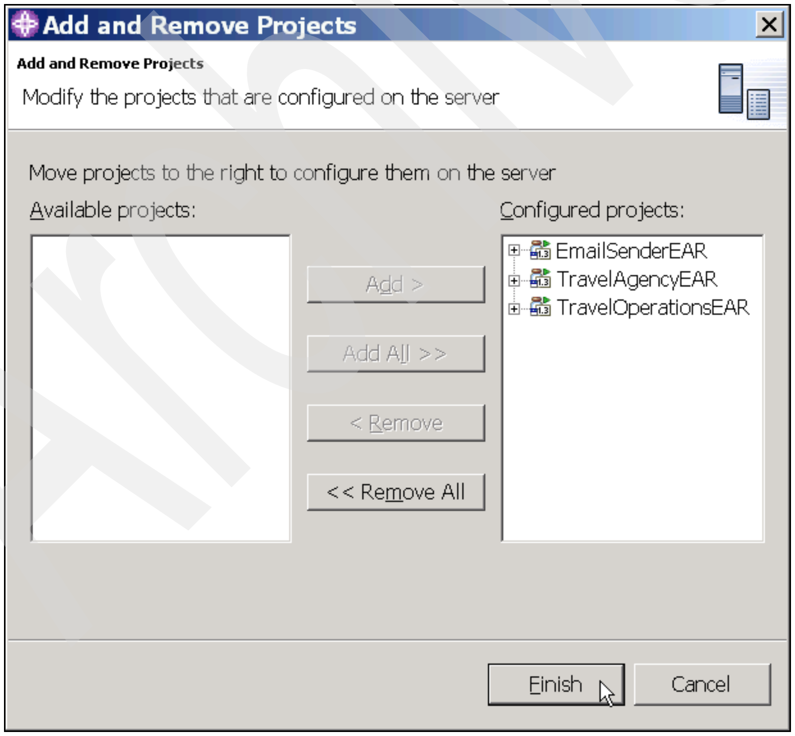
   Ensure that the **Referenced Partners** are assigned.

   Make sure that especially the travelPort is now referring to the TravelAgencyProcess_travelPort_EJB.wsdl file in the CommonArtifacts project.

   Click **OK**. This starts the generation of deployment code.
5. Add the projects EmailSenderEAR, TravelAgencyEAR and TravelOperationsEAR to the Unit Test Environment (UTE) within WebSphere Studio Application Developer Integration Edition.

   In the **Servers** view, right-click the UTE server, and select **Add and remove projects**.

   In the **Add and Remove Projects** dialog, click **Add All**. Click **Finish**.


Add and Remove Projects

Add and Remove Projects

Modify the projects that are configured on the server

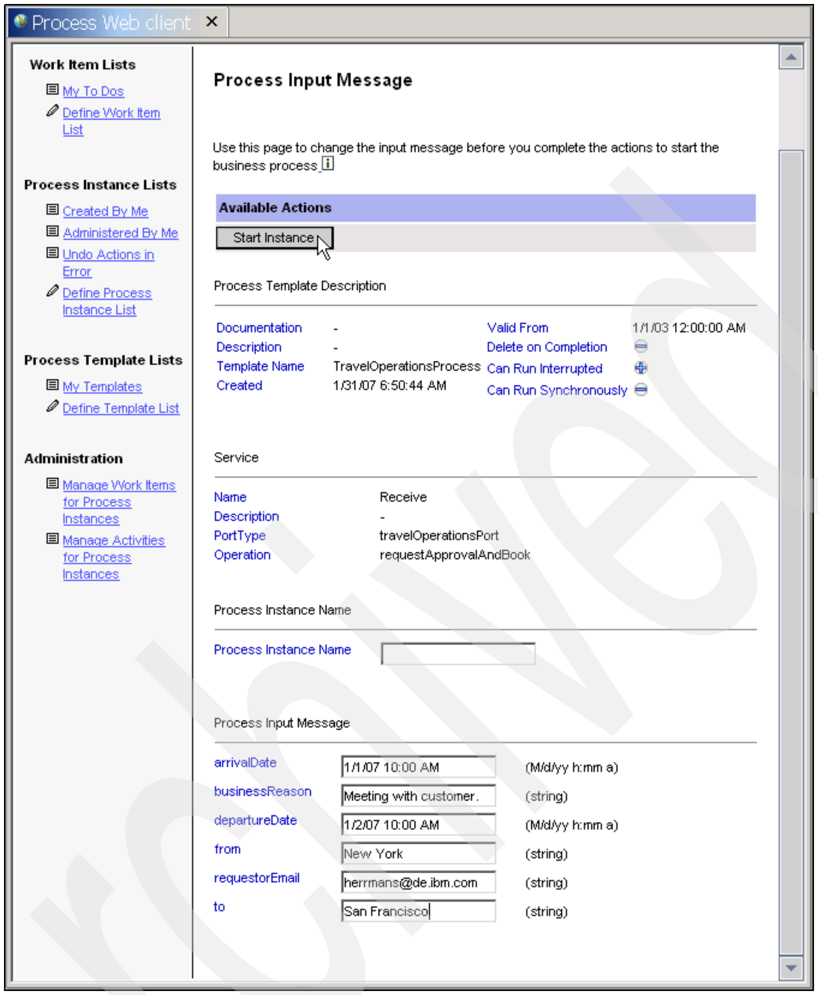
Move projects to the right to configure them on the server

Available projects:

Configured projects:
- EmailSenderEAR
- TravelAgencyEAR
- TravelOperationsEAR

Figure 9-8 Add and Remove Projects dialog

6. In the **Servers** view, right-click the UTE server, and select **Publish**.
7. In the **Servers** view, right-click the UTE server, and select **Create tables and datasources**.
8. Start the UTE server. Right-click he UTE server, and select **Start**.
9. A short function test is performed using the Business Process Choreographer Web Client. After the server starts, right-click the server, and select **Launch Business Process Web Client**. The web client is opened within WebSphere Studio Application Developer Integration Edition.
   a. Start a TravelOperationsProcess (Figure 9-9 on page 199) by selecting **My Templates** from the navigation pane within the **Process Web Client**. Select **TravelOperationsProcess**, and click **Start Instance.**
   b. Complete the **Process Input Message** fields, and click **Start Instance**.


Process Web client

Work Item Lists
- My To Dos
- Define Work Item List

Process Instance Lists
- Created By Me
- Administered By Me
- Undo Actions in Error
- Define Process Instance List

Process Template Lists
- My Templates
- Define Template List

Administration
- Manage Work Items for Process Instances
- Manage Activities for Process Instances

Process Input Message

Use this page to change the input message before you complete the actions to start the business process

Available Actions

Start Instance

Process Template Description

Documentation	-	Valid From	1/1/03 12:00:00 AM
Description	-	Delete on Completion	⊖
Template Name	TravelOperationsProcess	Can Run Interrupted	⊕
Created	1/31/07 6:50:44 AM	Can Run Synchronously	⊖

Service

Name	Receive
Description	-
PortType	travelOperationsPort
Operation	requestApprovalAndBook

Process Instance Name

Process Instance Name	

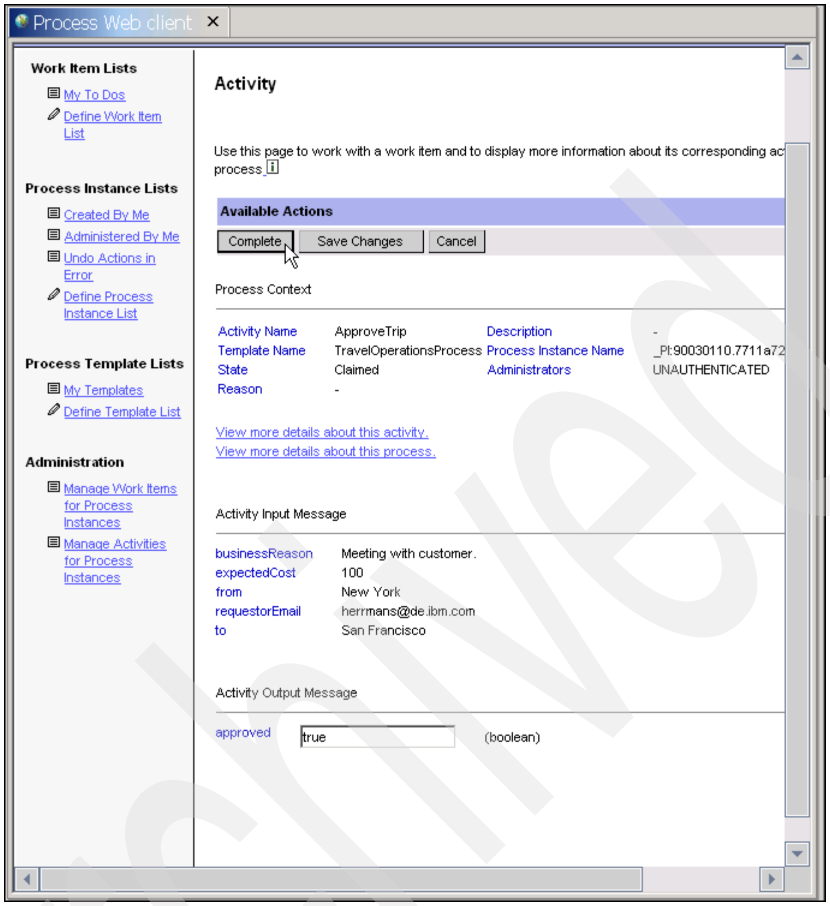
Process Input Message

arrivalDate	1/1/07 10:00 AM	(M/d/yy h:mm a)
businessReason	Meeting with customer.	(string)
departureDate	1/2/07 10:00 AM	(M/d/yy h:mm a)
from	New York	(string)
requestorEmail	herrmans@de.ibm.com	(string)
to	San Francisco	(string)

Figure 9-9 Process Web Client: Start a TravelOperationsProcess

c. A staff activity is created, as shown in Figure 9-10 on page 200. Click **My To Dos** in the Process Web Clients navigation pane. There is one **ApproveTrip** activity listed. Select it, and click **Claim**.

d. Click **ApproveTrip** to complete the activity. The Activity page opens. Enter `true` in the Activity Output Message field, and click **Complete**.



Activity

Use this page to work with a work item and to display more information about its corresponding ac process

Process Context

Activity Name	ApproveTrip	Description	-
Template Name	TravelOperationsProcess	Process Instance Name	_PI:90030110.7711a72
State	Claimed	Administrators	UNAUTHENTICATED
Reason	-		

View more details about this activity.
View more details about this process.

Activity Input Message

businessReason	Meeting with customer.
expectedCost	100
from	New York
requestorEmail	herrmans@de.ibm.com
to	San Francisco

Activity Output Message

approved	true	(boolean)

Figure 9-10 Process Web Client: Complete the staff activity

e. Because you entered `true` as result of the ApproveTrip activity, the TravelAgencyProcess must be invoked. Look at the SystemOut.log of the UTE server, and check whether the processing of the TravelAgencyProcess is successful.

Note: In case the SMTP host is not reachable, the last part of the output is slightly different. However, the scenario finishes in this case also successfully for the reason of simplicity.

Example 9-1 on page 201shows the SystemOut of the TravelAgencyProcess when you set the Approve Trip to `true`.

Example 9-1 SystemOut of the TravelAgencyProcess when 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 of the UTE server shows that the TravelAgencyProcess was invoked successfully.

f. If you redo steps a-e, but enter false as result of the ApproveTrip activity in step d, the TravelAgencyProcess is not invoked. Instead, the TravelOperationsProcess sends a rejection email, as Example 9-2 shows.

Example 9-2 SystemOut for TravelOperationsProcess when 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: ----------------------
```

The verification of the changes in the WebSphere Studio Application Developer Integration Edition projects in order to migrate to WebSphere Integration Developer are finished with this step.

Exporting the projects for migration

To export the projects for migration:

1. Choose **File** → **Export** → **File system** → **Next**.
2. Select CommonArtifacts, EmailSender, EmailSenderClient, EmailSenderEAR, TravelAgency, and TravelOperations.
3. 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 Integration Developer, the **Welcome** page (Figure 9-11 on page 203) is displayed.

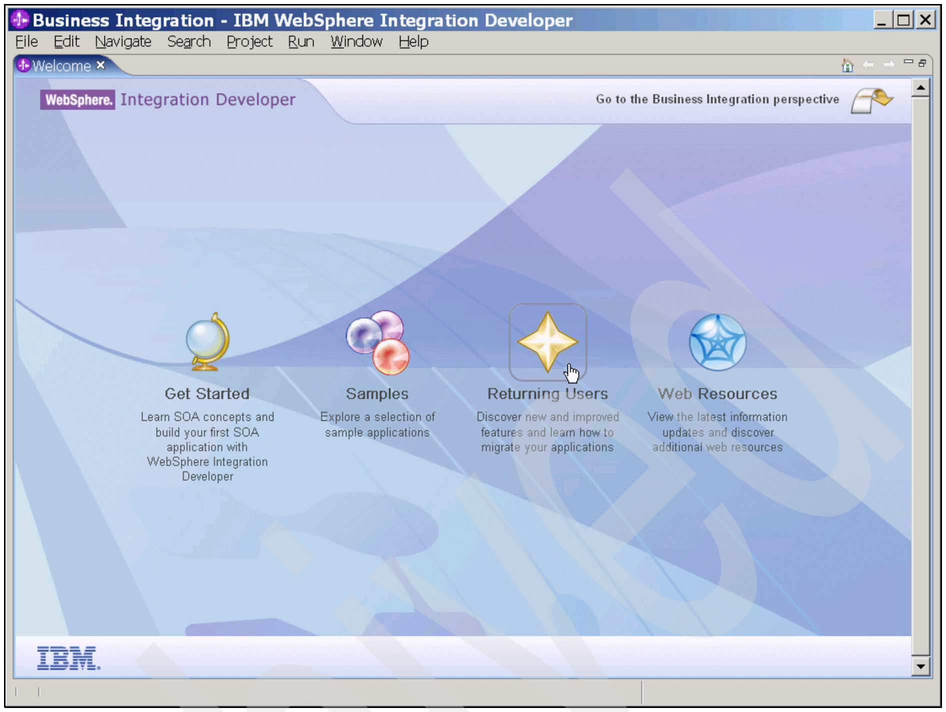
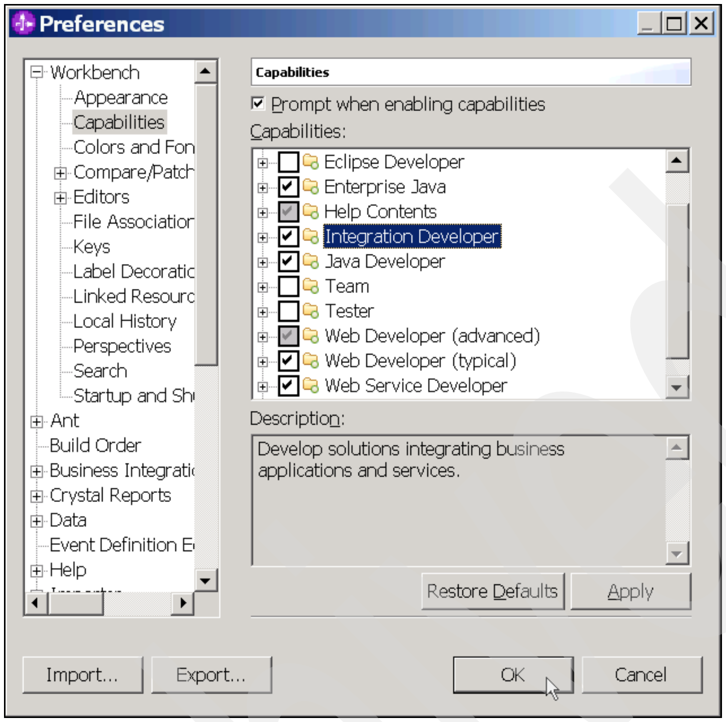

Business Integration - IBM WebSphere Integration Developer
File Edit Navigate Search Project Run Window Help
Welcome
WebSphere. Integration Developer
Go to the Business Integration perspective
Get Started
Learn SOA concepts and build your first SOA application with WebSphere Integration Developer
Samples
Explore a selection of sample applications
Returning Users
Discover new and improved features and learn how to migrate your applications
Web Resources
View the latest information updates and discover additional web resources
IBM

Figure 9-11 WebSphere Integration Developer Welcome page

2. Ensure that all Integration Developer features that are necessary for migration are enabled.
3. In WebSphere Integration Developer, go to **Window** → **Preferences**, as shown in Figure 9-12 on page 204. Go to **Workbench**, and select the category **Capabilities**.
   a. Select all features under the following categories:
      - Advanced J2EE
      - Enterprise Java
      - Integration Developer
      - Java Developer
      - Web Developer (typical)
      - Web Service Developer
      - XML Developer
   b. Click **OK**.



Capabilities

✓ Prompt when enabling capabilities

Capabilities:

- ☐ Eclipse Developer
- ✓ Enterprise Java
- ✓ Help Contents
- ✓ Integration Developer
- ✓ Java Developer
- ☐ Team
- ☐ Tester
- ✓ Web Developer (advanced)
- ✓ Web Developer (typical)
- ✓ Web Service Developer

Description:

Develop solutions integrating business applications and services.

Figure 9-12 Setting Integration Developer Capabilities for migration

4. Disable the automatic build. From the menu, select **Project**, and clear **Build Automatically.**

9.3.3 Implementation

In this section, we provide the detailed steps that are involved in migrating the Travel Operations scenario using WebSphere Integration Developer.

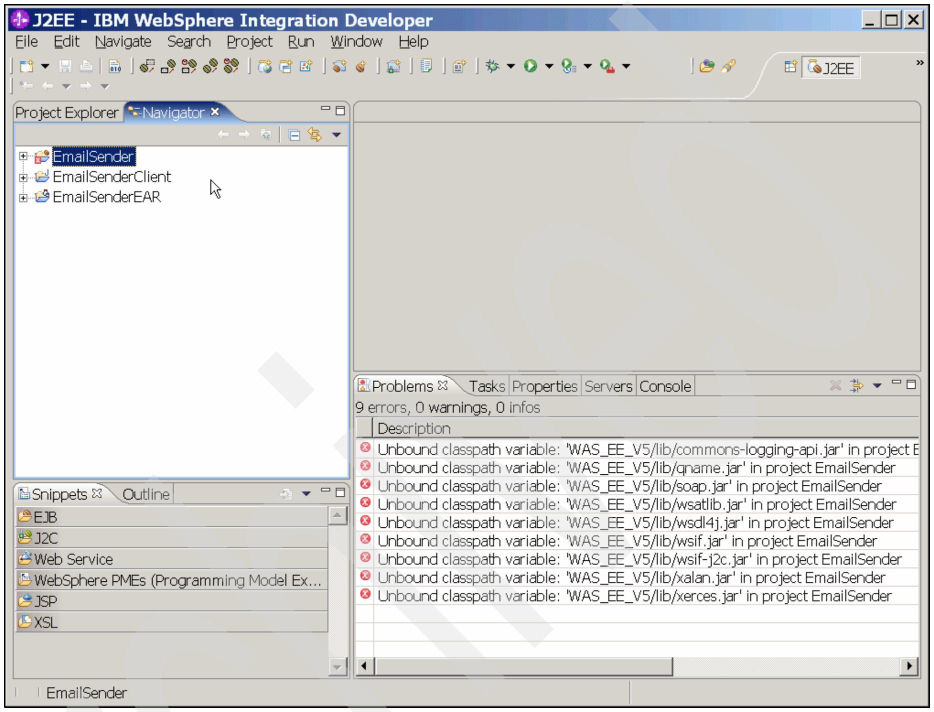
Migrating the J2EE projects

To migrate the J2EE projects:

1. Open the J2EE perspective by selecting **Window** → **Open Perspective** → **Other**. In the Select Perspective dialog, choose **J2EE**, and click **OK**.
2. Copy the projects EmailSenderClient, EmailSender, and EmailSenderEAR into the new workspace directory.
3. Import the projects EmailSenderClient, EmailSender, and EmailSenderEAR by clicking **File** → **Import** → **Existing Project into Workspace**, and select

the projects that were copied over to the new Integration Developer workspace.

Notice that there are three projects in the Integration Developer Navigator view after you complete the steps, as shown in Figure 9-13.



9 errors, 0 warnings, 0 infos

```
Unbound classpath variable: 'WAS_EE_V5/lib/commons-logging-api.jar' in project E
Unbound classpath variable: 'WAS_EE_V5/lib/qname.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/soap.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/wsatlib.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/wsdl4j.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/wsif.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/wsif-j2c.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/xalan.jar' in project EmailSender
Unbound classpath variable: 'WAS_EE_V5/lib/xerces.jar' in project EmailSender
```

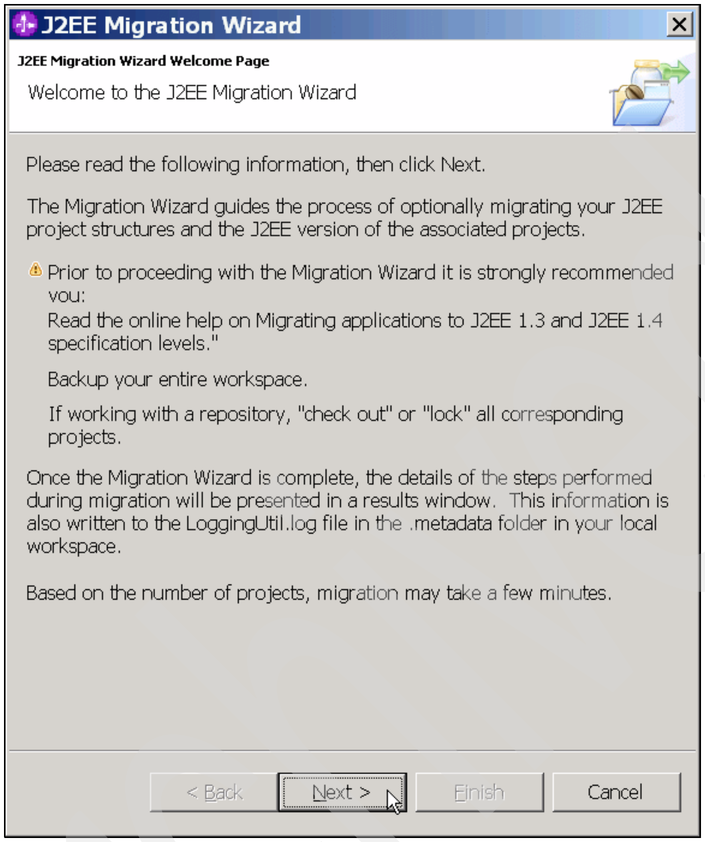
Figure 9-13 EmailSender projects import into Integration Developer workspace

These projects belong to a Version 1.3 J2EE application, and because of this you must migrate them to J2EE 1.4 by using the Rational Application Developer Migration wizard:

1. Right-click the EmailSenderEAR project, and select **Migrate** → **J2EE Migration wizard**.
2. Review the warning statements on the first page, shown in Figure 9-14 on page 206.

Note: 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.

Click **Next**.


J2EE Migration Wizard
J2EE Migration Wizard Welcome Page
Welcome to the J2EE Migration Wizard

Please read the following information, then click Next.

The Migration Wizard guides the process of optionally migrating your J2EE project structures and the J2EE version of the associated projects.

⚠ Prior to proceeding with the Migration Wizard it is strongly recommended you:
Read the online help on Migrating applications to J2EE 1.3 and J2EE 1.4 specification levels."

Backup your entire workspace.

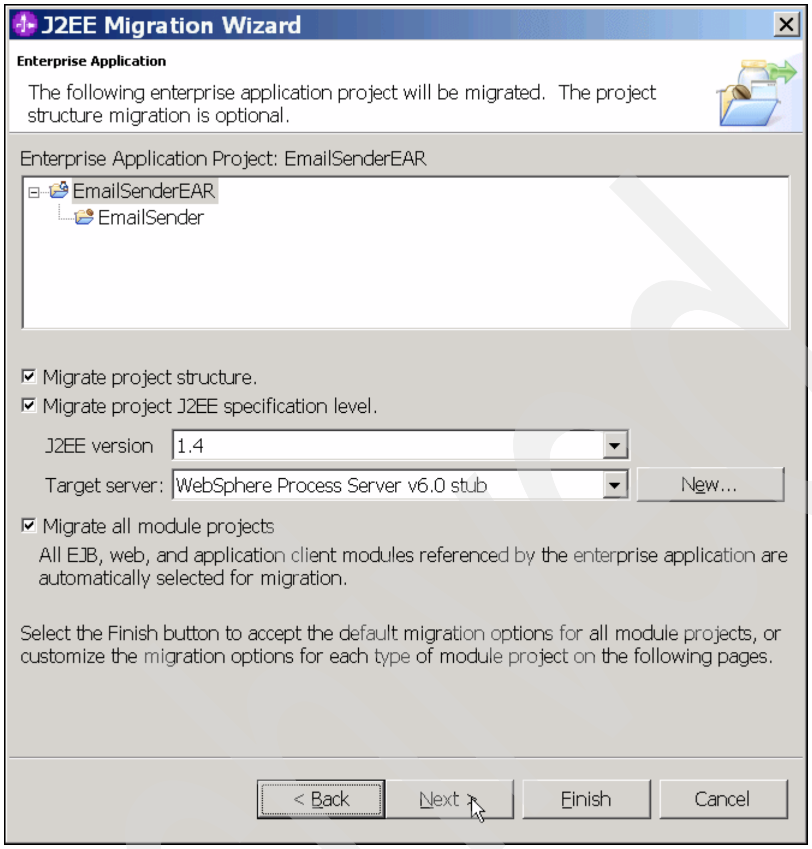
If working with a repository, "check out" or "lock" all corresponding projects.

Once the Migration Wizard is complete, the details of the steps performed during migration will be presented in a results window. This information is also written to the LoggingUtil.log file in the .metadata folder in your local workspace.

Based on the number of projects, migration may take a few minutes.

Figure 9-14 J2EE Migration wizard: Warning dialog

3. Ensure that the EmailSenderEAR project is highlighted as the project to be migrated, as shown in Figure 9-15 on page 207. Leave the **Migrate project structure** and **Migrate J2EE specification level** options selected. Choose **J2EE version 1.4** and **Target Server WebSphere Process Server v6.0 stub**. Click **Next**.



J2EE Migration Wizard

Enterprise Application

The following enterprise application project will be migrated. The project structure migration is optional.

Enterprise Application Project: EmailSenderEAR

EmailSenderEAR
EmailSender

☑ Migrate project structure.
☑ Migrate project J2EE specification level.

J2EE version: 1.4
Target server: WebSphere Process Server v6.0 stub

☑ Migrate all module projects
All EJB, web, and application client modules referenced by the enterprise application are automatically selected for migration.

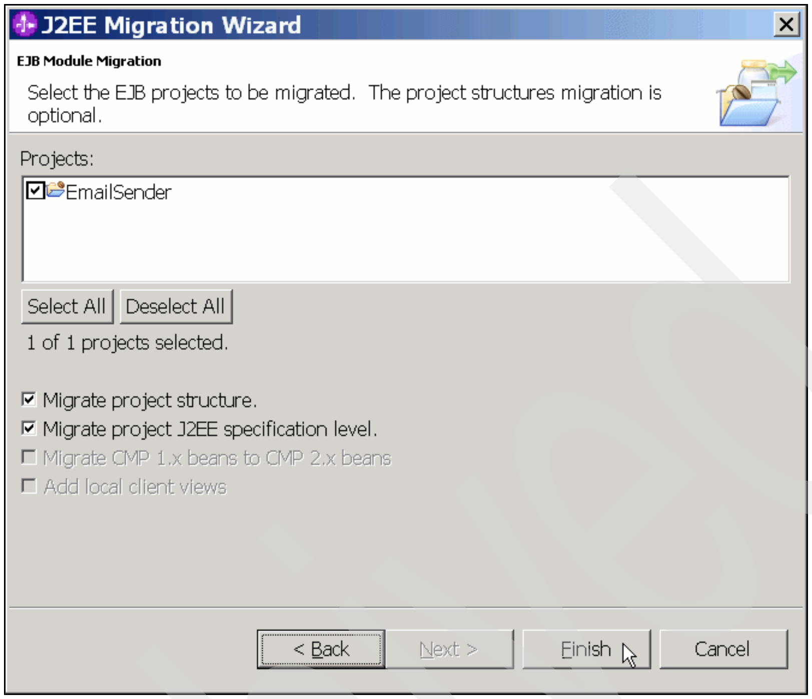
Select the Finish button to accept the default migration options for all module projects, or customize the migration options for each type of module project on the following pages.

Figure 9-15 J2EE Migration wizard: Migration options

4. In the Projects list, ensure that the EmailSender project is selected. Leave the **Migrate project structure** and **Migrate J2EE specification level** options selected, as shown in Figure 9-16 on page 208.

   Note: Depending on your EJB project, there might be other options that are appropriate.

   Click **Finish**.


J2EE Migration Wizard

EJB Module Migration

Select the EJB projects to be migrated. The project structures migration is optional.

Projects:

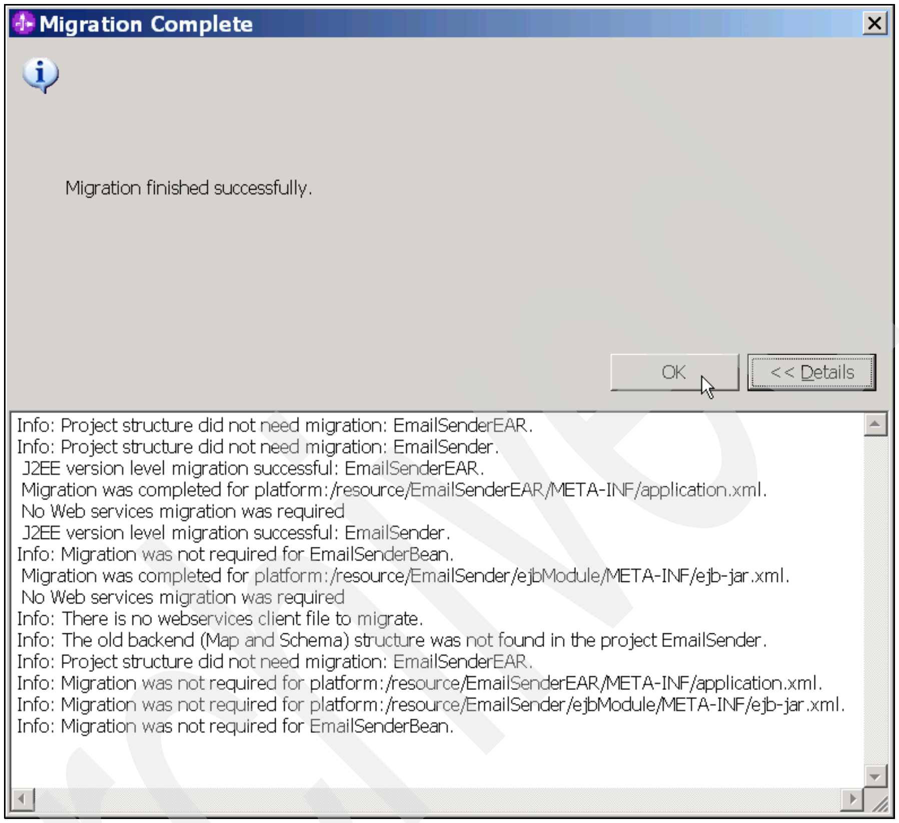
[✓] EmailSender

1 of 1 projects selected.

[✓] Migrate project structure.
[✓] Migrate project J2EE specification level.
[ ] Migrate CMP 1.x beans to CMP 2.x beans
[ ] Add local client views

Figure 9-16 J2EE Migration wizard: Migration options for EJB projects

5. If this step completes successfully, a successful migration message is displayed. Click **Details** to review what was migrated. Click **OK** to close the dialog, as shown in Figure 9-17.



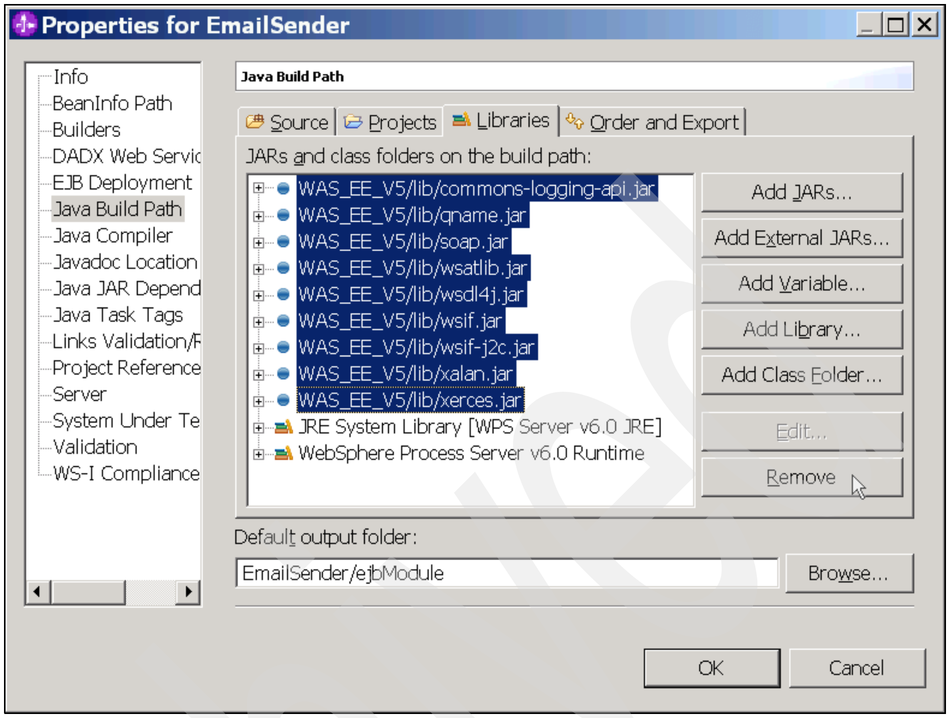
Migration finished successfully.

```
Info: Project structure did not need migration: EmailSenderEAR.
Info: Project structure did not need migration: EmailSender.
 J2EE version level migration successful: EmailSenderEAR.
 Migration was completed for platform:/resource/EmailSenderEAR/META-INF/application.xml.
 No Web services migration was required
 J2EE version level migration successful: EmailSender.
Info: Migration was not required for EmailSenderBean.
 Migration was completed for platform:/resource/EmailSender/ejbModule/META-INF/ejb-jar.xml.
 No Web services migration was required
Info: There is no webservices client file to migrate.
Info: The old backend (Map and Schema) structure was not found in the project EmailSender.
Info: Project structure did not need migration: EmailSenderEAR.
Info: Migration was not required for platform:/resource/EmailSenderEAR/META-INF/application.xml.
Info: Migration was not required for platform:/resource/EmailSender/ejbModule/META-INF/ejb-jar.xml.
Info: Migration was not required for EmailSenderBean.
```

Figure 9-17 J2EE Migration wizard: Migration completed successfully

To resolve the errors that are displayed in the Problems view, as shown in Figure 9-18 on page 210, fix the classpath for the EmailSender project. Remove all classpath entries that refer to V5 .jar files or libraries, and add the JRE™ System Library and WPS Server Target libraries, if applicable, to the classpath instead.

1. Right-click the EmailSender project, and select **Properties**. Go to the Java Build Path entry, and click the **Libraries** tab.
2. Select the jar files that are displayed in the Problems view (Figure 9-18 on page 210), and click **Remove**. Ensure that the JRE System Library and WPS Server Target libraries are present. Click **OK**.


Properties for EmailSender

Java Build Path

JARs and class folders on the build path:

```
WAS_EE_V5/lib/commons-logging-api.jar
WAS_EE_V5/lib/qname.jar
WAS_EE_V5/lib/soap.jar
WAS_EE_V5/lib/wsatlib.jar
WAS_EE_V5/lib/wsdl4j.jar
WAS_EE_V5/lib/wsif.jar
WAS_EE_V5/lib/wsif-j2c.jar
WAS_EE_V5/lib/xalan.jar
WAS_EE_V5/lib/xerces.jar
JRE System Library [WPS Server v6.0 JRE]
WebSphere Process Server v6.0 Runtime
```

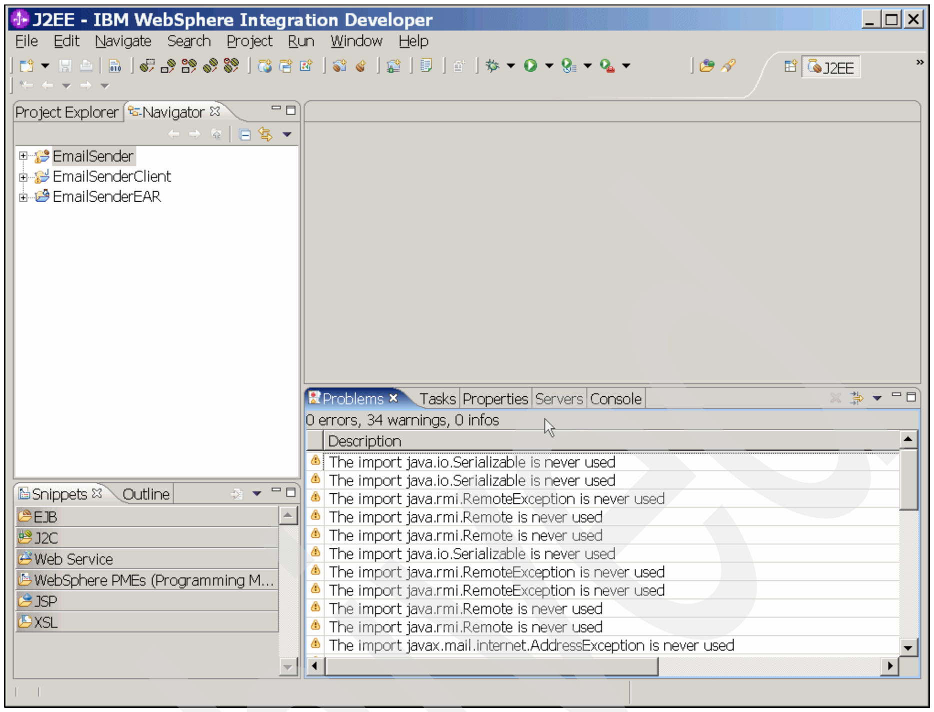
Default output folder: EmailSender/ejbModule

Figure 9-18 Java Build Path of EmailSender: clean up the classpath

3. 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 checked. Click **OK**.

   After the build completes, there are only warnings left in the Problems view, as seen in Figure 9-19 on page 211.



```
0 errors, 34 warnings, 0 infos
Description
The import java.io.Serializable is never used
The import java.io.Serializable is never used
The import java.rmi.RemoteException is never used
The import java.rmi.Remote is never used
The import java.rmi.Remote is never used
The import java.io.Serializable is never used
The import java.rmi.RemoteException is never used
The import java.rmi.RemoteException is never used
The import java.rmi.Remote is never used
The import java.rmi.Remote is never used
The import javax.mail.internet.AddressException is never used
```

Figure 9-19 The J2EE projects in Interface Editor after classpath clean up

This finishes the migration of the J2EE projects in this scenario.

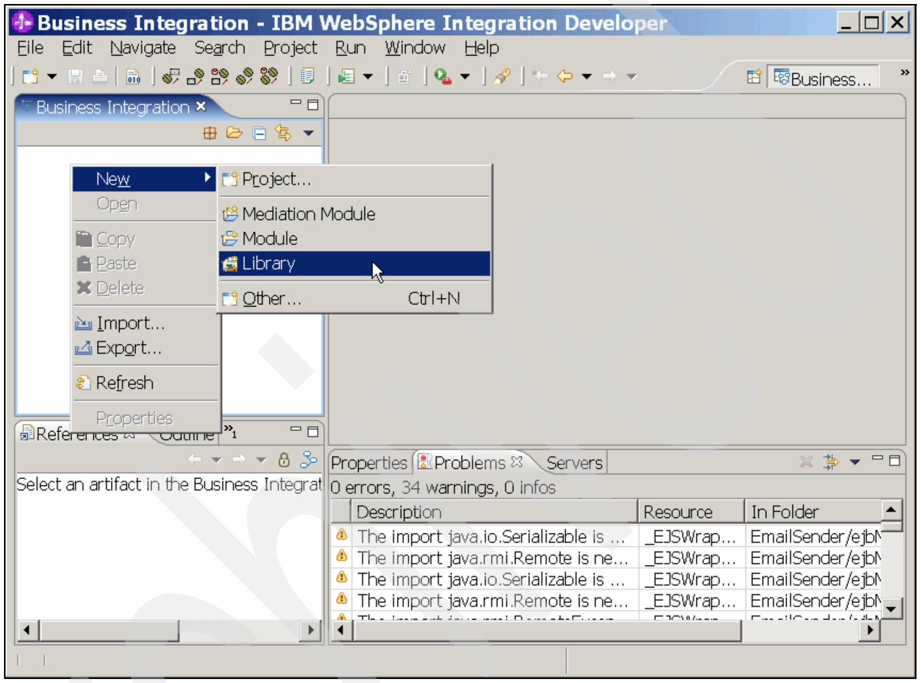
Migrating the CommonArtifacts Java project

The CommonArtifacts project is a Java project that contains .wsdl files, which are used by the TravelAgency and the TravelOperations service projects. You must manually move this project:

1. In WebSphere Integration Developer, create a new Business Integration Library project named CommonArtifacts.
2. Manually copy the old .wsdl and .xsd files from the 5.1 shared Java project to this new BI Library project folder.

> **Note:** Perform this before you migrate the service projects because the service projects have dependencies to this library. To resolve the dependencies during migration of the service projects, the library must already be present in the Integration Developer workspace.

3. In the WebSphere Integration Developer workspace open the Business Integration Perspective, and select **Window** → **Open Perspective** → **Business Integration**.
4. Right-click in the **Business Integration** view, and select **New** → **Library**, as shown in Figure 9-20.



Select an artifact in the Business Integrat

0 errors, 34 warnings, 0 infos

Description	Resource	In Folder
The import java.io.Serializable is ...	_EJSWrap...	EmailSender/ejbM
The import java.rmi.Remote is ne...	_EJSWrap...	EmailSender/ejbM
The import java.io.Serializable is ...	_EJSWrap...	EmailSender/ejbM
The import java.rmi.Remote is ne...	_EJSWrap...	EmailSender/ejbM

Figure 9-20 Creating a Business Library in Integration Developer

5. In the New Library dialog, enter `CommonArtifacts` as Library Name. Ensure that **Library Location Use default** is selected. Click **Finish**.
6. Using file system operations, go to the exported V5.1 CommonArtifacts project, and copy the folder `com`.
7. Got to the new V6 CommonArtifacts projects in the Integration Developer workspace directory, and paste the folder `com` there.
8. In the Business Integration view in WebSphere Integration Developer, right-click the CommonArtifacts Library project, and select **Refresh**. The moved .wsdl files are now displayed in the project structure, as shown in Figure 9-21 on page 213.

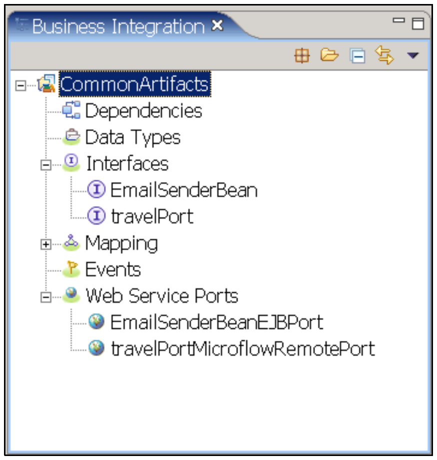


Figure 9-21 CommonArtifacts project structure after migration

9. 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 checked. Click **OK**.

This completes the migration of the CommonArtifacts project in this scenario.

Migrating the service projects

There are two service projects in this scenario:

- TravelAgency
- TravelOperation

These services projects are migrated using the WebSphere Integration Developer Migration wizard.

.

Note: You must migrate the service projects in their dependency order, for example, if a BPEL in service project A makes a process-to-process call to a BPEL in service project B, then service project B must be migrated before service project A. Otherwise, the process-to-process call cannot be configured correctly.

The TravelOperationProcess within TravelOperations project calls the TravelAgencyProcess, which means you must migrate the TravelAgency project first.

Note: There might get unresolved build errors in Integration Developer V6.0.2, which persists after you tried a clean and rebuild of all the projects in the workspace. This is a known issue in WebSphere Integration Developer V6.0.2.

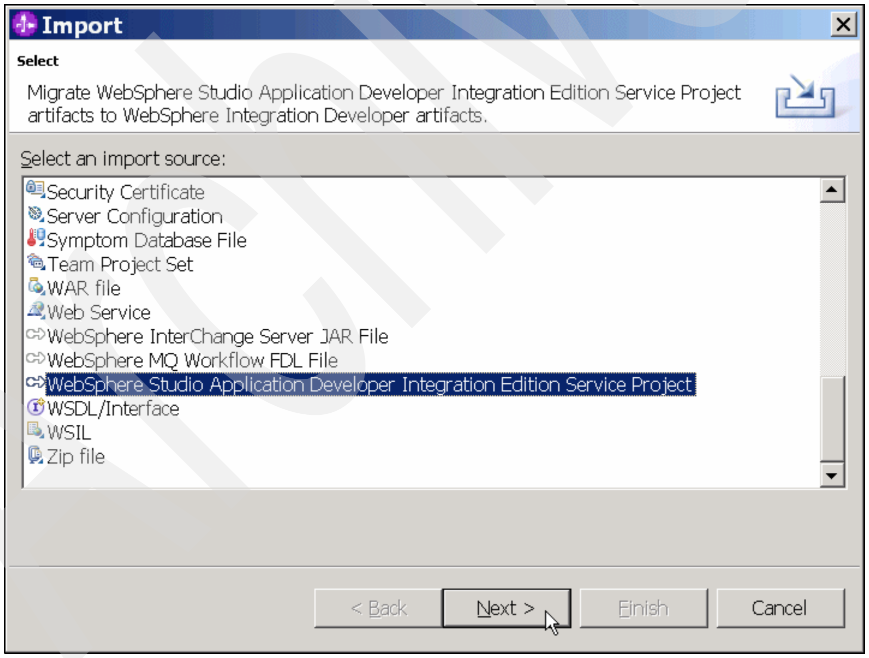
Install APAR JR25914 in order to solve this problem and rebuild the workspace.

For more information, refer to section 10.2, “Scenarios” on page 277, Build errors in Integration Developer.

Migrating the TravelAgency project

To migrate the TravelAgency project using the WebSphere Integration Developer Migration wizard:

1. Invoke the wizard by selecting **File** → **Import** → **WebSphere Studio Application Developer Integration Edition Service Project**. Click **Next**, as shown in Figure 9-22.



Migrate WebSphere Studio Application Developer Integration Edition Service Project artifacts to WebSphere Integration Developer artifacts.

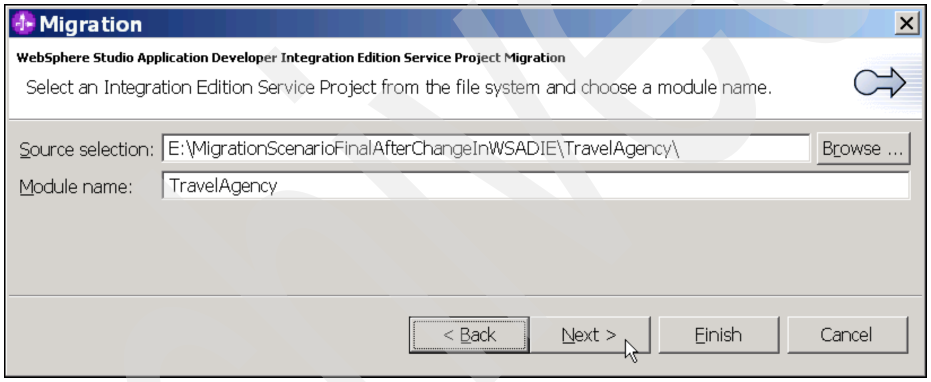
Select an import source:

- Security Certificate
- Server Configuration
- Symptom Database File
- Team Project Set
- WAR file
- Web Service
- WebSphere InterChange Server JAR File
- WebSphere MQ Workflow FDL File
- WebSphere Studio Application Developer Integration Edition Service Project
- WSDL/Interface
- WSIL
- Zip file

Figure 9-22 Import the WebSphere Studio Application Developer Integration Edition TravelAgency project.

2. The Migration wizard opens. For the **Source selection**, enter the path to the exported WebSphere Studio Application Developer Integration Edition project TravelAgency, as shown in Figure 9-23. Alternately, you can click **Browse**, and navigate to this folder. Choose a module name for the project in WebSphere Integration Developer. In this scenario, leave the TravelAgency as the Module name. Click **Next**.

Note: We recommend that you choose the name of the Service Project as the module name because if there are other projects in the WebSphere Studio Application Developer Integration Edition workspace that are dependent on this project, you do not have to update the dependent projects' classpaths after you import them into WebSphere Integration Developer.

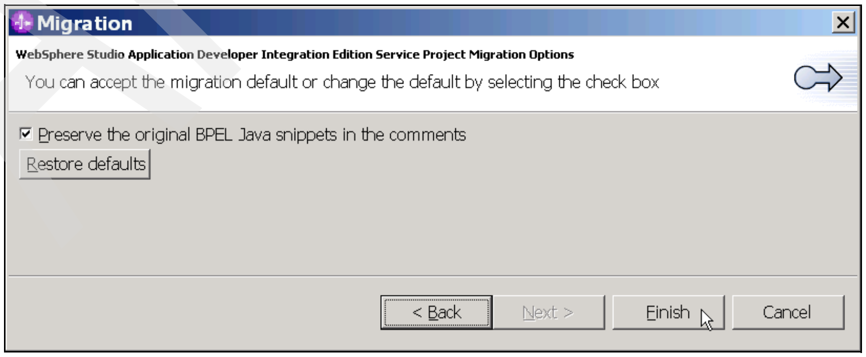

Migration
WebSphere Studio Application Developer Integration Edition Service Project Migration
< Back
Next >
Finish
Cancel
Browse ...

Select an Integration Edition Service Project from the file system and choose a module name.

Source selection: E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelAgency\
Module name: TravelAgency

Figure 9-23 Enter the location of the TravelAgency project to be migrated

3. Select **Preserve the original BPEL Java snippets in the comments**, as shown in Figure 9-24.


Migration
WebSphere Studio Application Developer Integration Edition Service Project Migration Options
Restore defaults
< Back
Next >
Finish
Cancel

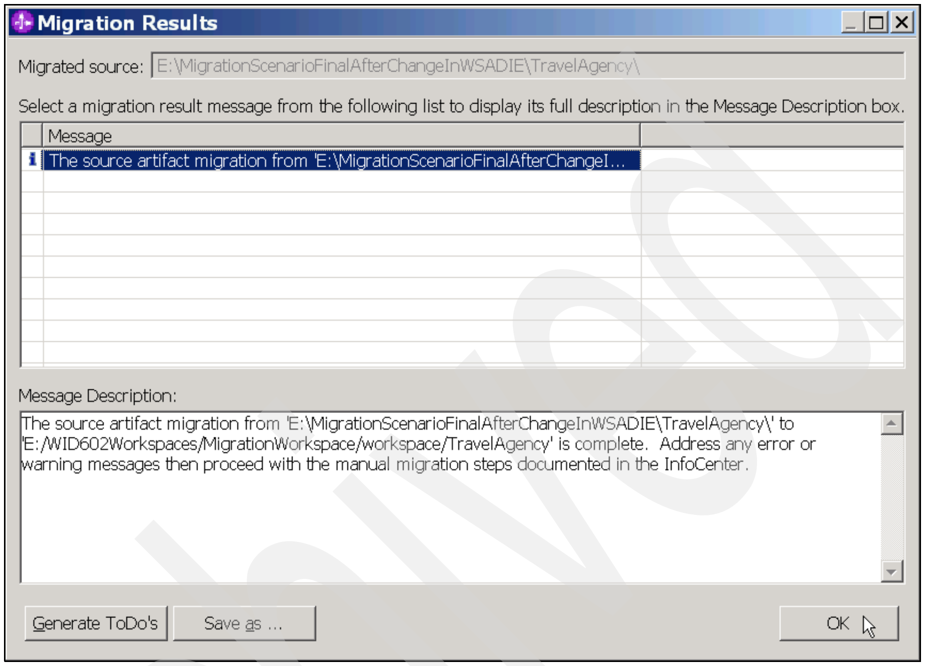
You can accept the migration default or change the default by selecting the check box

☑ Preserve the original BPEL Java snippets in the comments

Figure 9-24 Select Preserve the original BPEL Java snippets in the comments

4. Click **Finish**. This starts the migration of the TravelAgency project.

   After the wizard is finished, the results of the migration process are displayed as shown in Figure 9-25.



Migrated source: E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelAgency\

Select a migration result message from the following list to display its full description in the Message Description box.

Message
The source artifact migration from 'E:\MigrationScenarioFinalAfterChangeI...

Message Description:

The source artifact migration from 'E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelAgency\' to 'E:/WID602Workspaces/MigrationWorkspace/workspace/TravelAgency' is complete. Address any error or warning messages then proceed with the manual migration steps documented in the InfoCenter.

Figure 9-25 Migration Results for TravelAgency service project.

   Click **OK**.

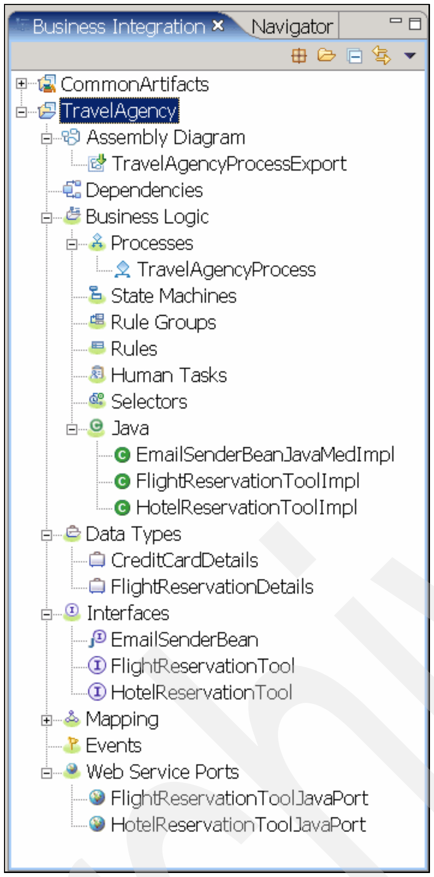
5. Explore the newly generated TravelAgency module in the Business Integration view, as shown in Figure 9-26 on page 217.

   – 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, you find the .xsd files CreditCardDetails and FlightReservationDetails that are used for the invocation of HotelReservationTool and FlightReservationTool.

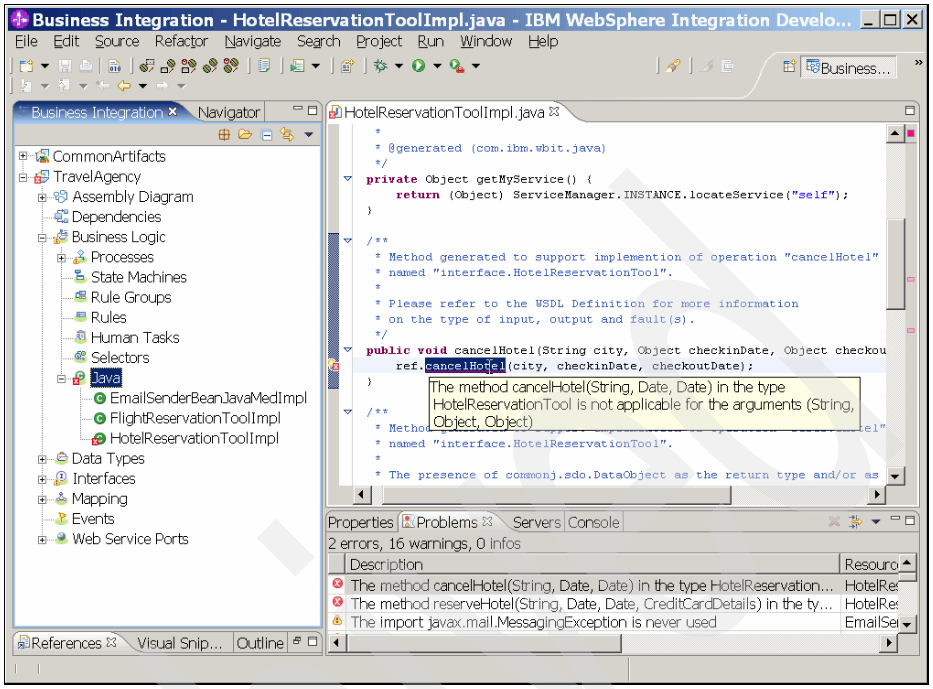
   – Under Interfaces, you find the .wsdl Interfaces for HotelReservationTool and FlightReservationTool. There is a new Java Interface that was created to invocate the EmailSender Session Bean from the TravelAgencyProcess.



```
CommonArtifacts
TravelAgency
    Assembly Diagram
        TravelAgencyProcessExport
    Dependencies
    Business Logic
        Processes
            TravelAgencyProcess
        State Machines
        Rule Groups
        Rules
        Human Tasks
        Selectors
        Java
            EmailSenderBeanJavaMedImpl
            FlightReservationToolImpl
            HotelReservationToolImpl
    Data Types
        CreditCardDetails
        FlightReservationDetails
    Interfaces
        EmailSenderBean
        FlightReservationTool
        HotelReservationTool
    Mapping
    Events
    Web Service Ports
        FlightReservationToolJavaPort
        HotelReservationToolJavaPort
```

Figure 9-26 TravelAgency module structure in Integration Developer

6. Clean and build all of the projects in the workspace. From the menu, select **Project** → **Clean**. In the Clean dialog, select **Clean all projects**, and select the **Start a build immediately** option. Click **OK**.
7. After the build is finished, observe the two errors that are displayed in the Problems view, as shown in Figure 9-27 on page 218. Double-click the first error to open the corresponding HotelReservationToolImpl.java file. This class was generated by the Migration wizard to route the HotelReservationTool invocations from the BPEL process to the HotelReservationTool.java class. This is necessary because the invocation layer changed to SCA and SDO.



```
 *
 * @generated (com.ibm.wbit.java)
 */
private Object getMyService() {
    return (Object) ServiceManager.INSTANCE.locateService("self");
}

/**
 * Method generated to support implemention of operation "cancelHotel"
 * named "interface.HotelReservationTool".
 *
 * Please refer to the WSDL Definition for more information
 * on the type of input, output and fault(s).
 */
public void cancelHotel(String city, Object checkinDate, Object checkou
    ref.cancelHotel(city, checkinDate, checkoutDate);
}

/**
 * Method
 * named "interface.HotelReservationTool".
 *
 * The presence of commonj.sdo.DataObject as the return type and/or as
```

The method cancelHotel(String, Date, Date) in the type HotelReservationTool is not applicable for the arguments (String, Object, Object)

2 errors, 16 warnings, 0 infos

	Description	Resourc
⊗	The method cancelHotel(String, Date, Date) in the type HotelReservation...	HotelRes
⊗	The method reserveHotel(String, Date, Date, CreditCardDetails) in the ty...	HotelRes
⚠	The import javax.mail.MessagingException is never used	EmailSer

Figure 9-27 Build errors in the HotelReservationToolImpl.java class.

The method in question requires input parameters of type java.util.Date instead of the java.lang.Object, which the Migration wizard generated. Cast the input parameters to java.util. Date, as shown in Example 9-3.

Save the file after you change it.

Example 9-3 Methods with input parameters cast to java.util.Date.

```
/**
* Method generated to support implementation of operation "cancelHotel"
* defined for WSDL port type named "interface.HotelReservationTool".
*
* Please refer to the WSDL Definition for more information
* on the type of input, output and fault(s).
*/
public void cancelHotel(String city, Object checkinDate, Object
checkoutDate) {

      ref.cancelHotel(city, (java.util.Date)checkinDate,
            (java.util.Date)checkoutDate);
}
```

```
/**
* Method generated to support implemention of operation "reserveHotel"
* defined for WSDL port type named "interface.HotelReservationTool".
*
* The presence of commonj.sdo.DataObject as the return type and/or as a
* parameter type conveys that its a complex type. Please refer to the
* WSDL Definition for more information on the type of input, output and
* fault(s).
*/
public boolean reserveHotel(String city, Object checkinDate,
   Object checkoutDate, DataObject creditCardDetails) {

      return ref.reserveHotel(city, (java.util.Date)checkinDate,
            (java.util.Date)checkoutDate,
            _convertTo_CreditCardDetails(creditCardDetails));
   }
```

8. Clean and build the TravelAgency project. From the menu, select **Project** → **Clean**. In the Clean dialog, select **Clean selected projects**, and choose TravelAgency. Select the **Start a build immediately** option. Click **OK**.

   After the build finishes, all errors in the Problems view are gone.

9. 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 9-4 displays. See also Figure 9-28 on page 221.

Table 9-4 Components, exports, and imports that the Migration wizard generated

Name	Type	Comment
TravelAgencyProcess	Process component	This component represents the migrated TravelAgencyProcess.bpel.

Name	Type	Comment
FlightReservationTool	Java component	This component represents the migrated FlightReservationTool Java service.
HotelReservationTool	Java component	This component represents the migrated HotelReservationTool Java service.
EmailSenderBeanJavaMed	Java component	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.
EmailSenderBean	Import with Stateless Session Bean Binding	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.
TravelAgencyProcessExport	Export with SCA Binding	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.

Observe that all these components are readily wired. Open their implementation by double-clicking the components.

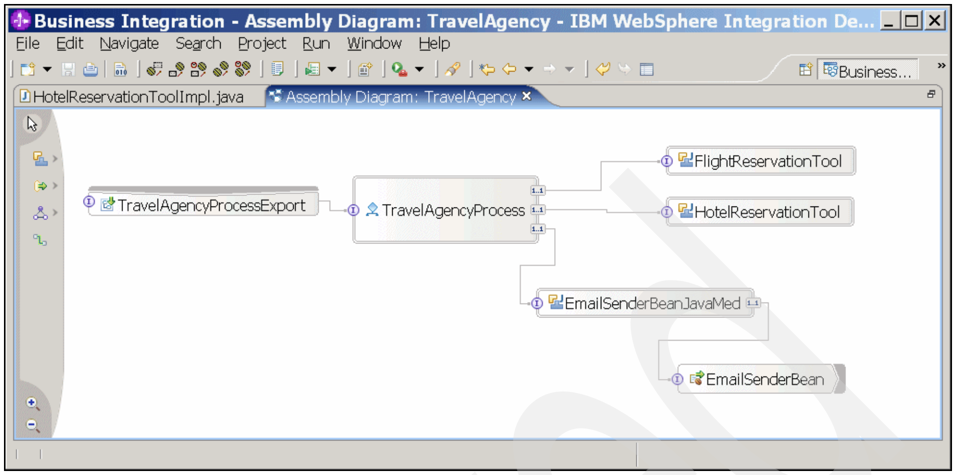

Business Integration - Assembly Diagram: TravelAgency - IBM WebSphere Integration De...
File Edit Navigate Search Project Run Window Help
Business...
HotelReservationToolImpl.java
Assembly Diagram: TravelAgency
TravelAgencyProcessExport
TravelAgencyProcess
FlightReservationTool
HotelReservationTool
EmailSenderBeanJavaMed
EmailSenderBean

Figure 9-28 Assembly Diagram of TravelAgency module

b. Double-click the **TravelAgencyProcess** component. When the Process Editor opens, inspect the migrated TravelAgencyProcess, as shown in Figure 9-28.

The Partner, Variable, and Correlation Set definitions are located at the right corner of the editor. Notice that Partners are divided in Interfaces Partners, which represent the interfaces of the process 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.

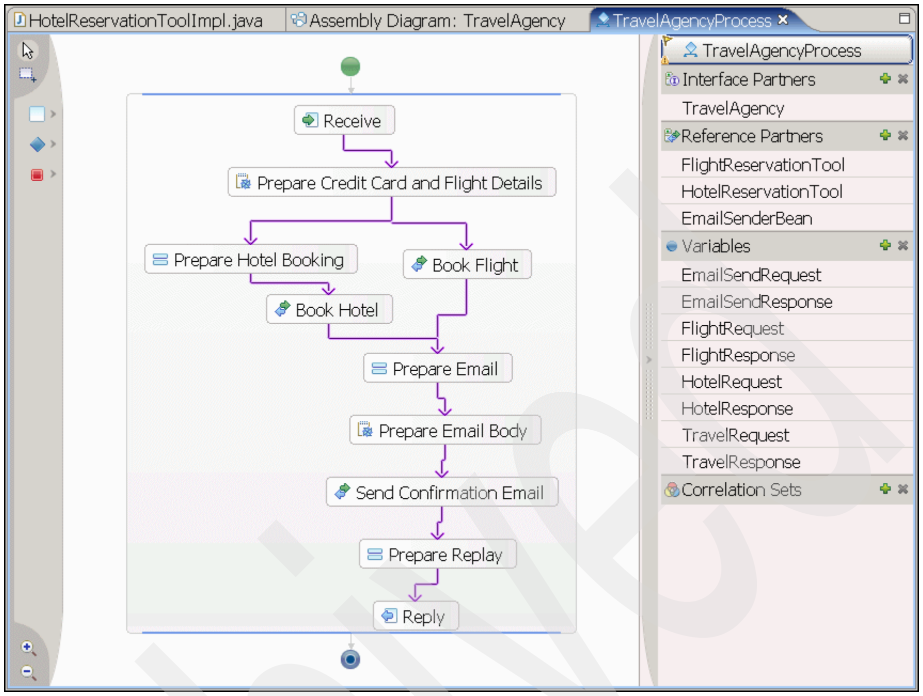


Figure 9-29 TravelAgencyProcess in Process Editor

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 9-4 shows the V5.1 code to set the CreditCardDetails in the FlightRequest variable. Compare Example 9-4 to Example 9-5 on page 223, which shows the Process Server Java programming model to fulfill this operation.

Example 9-4 Fill the CreditCardDetails in the FlightRequest variable in V5.1

```
// 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 9-5 shows the CreditCardDetails in the FlightRequest variable in V6.

Example 9-5 Fill in the CreditCardDetails in the FlightRequest variable in V6

```
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));
```

e. Close the TravelAgencyProcess by clicking the **X** on the top of the process editor.

f. 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. This Java class was generated by the Migration wizard to transform the data model used between calls from the TravelAgencyProcess to the HotelReservationTool Java implementation.

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 9-6 shows the converters for CreditCardDetails between DataObject and JavaBean.

Example 9-6 Converters for CreditCardDetails between DataObject and JavaBean, vice versa generated by the Migration wizard

```
/**
* 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;
}
```

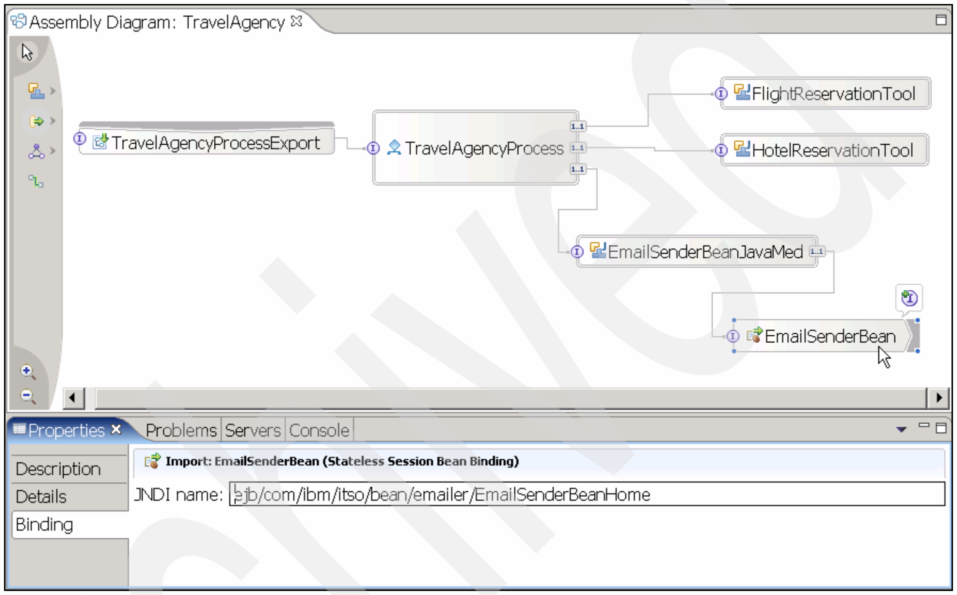
g. Close the HotelReservationToolImpl.java file by clicking the **X** on the top of the Java editor.

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 9-30. 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.



Import: EmailSenderBean (Stateless Session Bean Binding)

JNDI name: ejb/com/ibm/itso/bean/emailer/EmailSenderBeanHome

Figure 9-30 JNDI name in the EmailSenderBean import.

j. In the Assembly Diagram, double-click the **EmailSenderBeanJavaMed** component.

k. When the Java editor with the EmailSenderBeanJavaMedImpl.java opens, inspect the file. This class was generated by the Migration wizard 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 9-7.

Example 9-7 Locating the EmailSenderBean import

```
private com.ibm.itso.bean.emailer.EmailSenderBean ref =
   (com.ibm.itso.bean.emailer.EmailSenderBean) ServiceManager.INSTANCE
   .locateService("EmailSenderBean");
```

And calles the bean operation through the import when it is called:

Example 9-8 Calling the sendEmail method on the EmailSenderBean import

```
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.

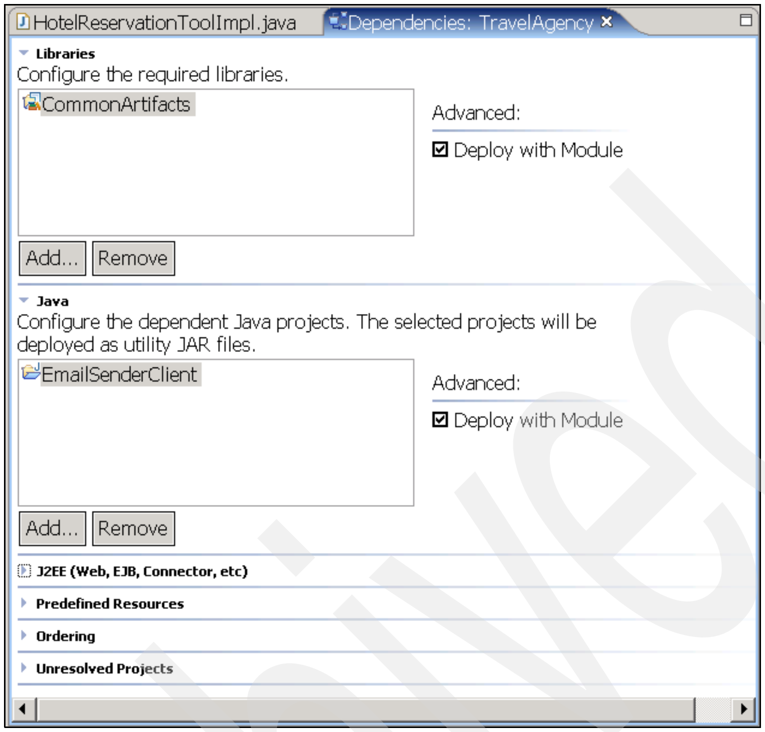
   This export is left in the Assembly Diagram because it can be used later by the TravelOperationsProcess to call the TravelAgencyProcess.

n. Close the Assembly Diagram by clicking **X** on the top of the Assembly Diagram editor.

o. Open the module dependency editor to ensure that the dependencies are set correctly. To do this, double-click **Dependencies** within the TravelAgency module in the **Business Integration** view.

   The Dependencies editor opens, as shown in Figure 9-31 on page 227. 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.

   Close the Dependencies editor by clicking the **X** on the top of the Dependency Editor.


HotelReservationToolImpl.java | Dependencies: TravelAgency

Libraries

Configure the required libraries.

CommonArtifacts

Advanced:

☑ Deploy with Module

Add... Remove

Java

Configure the dependent Java projects. The selected projects will be deployed as utility JAR files.

EmailSenderClient

Advanced:

☑ Deploy with Module

Add... Remove

J2EE (Web, EJB, Connector, etc)

Predefined Resources

Ordering

Unresolved Projects

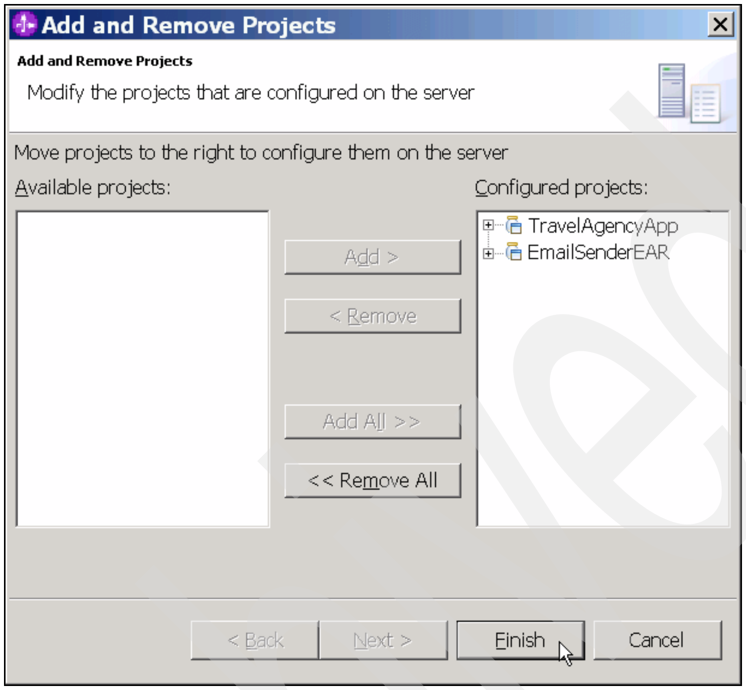
Figure 9-31 Dependencies of TravelAgency module.

This completes your inspection of the generated module.

10. Verify the migrated module operation, which is in the Integration Developer test environment.

   a. Go to the Servers view, and right-click WebSphere Process Server. Click **Start**.

   b. When the server is started, add the projects EmailSenderEAR and TravelAgencyApp to the server. In the **Servers** view, right-click the server, and select **Add and remove projects**. In the Add and Remove Projects dialog, click **Add All**. Click **Finish**, as shown in Figure 9-32.



Modify the projects that are configured on the server

Move projects to the right to configure them on the server

Available projects:

Configured projects:
- TravelAgencyApp
- EmailSenderEAR

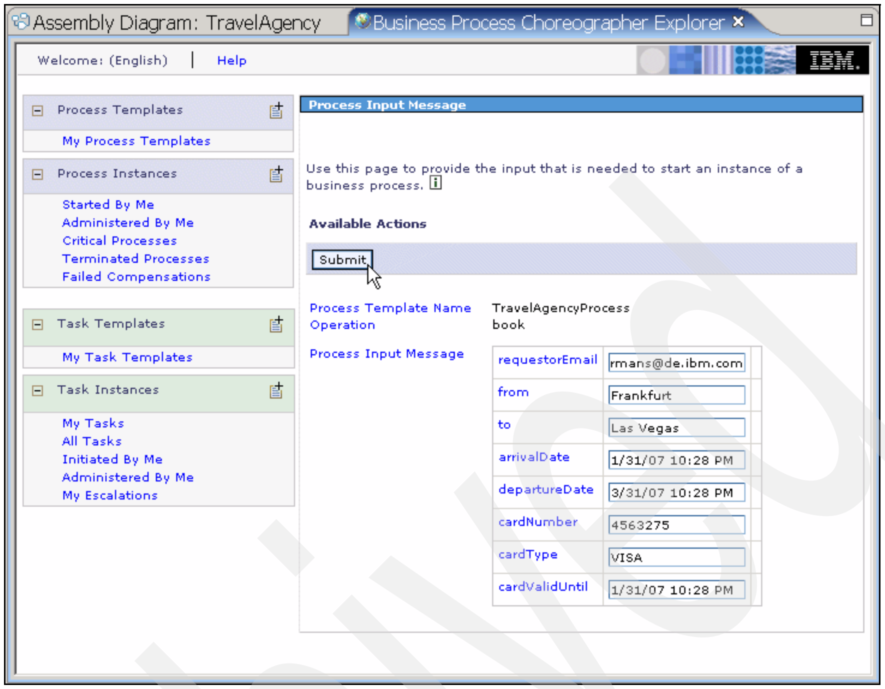
Figure 9-32 Add and Remove Projects dialog

This deploys and starts the TravelAgency module and the EmailSender enterprise application on the Process Server.

11.A short function test is performed using the Business Process Choreographer Explorer. Right-click the server, and select **Launch** → **Business Process Choreographer Explorer (BPC Explorer)**. The BPC Explorer is opened within WebSphere Integration Developer.

a. Start a TravelAgencyProcess by selecting **My Process Templates** from the navigation pane within the **BPC Explorer**. Check **TravelAgencyProcess**, and click **Start Instance.**

b. Complete the **Process Input Message** fields, and click **Submit**, as shown in Figure 9-33 on page 229.



Process Input Message

Use this page to provide the input that is needed to start an instance of a business process.

Process Template Name	TravelAgencyProcess
Operation	book

Process Input Message

requestorEmail	rmans@de.ibm.com
from	Frankfurt
to	Las Vegas
arrivalDate	1/31/07 10:28 PM
departureDate	3/31/07 10:28 PM
cardNumber	4563275
cardType	VISA
cardValidUntil	1/31/07 10:28 PM

Figure 9-33 BPC Explorer: Start a TravelAgencyProcess

c. Look at the SystemOut.log of the server, whether the processing of the TravelAgencyProcess is successful. Compare this output with the output of the V5.1 process that are listed in Example 9-1 on page 201.

   In this scenario, the TravelAgencyProcess is running through successfully.

d. Close the Business Process Explorer by clicking the **X** on top of the Explorer view.

e. Right-click the server in the **Servers** view, and click **Stop** to stop the server.

This finishes the initial function test of the migrated TravelAgency project in WebSphere Integration Developer.

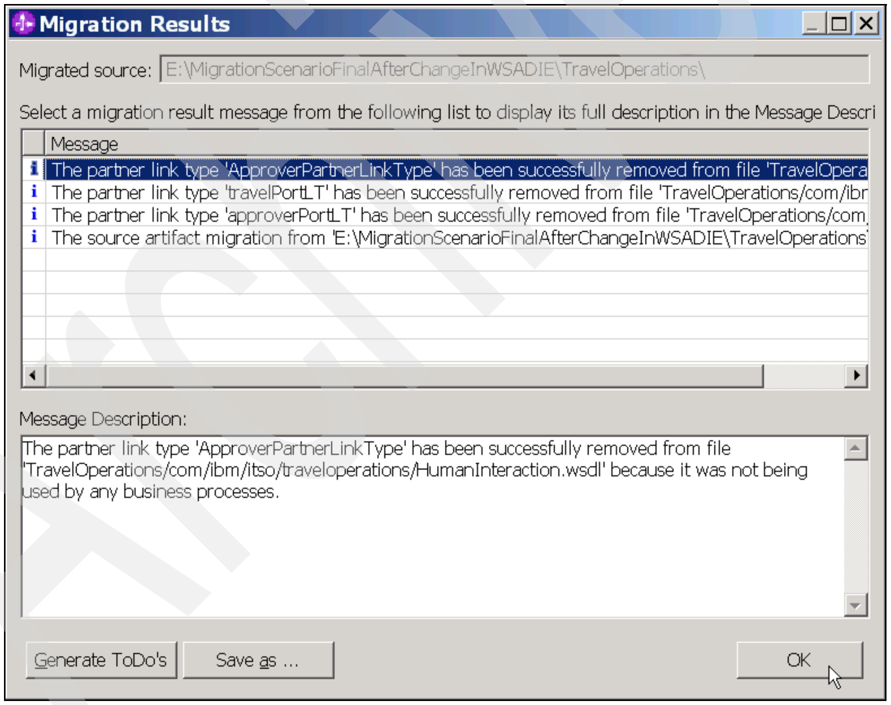
Migrating the TravelOperations project

Migrating the TravelOperations service project follows basically the same steps as migrating the TravelAgency project.

Use the WebSphere Integration Developer Migration wizard following these steps:

1. Invoke the wizard by selecting **File** → **Import** → **WebSphere Studio Application Developer Integration Edition Service Project**. Click **Next**.
2. The Migration wizard opens. For the Source selection, enter the path to the exported WebSphere Studio Application Developer Integration Edition project TravelOperations. Alternately, you can click **Browse**, and navigate to this folder. Choose a module name for the project in WebSphere Integration Developer. In this scenario, leave TravelOperations as the Module Name. Click **Next**.
3. Select the **Preserve the original BPEL Java snippets in the comments** option.
4. Click **Finish**. This starts the migration of the TravelOperations project.

   After the wizard is finished, the results of the migration process are displayed in a dialog box as shown in Figure 9-34. The Migration wizard indicates that it removed some partner link types. These were, for example in V5.1, needed for the interaction of the processes with the Staff activity.



Migrated source: E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelOperations\

Select a migration result message from the following list to display its full description in the Message Descri

	Message
i	The partner link type 'ApproverPartnerLinkType' has been successfully removed from file 'TravelOpera
i	The partner link type 'travelPortLT' has been successfully removed from file 'TravelOperations/com/ibr
i	The partner link type 'approverPortLT' has been successfully removed from file 'TravelOperations/com,
i	The source artifact migration from 'E:\MigrationScenarioFinalAfterChangeInWSADIE\TravelOperations'

Message Description:

The partner link type 'ApproverPartnerLinkType' has been successfully removed from file 'TravelOperations/com/ibm/itso/traveloperations/HumanInteraction.wsdl' because it was not being used by any business processes.

Figure 9-34 Migration Results for TravelOperations service project

Click **OK**.

5. Explore the newly generated TravelOperations module in the **Business Integration** view, as shown in Figure 9-35.

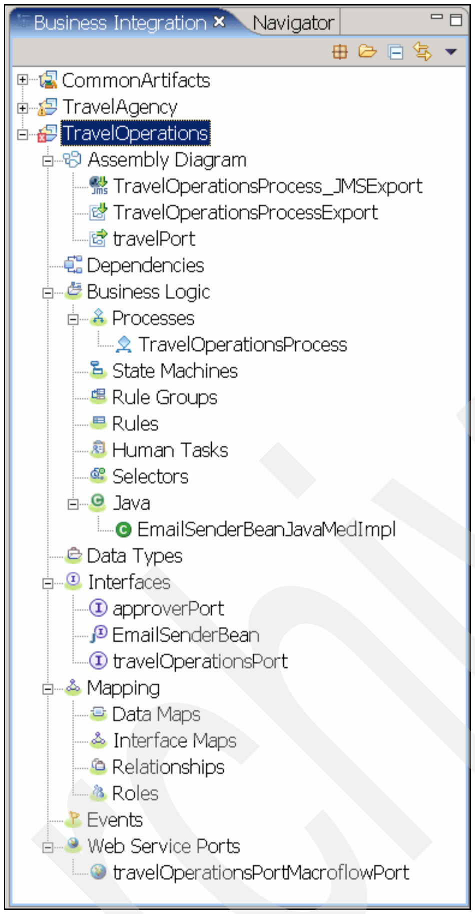

```
Business Integration   Navigator
CommonArtifacts
TravelAgency
TravelOperations
    Assembly Diagram
        TravelOperationsProcess_JMSExport
        TravelOperationsProcessExport
        travelPort
    Dependencies
    Business Logic
        Processes
            TravelOperationsProcess
        State Machines
        Rule Groups
        Rules
        Human Tasks
        Selectors
        Java
            EmailSenderBeanJavaMedImpl
    Data Types
    Interfaces
        approverPort
        EmailSenderBean
        travelOperationsPort
    Mapping
        Data Maps
        Interface Maps
        Relationships
        Roles
    Events
    Web Service Ports
        travelOperationsPortMacroflowPort
```

Figure 9-35   TravelOperations module structure in Integration Developer

There are errors in the Problems view. Resolve them by building all projects within the workspace.

6. Clean and build all projects in the workspace. From the menu, select **Project** → **Clean**. In the Clean dialog, select **Clean all projects** and select the **Start a build immediately** option. Click **OK**.

Troubleshooting: Repeat this step in case you encounter errors during the build.

After the build completes, all errors in the Problems view are gone.

7. 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 **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 9-5. See also Figure 9-36 on page 233.

Table 9-5 Components, Exports and Imports generated by the Migration wizard

Name	Type	Comment
TravelOperationsProcess	Process component	This component represents the migrated TravelOperationsProcess.bpel.
travelPort	Import with SCA binding	This component is the replacement for the EJB binding that was used to call the TravelAgencyProcess in V5.1. It is calling the TravelAgencyProcessExoprt export in the TravelAgency module.
EmailSenderBeanJavaMed	Java component	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.
EmailSenderBean	Import with Stateless Session Bean Binding	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.
TravelOperationsProcessExport	Export with SCA binding	This export enables the TravelOperations module to be called from another SCA module.

Name	Type	Comment
TraveOperationsProcess_JMSExport	Export with JMS binding	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.

Observe, that all this components are readily wired. Their implementation opens by double-clicking the components.

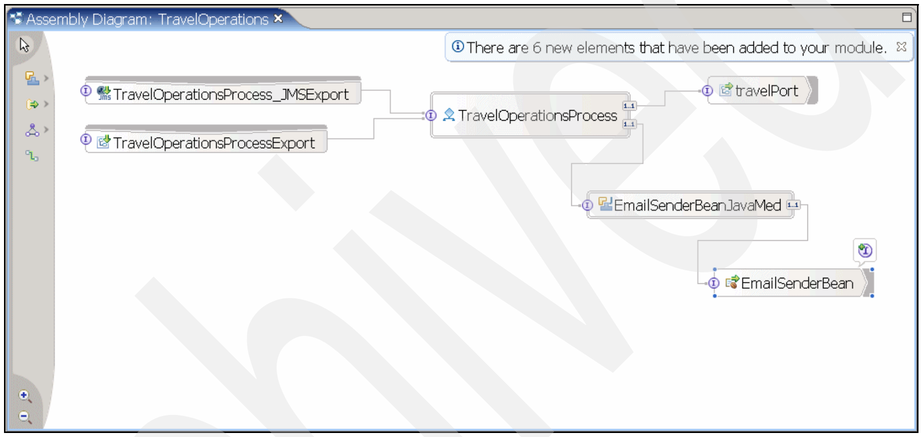

Assembly Diagram: TravelOperations
There are 6 new elements that have been added to your module.
TravelOperationsProcess_JMSExport
TravelOperationsProcessExport
TravelOperationsProcess
travelPort
EmailSenderBeanJavaMed
EmailSenderBean

Figure 9-36 Assembly Diagram of TravelOperations module.

b. Double-click the **TravelOperationsProcess** component. When the Process Editor opens, inspect the migrated TravelOperationsProcess, as shown in Figure 9-37.

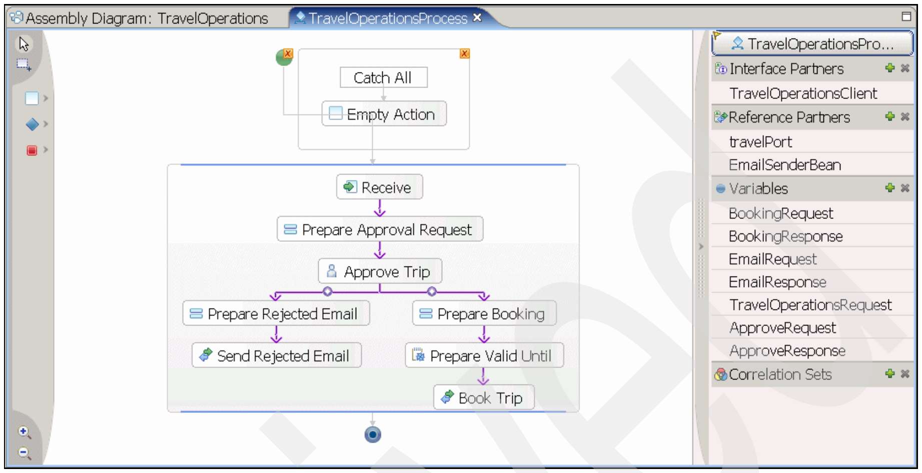
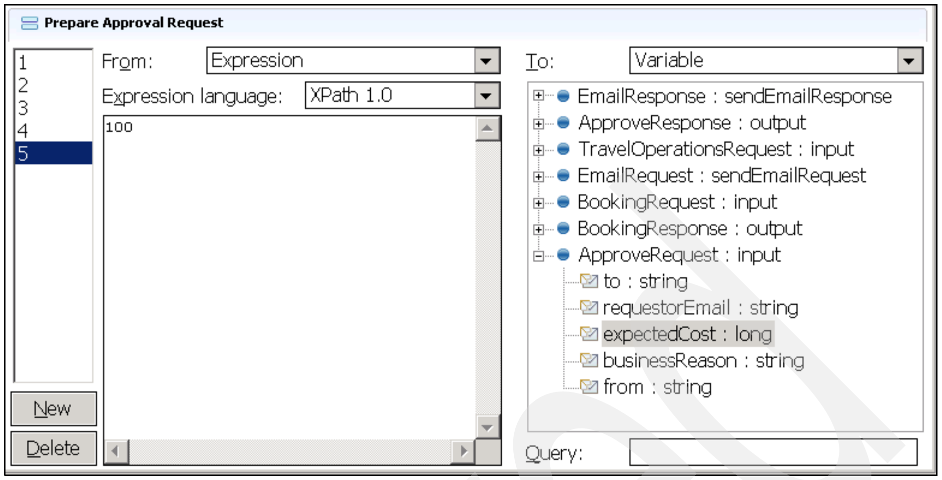


Figure 9-37 TravelOperationsProcess in Process Editor

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 selection 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 9-38 on page 235.



From: Expression
Expression language: XPath 1.0

```
100
```

To: Variable

- EmailResponse : sendEmailResponse
- ApproveResponse : output
- TravelOperationsRequest : input
- EmailRequest : sendEmailRequest
- BookingRequest : input
- BookingResponse : output
- ApproveRequest : input
  - to : string
  - requestorEmail : string
  - expectedCost : long
  - businessReason : string
  - from : string

Figure 9-38 Migrated Prepare Approval Request Assign activity

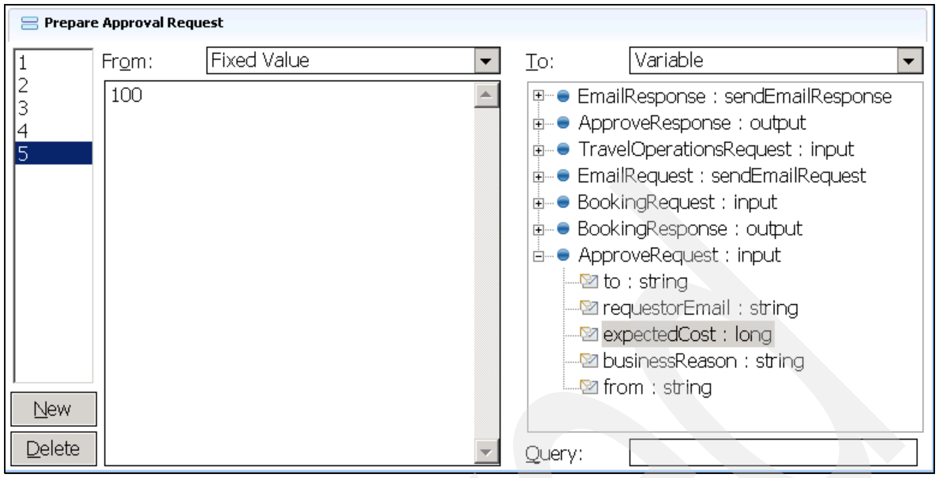
Assigns of fixed values are migrated to fixed values within XPath expressions.

Note: While running the migrated process in Process Server, an error was encountered, which relates to fixed values in XPath expressions. This turned out to be a known issue in WebSphere Process Server V6.0.2, which will be fixed in a future release.

To workaround this issue, the fixed value within the XPath expression is changed to fixed value.

For more information, refer to 10.2, “Scenarios” on page 277, Exception CWWBE0085E observed.

d. Clean up the Assign activities in question by changing the style to Fixed Value, as shown in Figure 9-39 on page 236 for assign item 5 in the Prepare Approval Request activity.


Prepare Approval Request

1
2
3
4
5

New
Delete

From: Fixed Value

100

To: Variable

- EmailResponse : sendEmailResponse
- ApproveResponse : output
- TravelOperationsRequest : input
- EmailRequest : sendEmailRequest
- BookingRequest : input
- BookingResponse : output
- ApproveRequest : input
  - to : string
  - requestorEmail : string
  - expectedCost : long
  - businessReason : string
  - from : string

Query:

Figure 9-39 Migrated Prepare Approval Request Assign activity with From Value

e. Repeat the same step for assign item 5 in the Prepare Booking activity.

f. Look at the Approve Trip activity. This 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.

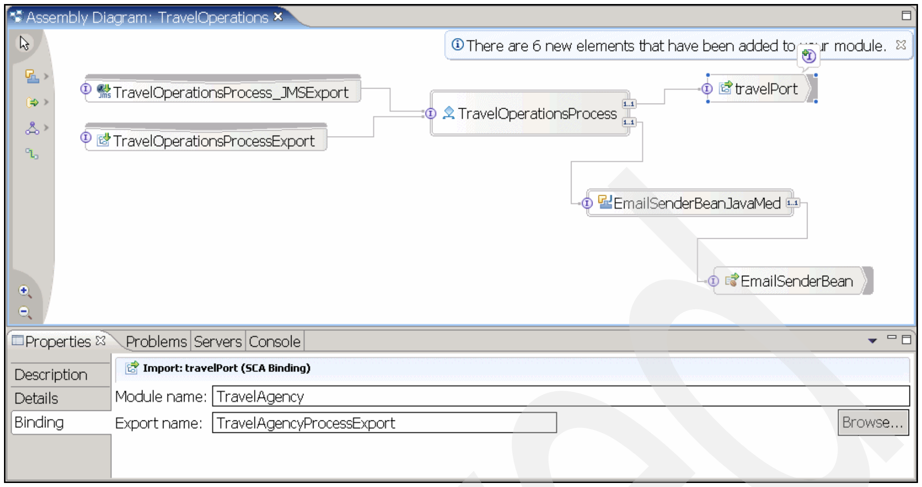
g. Inspect the task, and ensure that the staff settings, the staff provider settings, and the interfaces of the task are properly migrated, as shown in Figure 9-40 on page 237.

h. Close the ApproveTripTask by clicking **X** on the top of the task editor pane.

i. When you finish inspecting the TravelOperationsProcess, close it by clicking the **X** on the top of the process editor.

j. Click **Yes**, when asked for saving the file.

k. In the Assembly Diagram, select the **travelPort** import to display its properties in the Properties pane. Inspect its properties.


Assembly Diagram: TravelOperations
There are 6 new elements that have been added to your module.
TravelOperationsProcess_JMSExport
TravelOperationsProcessExport
TravelOperationsProcess
travelPort
EmailSenderBeanJavaMed
EmailSenderBean
Properties
Problems
Servers
Console
Description
Details
Binding
Browse...

Import: travelPort (SCA Binding)

Module name: TravelAgency

Export name: TravelAgencyProcessExport

Figure 9-40 TravelPort import directing to the TravelAgency module

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 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. When you want to take advantage of versioning, you need to remodel that part in the Assembly Diagram for late binding. More information on this is in "Migrating EJB process bindings" on page 111.

l. In the Assembly Diagram, select the **EmailSenderBean** import to display its properties in the **Properties** pane. Inspect its properties.

m. 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.

n. 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.

o. The **TravelOperationsProcessExport** export is not needed in our scenario; therefore, delete it. Right-click the

TravelOperationsProcessExport in the Assembly Diagram, and click **Delete**.

p. 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. Right-click the TravelOperationsProcess_JMSExport in the Assembly Diagram, and click **Delete**.

q. Save the Assembly Diagram. It now looks similar to Figure 9-41.

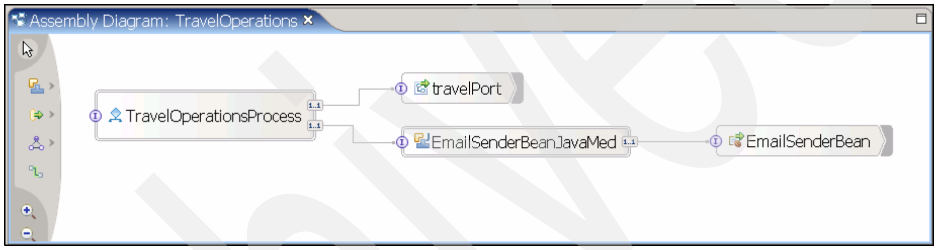
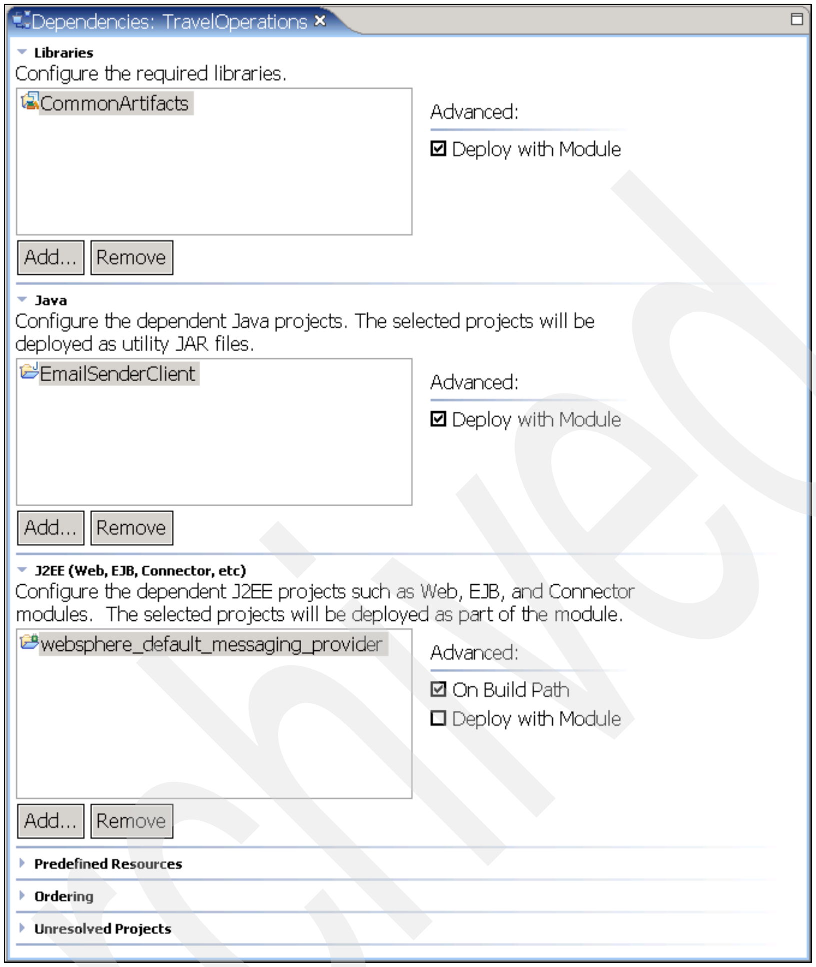


Figure 9-41 Assembly Diagram of TravelOperations module after deleting the exports

r. Close the Assembly Diagram by clicking the **X** on the top of the Assembly Diagram editor.

s. 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 9-42 on page 239). Both of these dependencies also existed in the former WebSphere Studio Application Developer Integration Edition project.


Dependencies: TravelOperations

Libraries

Configure the required libraries.

CommonArtifacts

Advanced:

☑ Deploy with Module

Java

Configure the dependent Java projects. The selected projects will be deployed as utility JAR files.

EmailSenderClient

Advanced:

☑ Deploy with Module

J2EE (Web, EJB, Connector, etc)

Configure the dependent J2EE projects such as Web, EJB, and Connector modules. The selected projects will be deployed as part of the module.

websphere_default_messaging_provider

Advanced:

☑ On Build Path

☐ Deploy with Module

Figure 9-42 Dependencies of TravelOperations module

t. 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 click **Remove**.

u. Close the Dependencies editor by clicking the **X** on the top of the Dependency Editor. Click **Yes**, when asked for saving the changes.

8. Changes were made to the TravelOperations module. Clean and build all projects in the workspace. From the menu, select **Project** → **Clean**. In the

Clean dialog, select **Clean all projects** and select the **Start a build immediately** option. Click **OK**.

9. Verify the migrated module operation in the Integration Developer test environment:

   a. Go to the **Servers** view, and right-click the WebSphere Process Server. Click **Start**.

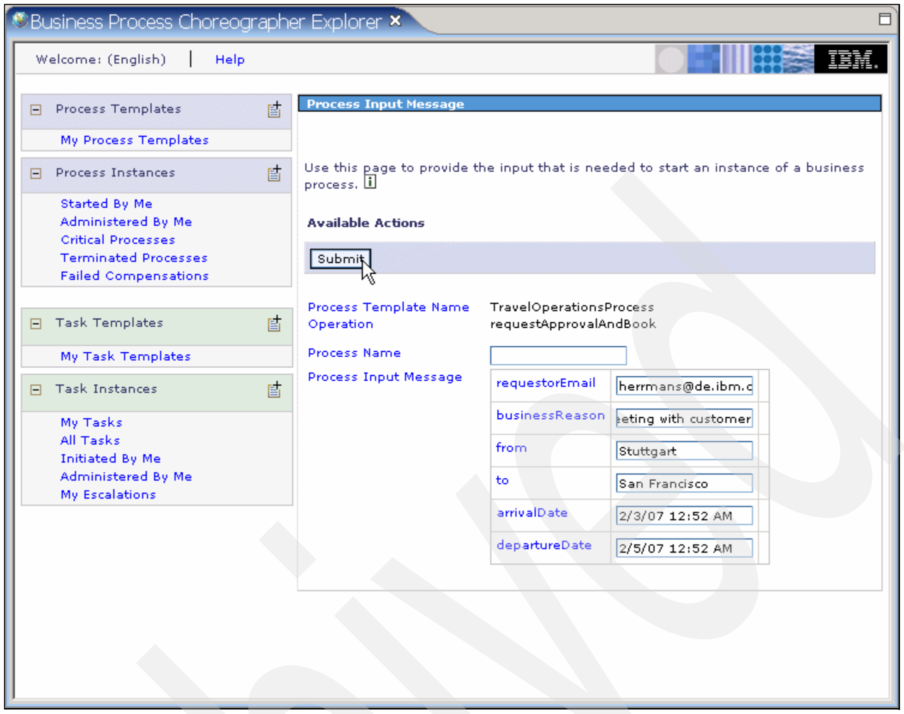
   b. When the server is started, add the project TravelOperationsApp to the server. In the Servers view, right-click the server, and select **Add and remove projects**. In the Add and Remove Projects dialog, click **Add All**. Click **Finish**.

   This deploys and starts the TravelOperations module on the Process Server. The TravelAgency module and the EmailSender bean are already on the running on the server.

10. A short function test is performed using the Business Process Choreographer Explorer. Right-click the server, and select **Launch** → **Business Process Choreographer Explorer (BPC Explorer)**. The BPC Explorer opens within WebSphere Integration Developer, as shown in Figure 9-43 on page 241:

    a. Start a TravelOperationsProcess by selecting **My Process Templates** from the navigation pane within the **BPC Explorer**. Select **TravelOperationsProcess**, and click **Start Instance.**

    b. Complete the **Process Input Message** fields, and click **Submit.**



Use this page to provide the input that is needed to start an instance of a business process.

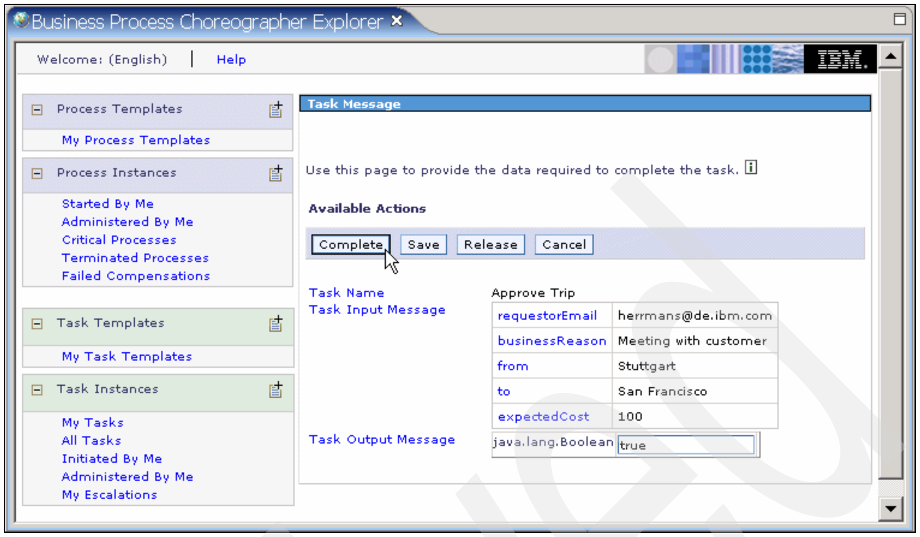
Process Template Name	TravelOperationsProcess
Operation	requestApprovalAndBook
Process Name	
Process Input Message	

requestorEmail	herrmans@de.ibm.c
businessReason	eting with customer
from	Stuttgart
to	San Francisco
arrivalDate	2/3/07 12:52 AM
departureDate	2/5/07 12:52 AM

Figure 9-43 BPC Explorer: Start a TravelOperationsProcess

c. A staff activity is created. Click **My Tasks** in the BPC Explorer navigation pane. There is one Approve Trip task listed. Select it, and click **Work on**.

d. Enter `true` in the Task Output Message field, and click **Complete**, as shown in Figure 9-44 on page 242.



Use this page to provide the data required to complete the task.

Task Name	Approve Trip
Task Input Message	

requestorEmail	herrmans@de.ibm.com
businessReason	Meeting with customer
from	Stuttgart
to	San Francisco
expectedCost	100

Task Output Message	java.lang.Boolean: true

Figure 9-44 BPC Explorer: Complete the task

e. Because you entered `true` as the result of the **ApproveTrip** activity, you must invoke the TravelAgencyProcess. Look at the SystemOut.log of the server, and see whether the processing of the TravelAgencyProcess is successful. Compare this output with the output of the V5.1 process shown in Example 9-1 on page 201.

f. By repeating steps a-e, but entering `false` as a result of the ApproveTrip activity in step d, the TravelAgencyProcess is not invoked. Instead, the TravelOperationsProcess calls the EmailSender bean to send a rejection email. 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 9-2 on page 201.

g. Close the Business Process Explorer by clicking the **X** on top of the Explorer view.

h. Right-click the server in the **Servers** view, and click **Stop** to stop the server.

This finishes the initial function test of the migrated TravelOperations project in WebSphere Integration Developer.

Deploying the applications

The final step involved in the migration process is to export the application EAR files and deploy them to WebSphere Process Server.

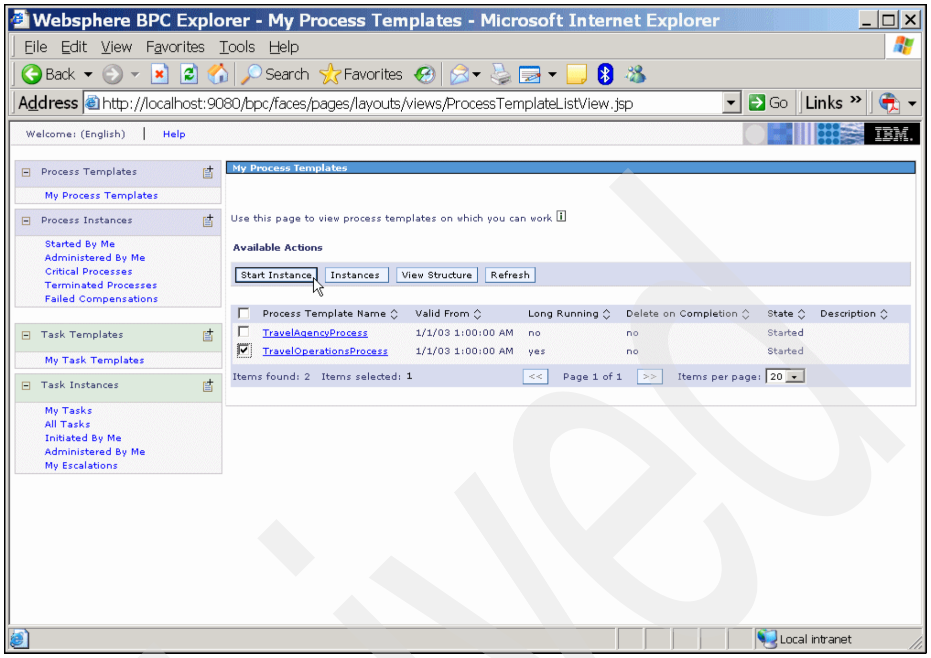
1. Click **File**, and select **Export**.
2. Select EAR file, and click **Next**.
3. Specify the EAR Project as *EmailSenderEAR,* and specify a destination.
4. Click **Finish**.
5. Repeat the steps for the other two projects: TravelAgencyApp and TravelOperationsApp.
6. Install all of the three applications using the admin console.
7. Start the applications on the server.

9.3.4 Verification

In this section, we explain how to test the migrated scenario in WebSphere Process Server and verify the results. We use the BPC Explorer to test the scenario.

To test the migrated scenario in WebSphere Process Server:

1. Open the BPC Explorer in Internet Explorer® or another supported Web Browser. In this scenario, the Process Server uses the default port 9080. Enter *http://localhost:9080/bpc/* as the URL.
2. Click **My Process Templates**. Select TravelOperationsProcess, and click **Start Instance**, as shown in Figure 9-45 on page 244.


Websphere BPC Explorer - My Process Templates - Microsoft Internet Explorer

Address http://localhost:9080/bpc/faces/pages/layouts/views/ProcessTemplateListView.jsp

Welcome: (English) | Help

Process Templates
- My Process Templates

Process Instances
- Started By Me
- Administered By Me
- Critical Processes
- Terminated Processes
- Failed Compensations

Task Templates
- My Task Templates

Task Instances
- My Tasks
- All Tasks
- Initiated By Me
- Administered By Me
- My Escalations

My Process Templates

Use this page to view process templates on which you can work

Available Actions

Start Instance | Instances | View Structure | Refresh

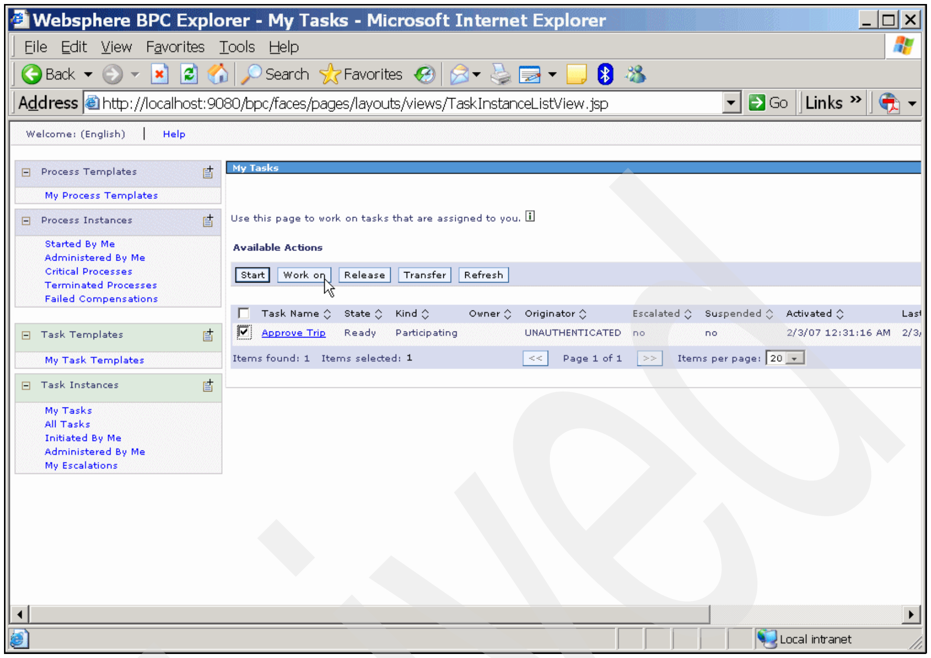
	Process Template Name	Valid From	Long Running	Delete on Completion	State	Description
☐	TravelAgencyProcess	1/1/03 1:00:00 AM	no	no	Started	
☑	TravelOperationsProcess	1/1/03 1:00:00 AM	yes	no	Started	

Items found: 2 Items selected: 1 Page 1 of 1 Items per page: 20

Local intranet

Figure 9-45 Starting instance of TravelOperationsProcess in BPC Explorer

3. Complete the **Process Input Message** fields, and click **Submit**. A staff activity is created.
4. In the BPC Explorer navigation pane, click **My Tasks**. Select the only **Approve Trip** task that is listed, and click **Work on**, as shown in Figure 9-46 on page 245.



Use this page to work on tasks that are assigned to you.

	Task Name	State	Kind	Owner	Originator	Escalated	Suspended	Activated	Last
☑	Approve Trip	Ready	Participating		UNAUTHENTICATED	no	no	2/3/07 12:31:16 AM	2/3/

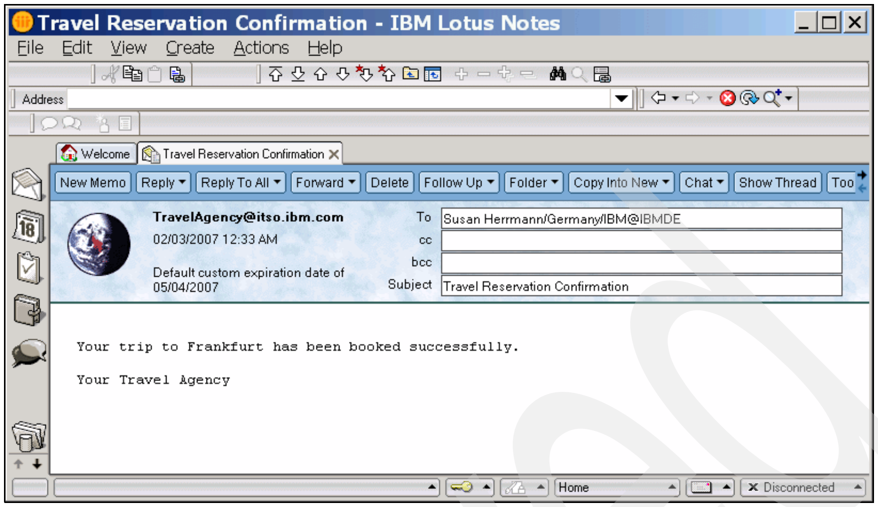
Items found: 1 Items selected: 1 << Page 1 of 1 >> Items per page: 20

Figure 9-46 Work on Approve Trip task in BPC Explorer

5. Enter `true` in the **Task Output Message** field, and click **Complete**.
6. Because you entered `true` as the result of the **ApproveTrip** activity, you must invoke the TravelAgencyProcess. Look at the SystemOut.log of the server, and see whether the processing of the TravelAgencyProcess is successful.

   The SystemOut.log can be found in <*WPS profile directory*>/logs/<*servername*>.
7. Compare this output with the output of the V5.1 process that is shown in Example 9-1 on page 201.

   If you entered a valid SMTP host in the TravelAgencyProcess, you can also verify whether the email was sent by the Process Server, as shown in Figure 9-47 on page 246.


Travel Reservation Confirmation - IBM Lotus Notes
File Edit View Create Actions Help
Address
Welcome
Travel Reservation Confirmation
New Memo
Reply
Reply To All
Forward
Delete
Follow Up
Folder
Copy Into New
Chat
Show Thread
Too
Home
Disconnected

TravelAgency@itso.ibm.com
02/03/2007 12:33 AM
Default custom expiration date of 05/04/2007

To: Susan Herrmann/Germany/IBM@IBMDE
cc:
bcc:
Subject: Travel Reservation Confirmation

```
Your trip to Frankfurt has been booked successfully.

Your Travel Agency
```

Figure 9-47 Email received from the TravelAgencyProcess running in Process Server

8. By repeating steps 1-5, but entering `false` as the result of the ApproveTrip activity in step 5, the TravelAgencyProcess is not invoked. Instead, the TravelOperationsProcess calls the EmailSender bean to send a rejection email. 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 that is shown in Example 9-2 on page 201.

This completes the verification in WebSphere Process Server of the migrated TravelOperations project in WebSphere Integration Developer.

Note: In real migration projects, you must perform extensive testing to verify the outcome of the migration.

9.4 Travel Booking scenario

The model to migrate from WebSphere Business Integration Modeler V5 to WebSphere Business Modeler V6 shows a simple booking process. The focus of this scenario is not to have a complete process defined in Modeler but to use the migration tool from Version 5 to Version 6 and to show the changes you need to perform in the artifacts to follow the new programming model in Process Server.

9.4.1 Overview

The Travel Booking scenario has some human task activities and a service invocation. Because business modeler is not the tool to define service invocations, we concentrate on the process definition, the business items management, and in the human task activities definition.

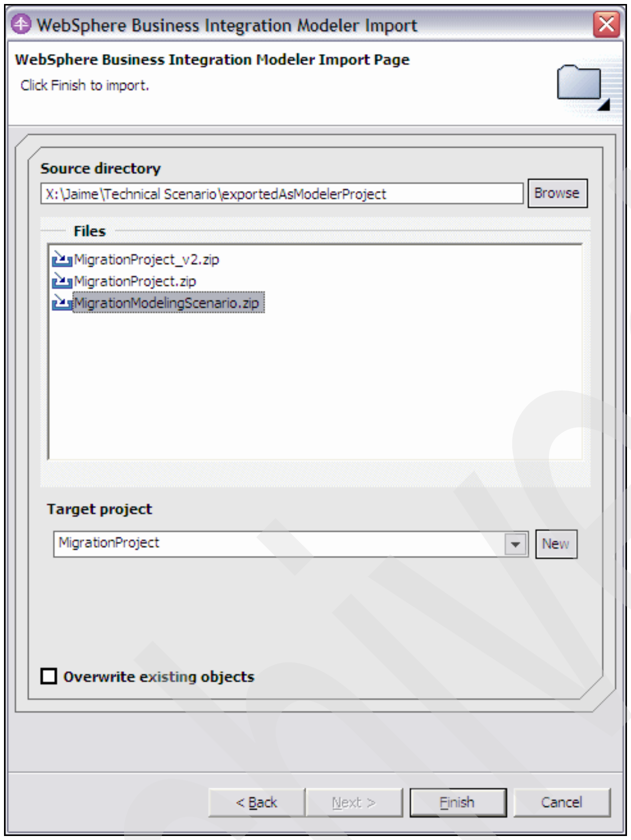
The migration scenario model has the following components:

- 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
- The process itself

9.4.2 Preparation

The WebSphere Business Integration Modeler V5 project needs to be imported in the modeler Version 5.1.1:

1. Go to Appendix A, “Additional material” on page 285 to download the material.
2. Unzip the source file, and look for the *MigrationProjectFrom5To6.zip* file.
3. Import the model as usual.
4. In the modeler, select **File** → **Import** → **WebSphere Business Integration Modeler Import** → **WebSphere Business Integration Modeler V5 Project.**
5. Choose the folder where you have the MigrationModelingScenario.zip file, and select this file.
6. Import in a new project.
7. The new service project uses the default *Catalog* name for *Business Items* and for *Process catalogs*, and the imported project uses a different name. The import wizard shows a message window, which indicates that these catalogs need to be combined. Answer **yes** to the prompt that asks for this merge. See Figure 9-48 on page 248.


WebSphere Business Integration Modeler Import

WebSphere Business Integration Modeler Import Page

Click Finish to import.

Source directory: X:\Jaime\Technical Scenario\exportedAsModelerProject

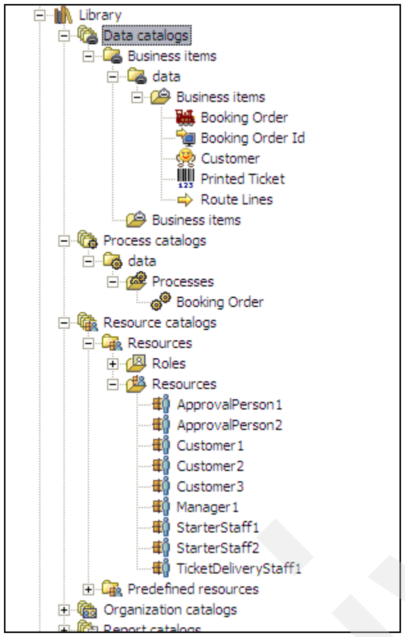
Files:
- MigrationProject_v2.zip
- MigrationProject.zip
- MigrationModelingScenario.zip

Target project: MigrationProject

☐ Overwrite existing objects

Figure 9-48 Importing model in WebSphere Business Integration Modeler V5

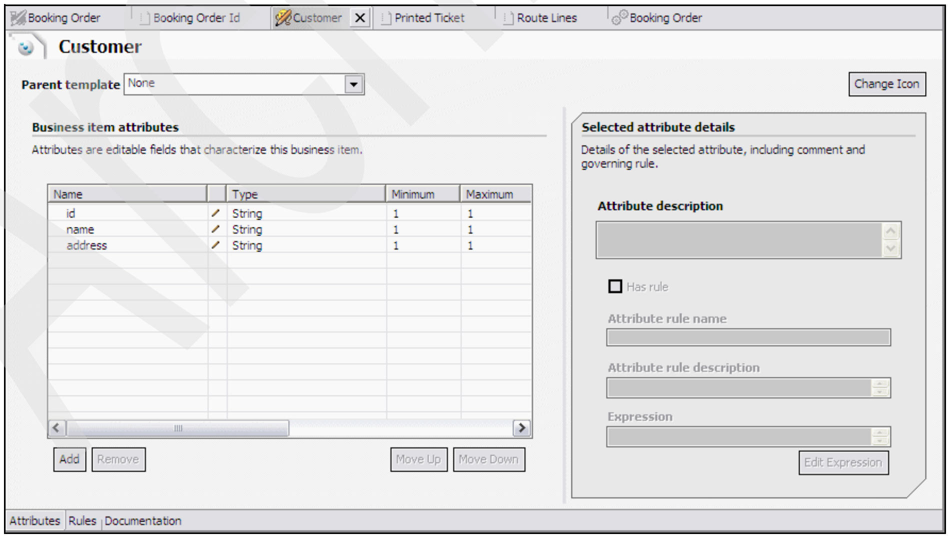
We describe the business items in your project in the Project Tree, which is in Figure 9-49 on page 249.



```
Library
  Data catalogs
    Business items
      data
        Business items
          Booking Order
          Booking Order Id
          Customer
          Printed Ticket
          Route Lines
      Business items
  Process catalogs
    data
      Processes
        Booking Order
  Resource catalogs
    Resources
      Roles
      Resources
        ApprovalPerson1
        ApprovalPerson2
        Customer1
        Customer2
        Customer3
        Manager1
        StarterStaff1
        StarterStaff2
        TicketDeliveryStaff1
    Predefined resources
  Organization catalogs
```

Figure 9-49 model to migrate Project Tree

8. Review the business items. Figure 9-50 shows the definition of the Customer entity.



Customer

Parent template: None

Change Icon

Business item attributes

Attributes are editable fields that characterize this business item.

Name	Type	Minimum	Maximum
id	String	1	1
name	String	1	1
address	String	1	1

Selected attribute details

Details of the selected attribute, including comment and governing rule.

Attribute description

Has rule

Attribute rule name

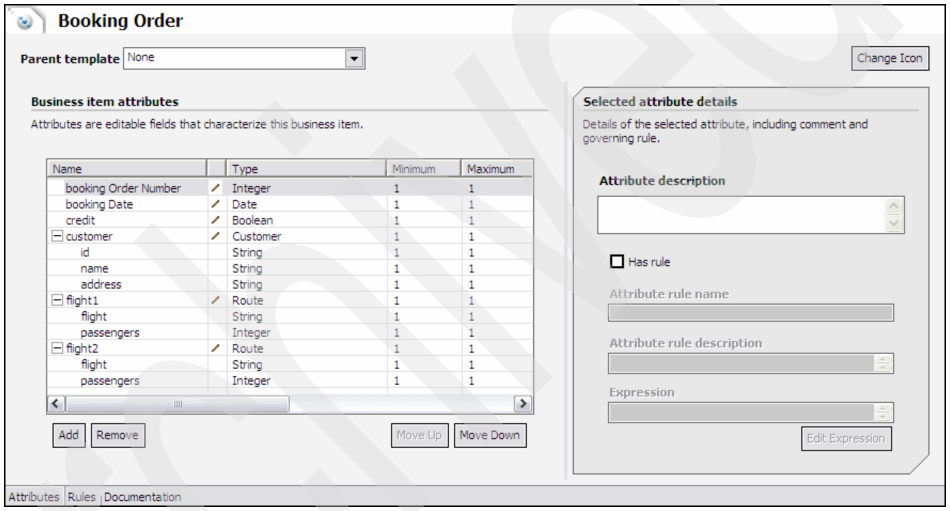
Attribute rule description

Expression

Figure 9-50 Customer business item

The booking order uses (Figure 9-51):

- A purchase order number to be provided for the customer
- The booking date
- Customer
- Route to flight information. The cardinality of this element could be `1..n`, but to make an easy test in the business process execution explorer (BPC Explorer) that is installed with WebSphere Business Integration Server Foundation, we defined a fix two lines in the order. If the multiplicity is changed to *n*, the migration can be continued without testing the process in Server Foundation as it performs without problems in WebSphere Process Server.



Booking Order

Parent template: None

Business item attributes

Attributes are editable fields that characterize this business item.

Name		Type	Minimum	Maximum
booking Order Number	✎	Integer	1	1
booking Date	✎	Date	1	1
credit	✎	Boolean	1	1
⊟ customer	✎	Customer	1	1
id		String	1	1
name		String	1	1
address		String	1	1
⊟ flight1	✎	Route	1	1
flight		String	1	1
passengers		Integer	1	1
⊟ flight2	✎	Route	1	1
flight		String	1	1
passengers		Integer	1	1

Selected attribute details

Details of the selected attribute, including comment and governing rule.

Attribute description

☐ Has rule

Attribute rule name

Attribute rule description

Expression

Figure 9-51 Booking Order business item

The process consists of the activities in Figure 9-52.

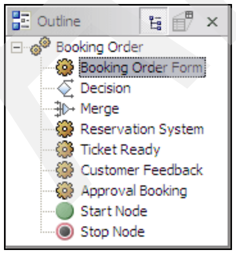
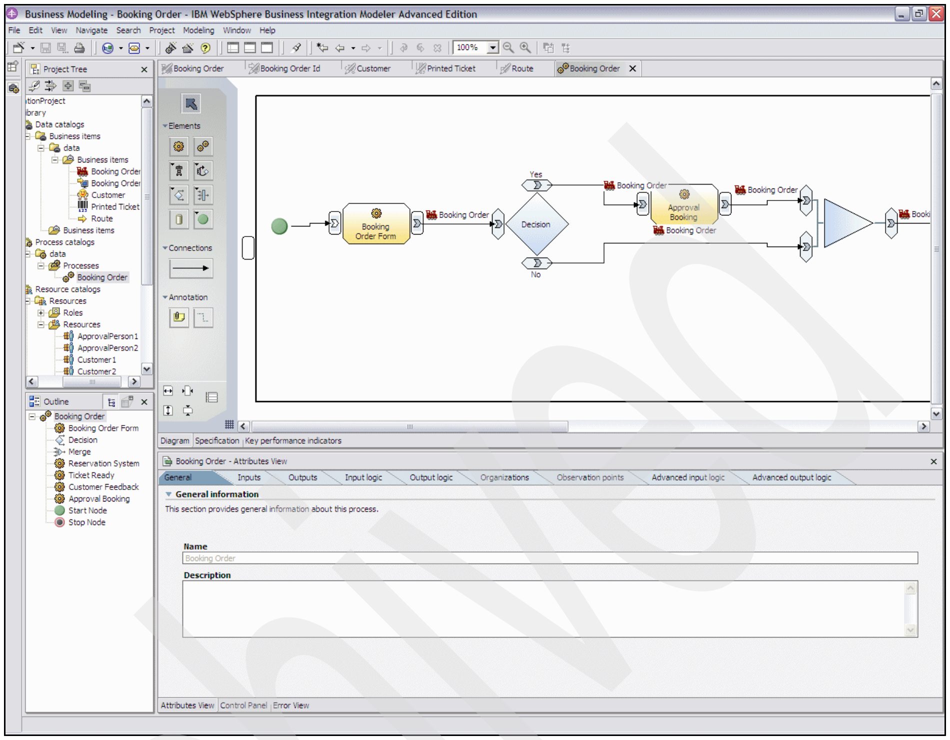


Figure 9-52 Activities in Booking Order

Figure 9-53 shows what the process looks like.



General information

This section provides general information about this process.

Name
Booking Order

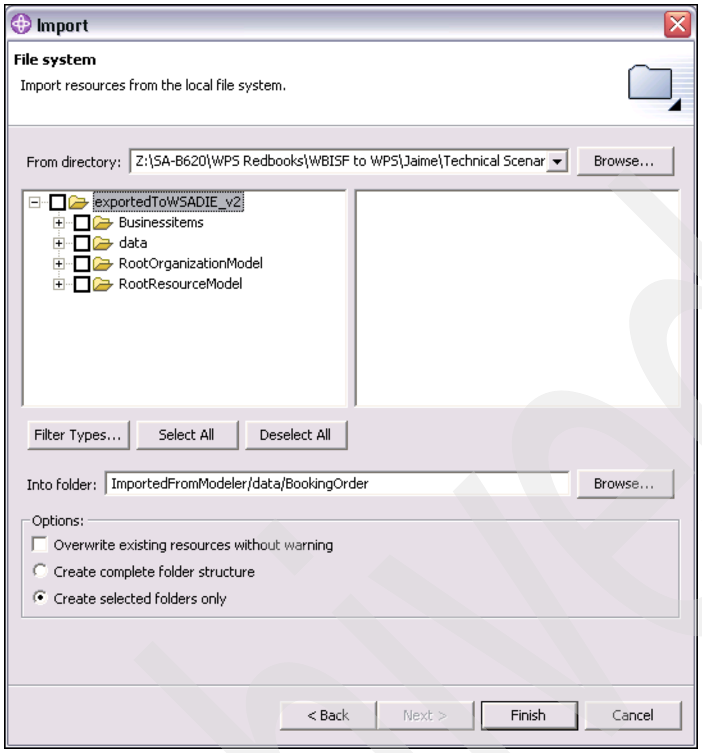
Description

Figure 9-53 Booking Order process

Running the process in Server Foundation

To run the process in Server Foundation, you must export the model to WebSphere Studio Application Developer Integration Edition:

1. Import the process **File** → **Import** → **File System** in WebSphere Studio Application Developer Integration Edition, as shown in Figure 9-54 on page 252.



From directory: Z:\SA-B620\WPS Redbooks\WBISF to WPS\Jaime\Technical Scenar

- [ ] exportedToWSADIE_v2
  - [ ] Businessitems
  - [ ] data
  - [ ] RootOrganizationModel
  - [ ] RootResourceModel

Into folder: ImportedFromModeler/data/BookingOrder

Options:
- [ ] Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

Figure 9-54 Importing process in WSADIE

2. Generate the Java service for the Reservation System activity.
3. Open the service project, and select the BookingOrderInterface.wsdl file.
4. Create a new service. From the context menu, select **New** → **Build from Service**, as shown in Figure 9-55 on page 253. The new service skeleton wizard is displayed.
5. Select a Java service skeleton.

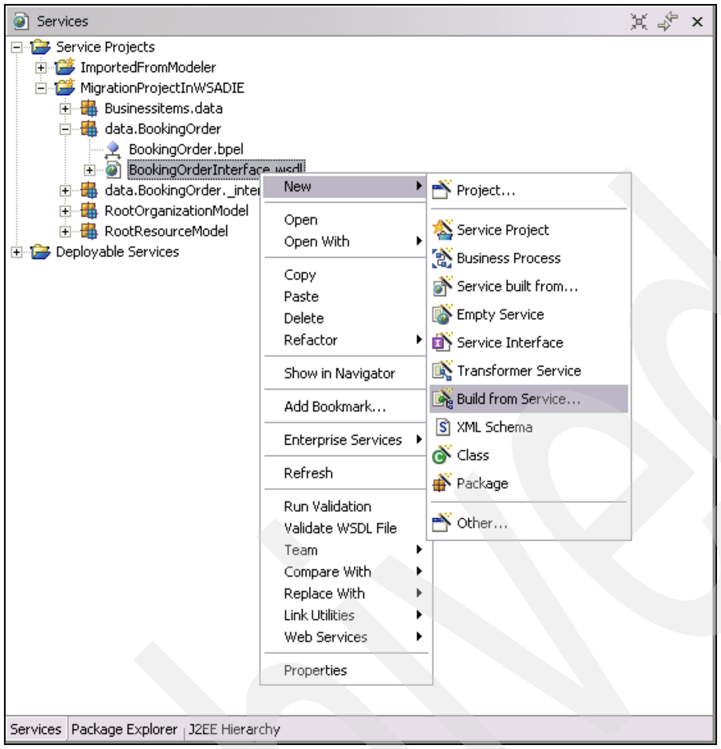
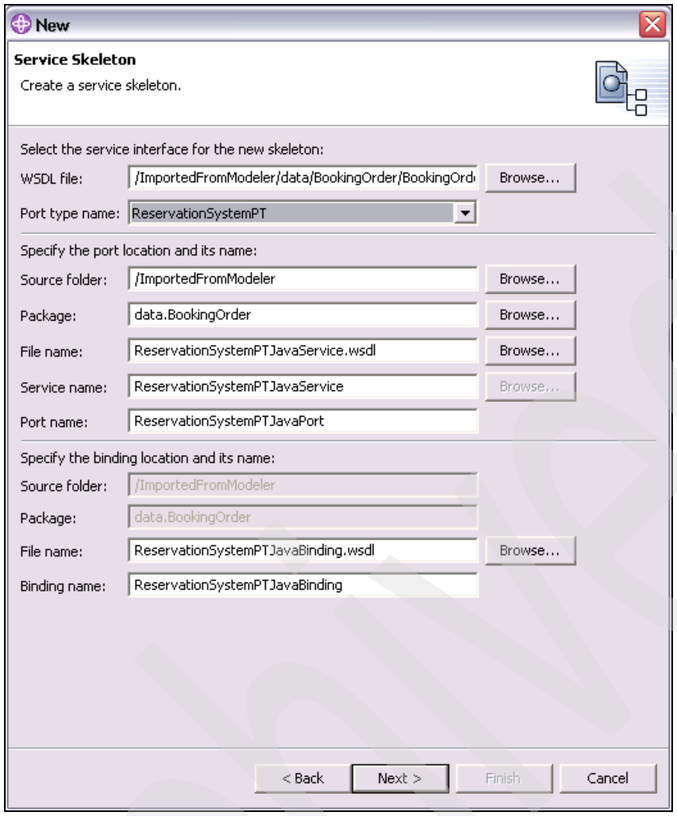


Figure 9-55 Implementing the Reservation System activity. Build Service.

6. In the Port type name, select the ReservationSystemPT, as shown in Figure 9-56 on page 254.



Select the service interface for the new skeleton:

WSDL file: /ImportedFromModeler/data/BookingOrder/BookingOrd

Port type name: ReservationSystemPT

Specify the port location and its name:

Source folder: /ImportedFromModeler

Package: data.BookingOrder

File name: ReservationSystemPTJavaService.wsdl

Service name: ReservationSystemPTJavaService

Port name: ReservationSystemPTJavaPort

Specify the binding location and its name:

Source folder: /ImportedFromModeler

Package: data.BookingOrder

File name: ReservationSystemPTJavaBinding.wsdl

Binding name: ReservationSystemPTJavaBinding

Figure 9-56 Implementing the Reservation System activity. Service port type definition.

7. Open the Java skeleton to add the code in Example 9-9.

Example 9-9 ReservationSystem activity java implementation

```
/**
    * 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. Now generate the deploy code. Select the BPEL element. From the context menu, select **Enterprise Service** → **Generate Deploy Code**, as shown in Figure 9-57 on page 256.

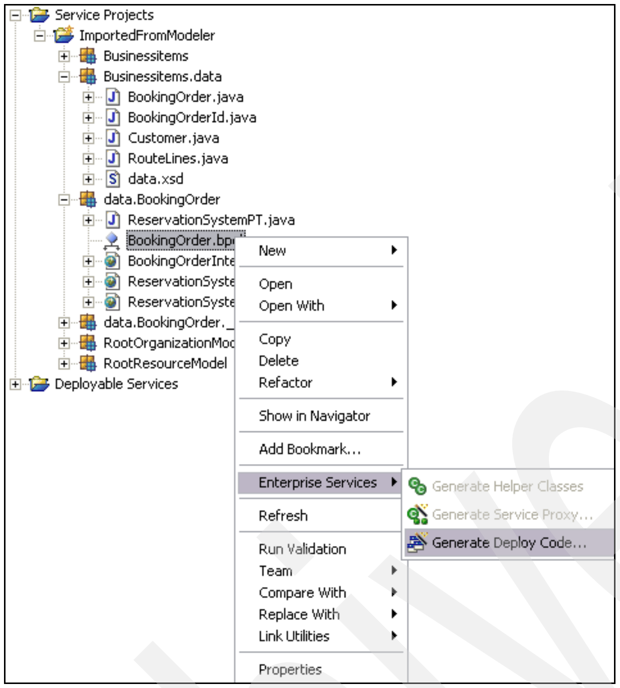
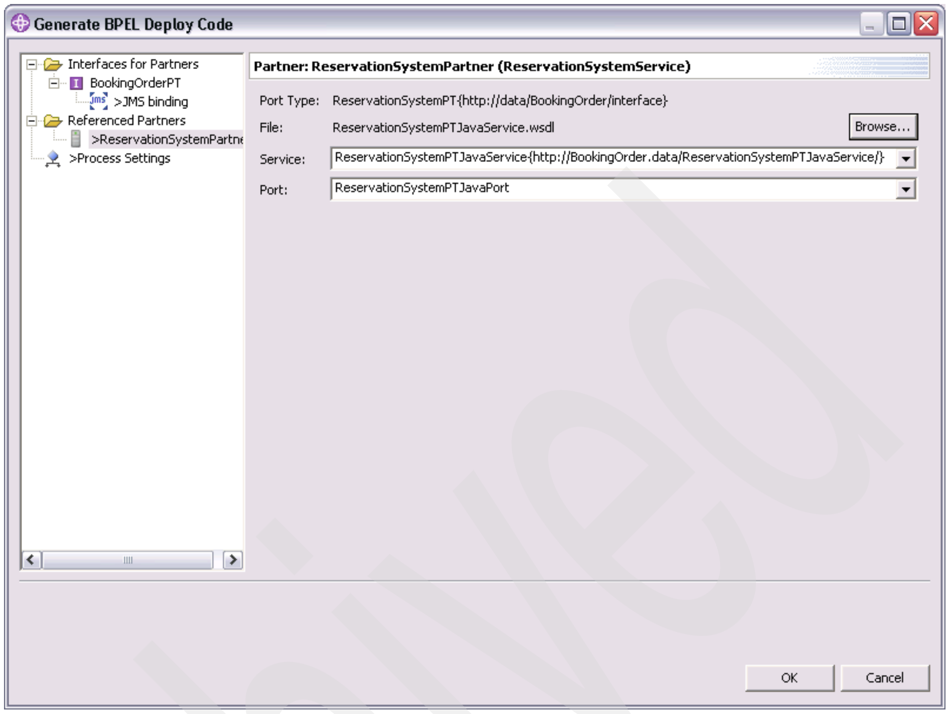


Figure 9-57 Generating the Server Foundation deploy code.

9. In the Referenced Partners section, select ReservationSystemPTJavaService.wsdl and the Service and Port that are already created. See Figure 9-58 on page 257.



Partner: ReservationSystemPartner (ReservationSystemService)

Port Type: ReservationSystemPT{http://data/BookingOrder/interface}

File: ReservationSystemPTJavaService.wsdl

Service: ReservationSystemPTJavaService{http://BookingOrder.data/ReservationSystemPTJavaService/}

Port: ReservationSystemPTJavaPort

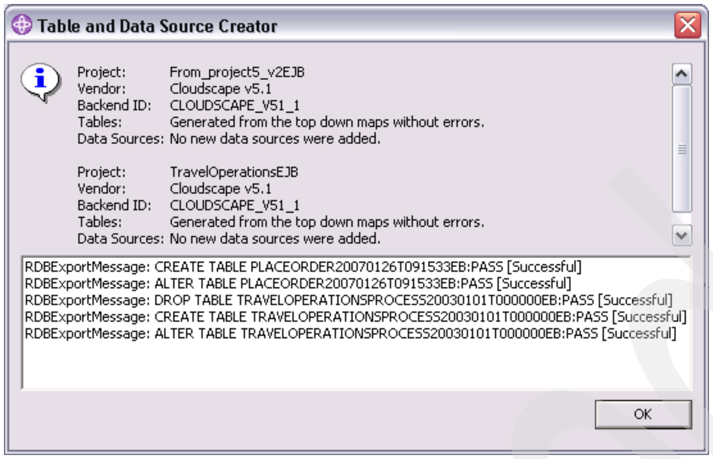
Figure 9-58 Deploy code generation. Partner selection.

In the Process setting, for the Staff provider JNDI name you can define:

```
bpe/staff/everybodyconfiguration
```

10. To deploy, remember to add the project to the server, and create tables and data sources. Select **Server** → **Add and remove projects**, and add the project. Click **Server** → **Create tables and data sources**.

    Figure 9-59 on page 258 shows the tables that were created in the server.



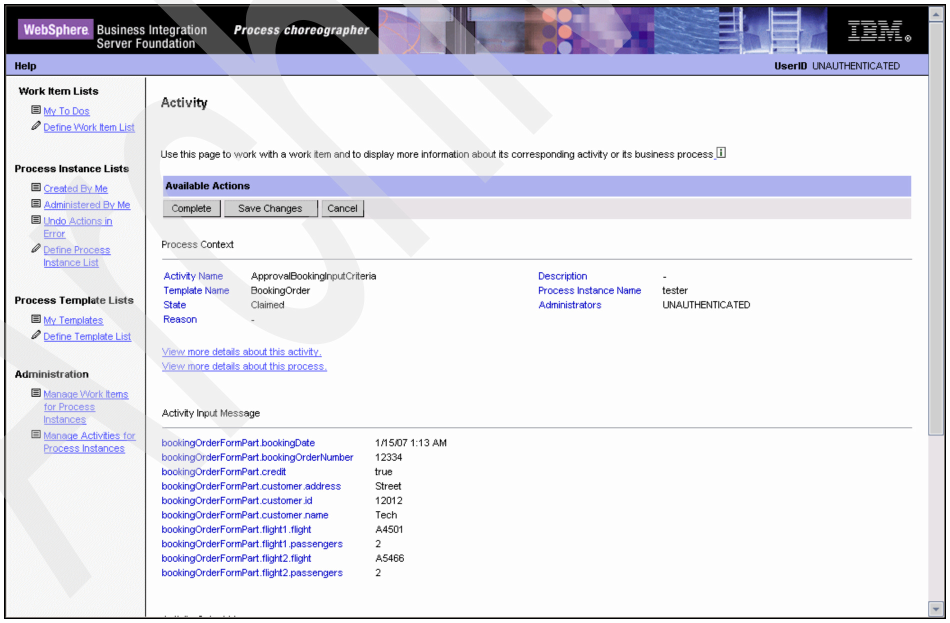
```
Project:      From_project5_v2EJB
Vendor:       Cloudscape v5.1
Backend ID:   CLOUDSCAPE_V51_1
Tables:       Generated from the top down maps without errors.
Data Sources: No new data sources were added.

Project:      TravelOperationsEJB
Vendor:       Cloudscape v5.1
Backend ID:   CLOUDSCAPE_V51_1
Tables:       Generated from the top down maps without errors.
Data Sources: No new data sources were added.
```

```
RDBExportMessage: CREATE TABLE PLACEORDER20070126T091533EB:PASS [Successful]
RDBExportMessage: ALTER TABLE PLACEORDER20070126T091533EB:PASS [Successful]
RDBExportMessage: DROP TABLE TRAVELOPERATIONSPROCESS20030101T000000EB:PASS [Successful]
RDBExportMessage: CREATE TABLE TRAVELOPERATIONSPROCESS20030101T000000EB:PASS [Successful]
RDBExportMessage: ALTER TABLE TRAVELOPERATIONSPROCESS20030101T000000EB:PASS [Successful]
```

Figure 9-59 Tables created in server

11.Start the server, and launch the Business Process Web Client by selecting **Server → Launch Business Process Web Client**.

Figure 9-60 shows the Process Navigation in Server Foundation.



UserID UNAUTHENTICATED

Activity

Use this page to work with a work item and to display more information about its corresponding activity or its business process

Process Context

Activity Name	ApprovalBookingInputCriteria	Description	-
Template Name	BookingOrder	Process Instance Name	tester
State	Claimed	Administrators	UNAUTHENTICATED
Reason	-		

Activity Input Message

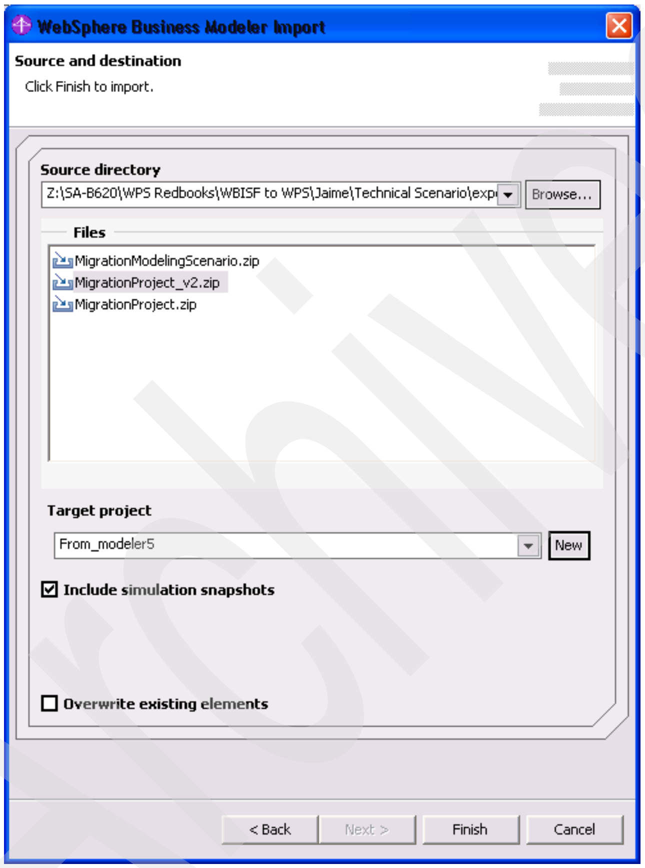
bookingOrderFormPart.bookingDate	1/15/07 1:13 AM
bookingOrderFormPart.bookingOrderNumber	12334
bookingOrderFormPart.credit	true
bookingOrderFormPart.customer.address	Street
bookingOrderFormPart.customer.id	12012
bookingOrderFormPart.customer.name	Tech
bookingOrderFormPart.flight1.flight	A4501
bookingOrderFormPart.flight1.passengers	2
bookingOrderFormPart.flight2.flight	A5466
bookingOrderFormPart.flight2.passengers	2

Figure 9-60 Process Navigation in Server Foundation

9.4.3 Implementation

Start the migration process to Modeler Version 6. In WebSphere Business Integration Modeler V5, export the modeler project as a *Modeler project*, or open WebSphere Business Modeler V6 in the old workspace.

Import the project by selecting the path of the exported project, as shown in Figure 9-61.



Source directory: Z:\SA-B620\WPS Redbooks\WBISF to WPS\Jaime\Technical Scenario\exp

Files
- MigrationModelingScenario.zip
- MigrationProject_v2.zip
- MigrationProject.zip

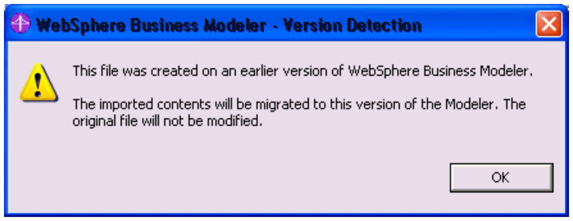
Target project: From_modeler5

☑ Include simulation snapshots

☐ Overwrite existing elements

Figure 9-61 Importing the project version 5.1.1 in modeler v6.0.2

Modeler shows an alert window that tells you that this is a migration from an early version project, as shown in Figure 9-62 on page 260.

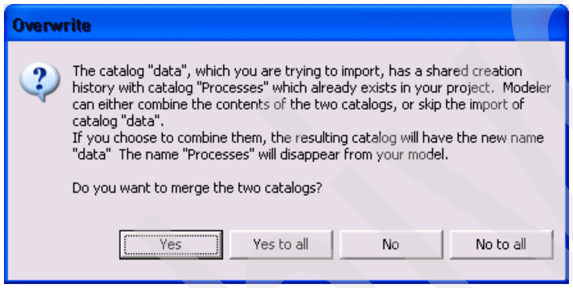


This file was created on an earlier version of WebSphere Business Modeler.

The imported contents will be migrated to this version of the Modeler. The original file will not be modified.

Figure 9-62   Migration message

Because your project is using a different catalog name to the default, a prompt window (Figure 9-63) appears asking to combine the catalog in your imported project with the catalog in the new project. Click **Yes**.

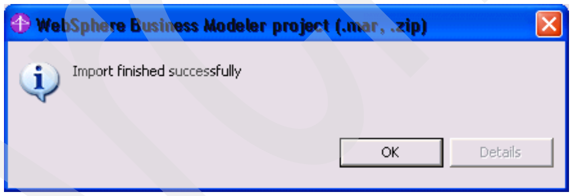


The catalog "data", which you are trying to import, has a shared creation history with catalog "Processes" which already exists in your project. Modeler can either combine the contents of the two catalogs, or skip the import of catalog "data".
If you choose to combine them, the resulting catalog will have the new name "data" The name "Processes" will disappear from your model.

Do you want to merge the two catalogs?

Figure 9-63   Combine catalogs

Finally, you receive the confirmation of the migration, as shown in Figure 9-64.



Import finished successfully

Figure 9-64   Import finished

Look for errors and warnings in the Error view.

Migration process log

The migration tool from Modeler Version 5 to Version 6 generates a log file located, which is located in:

project directory/.metadata/WBModelerMigration.log

Example 9-10 on page 261 provides an example of the WBModelerMigration log.

Example 9-10 Example of WBModelerMigration.log

```
INFO MLM_2500:Entry into
IMigrationManager.prepareProject(C:\ITSO\workspace_from5to6\ImportedFromV5). additional
info:
INFO MLM_2501:Exit from IMigrationManager.prepareProject(). additional info:
INFO MLM_2502:Entry into
IMigrationManager.preprocessFiles(C:\ITSO\workspace_from5to6\MigrationProjectFrom5To6).
additional info:
INFO BOMMC0100:Created Ecore file handle for c:\Program
Files\IBM\WebSphere\Modeler6\eclipse\plugins\com.ibm.btools.blm.migration.contributor_6.
0.2.000\model\511\expression.bom.model.ecore. additional info:
INFO BOMMC0100:Created Ecore file handle for c:\Program
Files\IBM\WebSphere\Modeler6\eclipse\plugins\com.ibm.btools.blm.migration.contributor_6.
0.2.000\model\511\expression.model.ecore. additional info:
...
INFO MLM_2503:Exit from IMigrationManager.preprocessFiles(). additional info:
INFO MLM_2504:Entry into
IMigrationManager.completePreprocessing(C:\ITSO\workspace_from5to6\ImportedFromV5).
additional info:
INFO MLM_2505:Exit from IMigrationManager.completePreprocessing(). additional info:
```

Figure 9-65 shows that the Modeler shows the imported model.

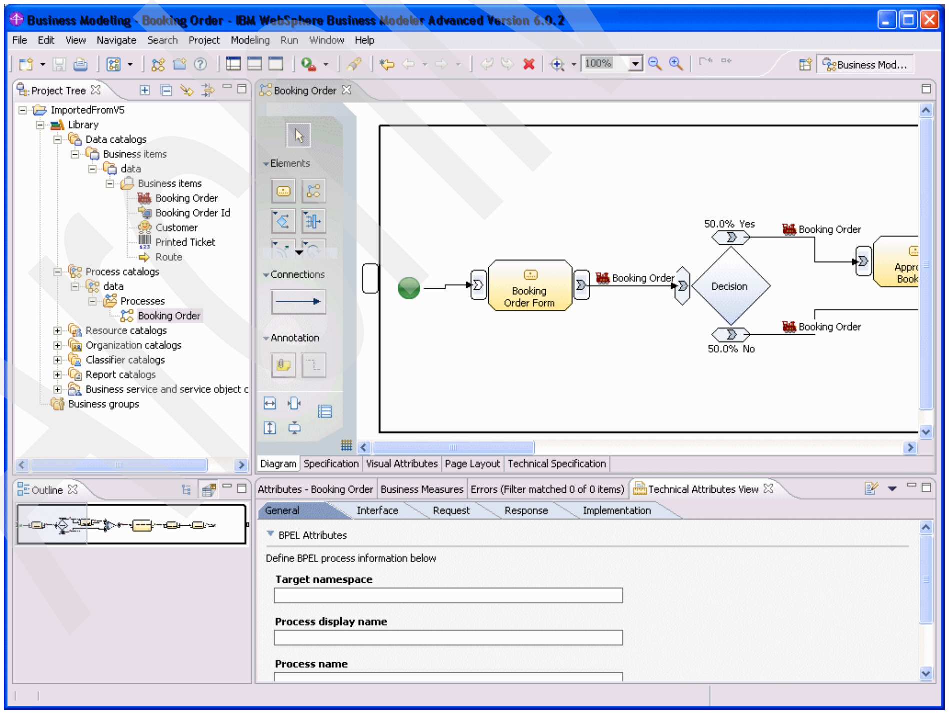
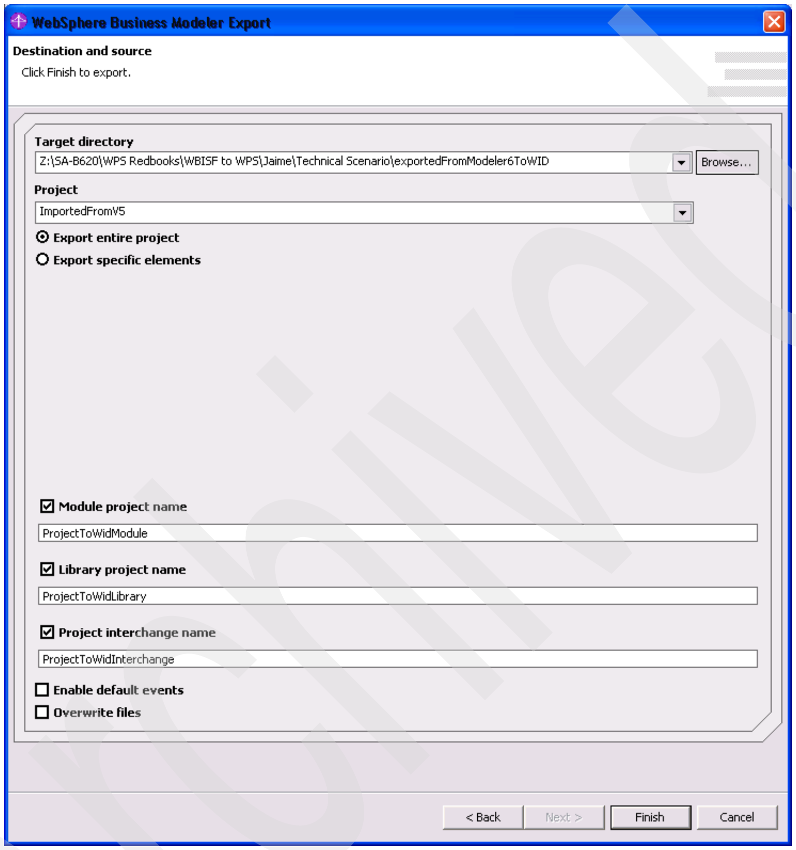

Business Modeling - Booking Order - IBM WebSphere Business Modeler Advanced Version 6.0.2
File Edit View Navigate Search Project Modeling Run Window Help
Project Tree
ImportedFromV5
Library
Data catalogs
Business items
data
Business items
Booking Order
Booking Order Id
Customer
Printed Ticket
Route
Process catalogs
data
Processes
Booking Order
Resource catalogs
Organization catalogs
Classifier catalogs
Report catalogs
Business service and service object c
Business groups
Booking Order
Elements
Connections
Annotation
Booking Order Form
Booking Order
Decision
50.0% Yes
50.0% No
Booking Order
Booking Order
Diagram Specification Visual Attributes Page Layout Technical Specification
Outline
Attributes - Booking Order Business Measures Errors (Filter matched 0 of 0 items) Technical Attributes View
General Interface Request Response Implementation
BPEL Attributes
Define BPEL process information below
Target namespace
Process display name
Process name

Figure 9-65 Model in WebSphere Business Modeler V6

Exporting to WebSphere Integration Developer

To export to WebSphere Integration Developer:

1. Export the project to WebSphere Integration Developer as an Interchange Project, as shown in Figure 9-66.



Target directory: Z:\SA-B620\WPS Redbooks\WBISF to WPS\Jaime\Technical Scenario\exportedFromModeler6ToWID

Project: ImportedFromV5

[✓ Export entire project] Export specific elements

[✓] Module project name: ProjectToWidModule

[✓] Library project name: ProjectToWidLibrary

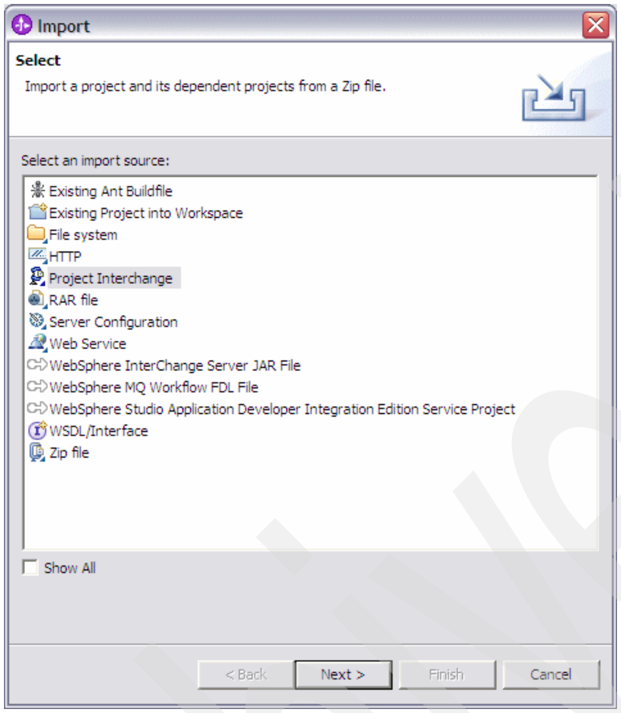
[✓] Project interchange name: ProjectToWidInterchange

[ ] Enable default events

[ ] Overwrite files

Figure 9-66 Exported as Project Interchange V6

2. Import the Interchange project in WebSphere Integration Developer, as shown in Figure 9-67 on page 263.


Import

Select

Import a project and its dependent projects from a Zip file.

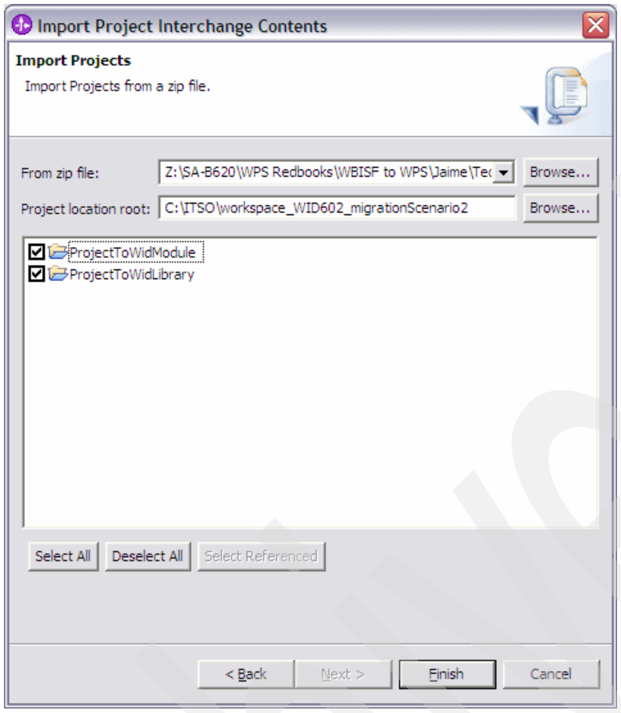
Select an import source:

- Existing Ant Buildfile
- Existing Project into Workspace
- File system
- HTTP
- Project Interchange
- RAR file
- Server Configuration
- Web Service
- WebSphere InterChange Server JAR File
- WebSphere MQ Workflow FDL File
- WebSphere Studio Application Developer Integration Edition Service Project
- WSDL/Interface
- Zip file

Show All

Figure 9-67 Project Interchange Import in WebSphere Integration Developer

3. Select the module and the library projects, as shown in Figure 9-68 on page 264.


Import Project Interchange Contents

Import Projects

Import Projects from a zip file.

From zip file: Z:\SA-B620\WPS Redbooks\WBISF to WPS\Jaime\Tec

Project location root: C:\ITSO\workspace_WID602_migrationScenario2

☑ ProjectToWidModule
☑ ProjectToWidLibrary

Figure 9-68 Import projects in WebSphere Integration Developer module and Library.

4. Open the Assembly Diagram in the module. This editor and the Problems view show that the ReservationSystem has no implementation yet. Over the ReservationSystem element, from the context menu, select **Generate Implementation** → **Java**, as shown in Figure 9-69 on page 265.

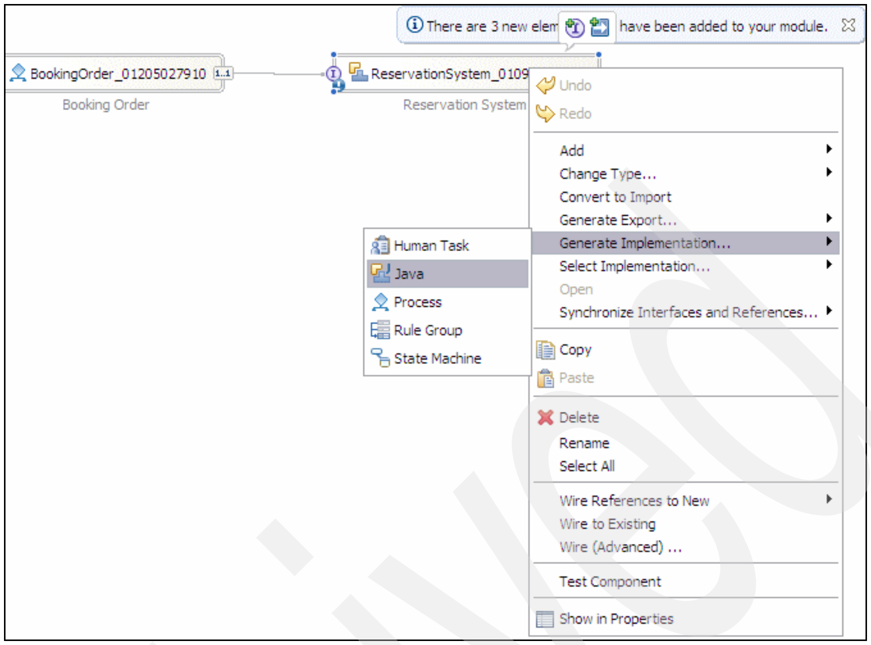
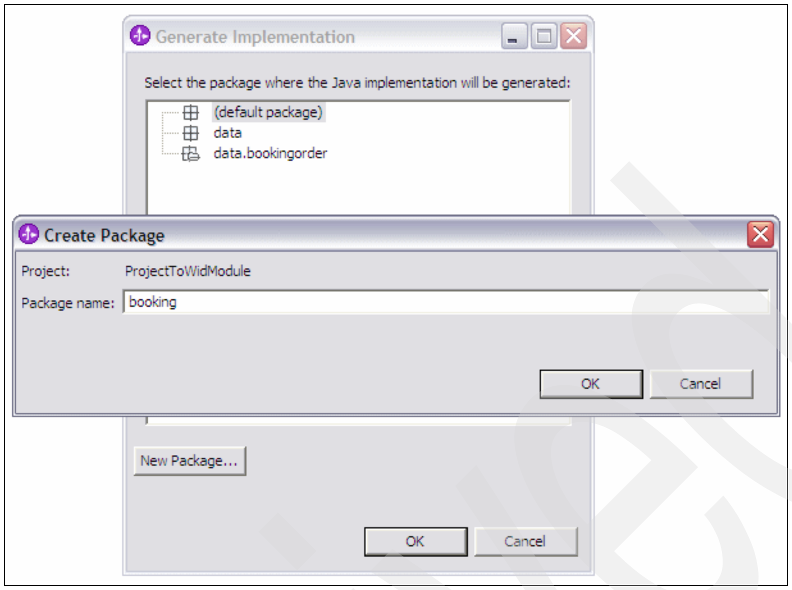


Figure 9-69 Generating the Java Implementation

5. Create and select a new package booking, as shown in Figure 9-70 on page 266.



Project: ProjectToWidModule

Package name: booking

Figure 9-70 Generating the Java Implementation. Select the package

6. The Implementation Java file opened. Look for the InputCriteria method and add the code that is in Example 9-11.

Example 9-11 InputCriteria java implementation

```
public DataObject InputCriteria(DataObject reservationInput) {
  int bookingId = reservationInput.getInt("bookingOrderNumber");
  System.out.println("the input is the " + bookingId + " booking 
order.");

  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://Businessitems/data",
      "BookingOrderId");
  output.setString("id", "Booking: " + bookingId);
  System.out.println(" setting the output " + output);
  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(" > flight " + orderLine.getString("flight"));
  System.out.println(" > passengers " +
orderLine.getInt("passengers"));
}
```

7. To test the process without security configuration, change the JNDI name of staff plugin configuration value to *bpe/staff/everybodyconfiguration* for each Human task activity in the process.
8. Save, and review for errors. Assure that the **Project** → **Build Automatically** and the **Project** → **Generate Deploy Code In Build** are checked or rebuilt, and generate the deploy code.
9. Assure that the server is started, and add the project **Server** → **Add and remove projects**, as shown in Figure 9-71 on page 268.

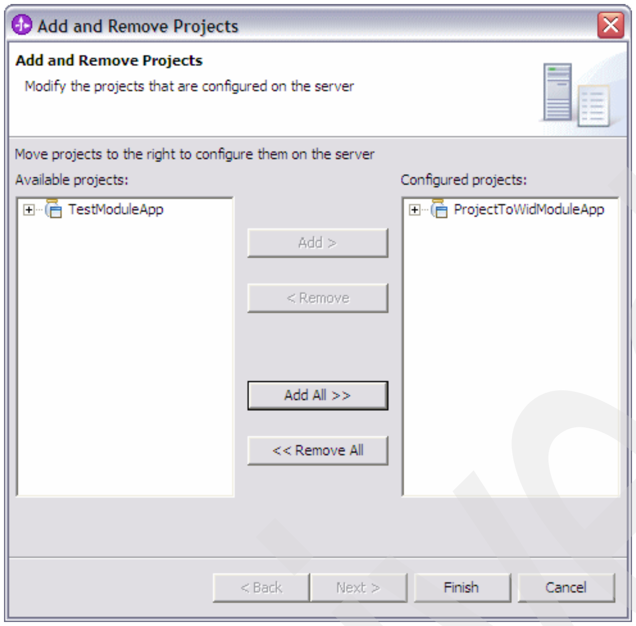
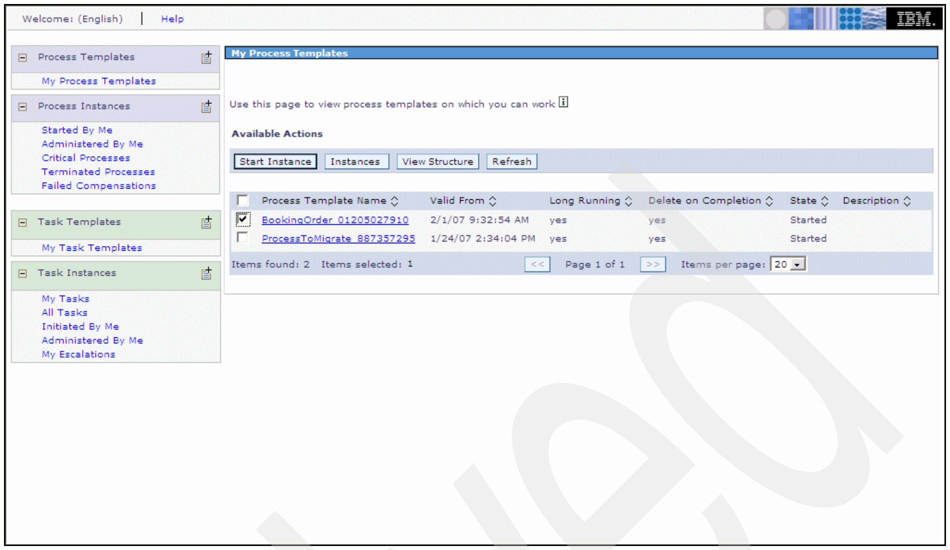


Figure 9-71 Adding the project to WebSphere Process Server

9.4.4 Verification

To verify:

1. Launch the BPC Explorer by selecting **Server** → **Launch** → **BPCExplorer**, and click the **My Process Templates** link. Select the BookingOrder process, and click **Start Instance**, as shown in Figure 9-72 on page 269.



Welcome: (English) | Help

Process Templates
My Process Templates
Process Instances
Started By Me
Administered By Me
Critical Processes
Terminated Processes
Failed Compensations
Task Templates
My Task Templates
Task Instances
My Tasks
All Tasks
Initiated By Me
Administered By Me
My Escalations

My Process Templates

Use this page to view process templates on which you can work

Available Actions

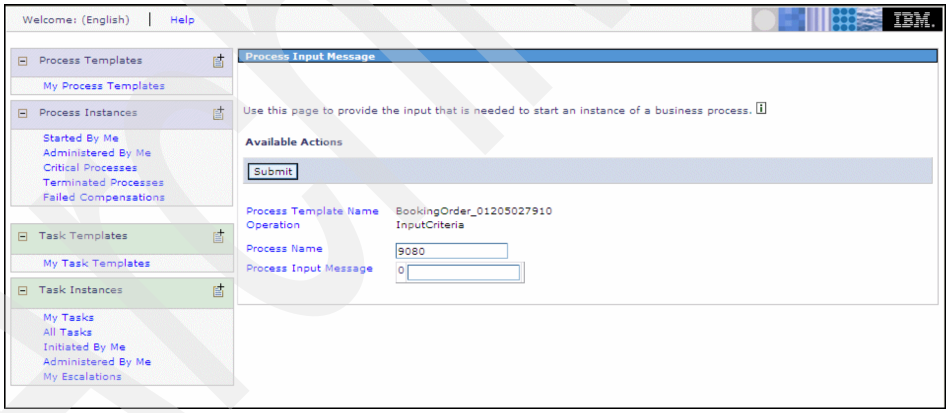
Start Instance | Instances | View Structure | Refresh

	Process Template Name	Valid From	Long Running	Delete on Completion	State	Description
✓	BookingOrder_01205027910	2/1/07 9:32:54 AM	yes	yes	Started	
	ProcessToMigrate_887357295	1/24/07 2:34:04 PM	yes	yes	Started	

Items found: 2 Items selected: 1 Page 1 of 1 Items per page: 20

Figure 9-72 Starting instance of the process

2. Complete the name of the process instance (only), as shown in Figure 9-73.



Welcome: (English) | Help

Process Templates
My Process Templates
Process Instances
Started By Me
Administered By Me
Critical Processes
Terminated Processes
Failed Compensations
Task Templates
My Task Templates
Task Instances
My Tasks
All Tasks
Initiated By Me
Administered By Me
My Escalations

Process Input Message

Use this page to provide the input that is needed to start an instance of a business process.

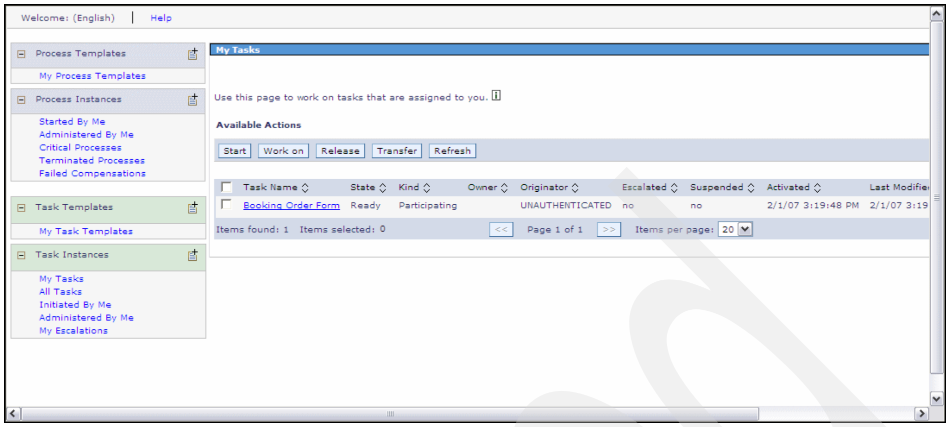
Available Actions

Submit

Process Template Name	BookingOrder_01205027910
Operation	InputCriteria
Process Name	9080
Process Input Message	0

Figure 9-73 Process Name

3. Select the My Tasks link, as shown in Figure 9-74 on page 270.



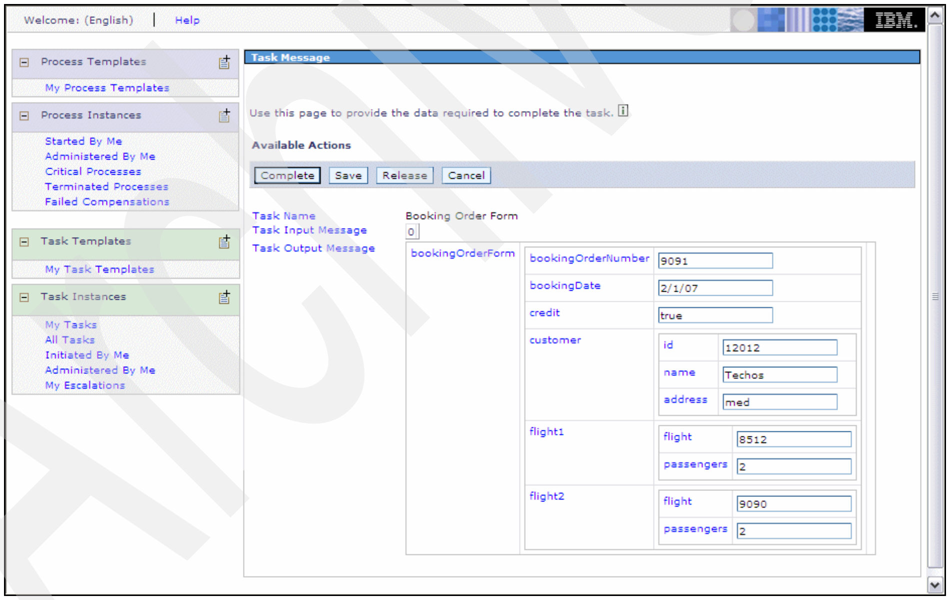
Use this page to work on tasks that are assigned to you.

	Task Name	State	Kind	Owner	Originator	Escalated	Suspended	Activated	Last Modifie
☐	Booking Order Form	Ready	Participating		UNAUTHENTICATED	no	no	2/1/07 3:19:48 PM	2/1/07 3:19

Items found: 1 Items selected: 0 Page 1 of 1 Items per page: 20

Figure 9-74 Initial activity in process

4. Select the human task activity **Booking order Form**, and click **Work on.** Complete the initial form, and click **Complete**, as shown in Figure 9-75.



Use this page to provide the data required to complete the task.

Task Name: Booking Order Form
Task Input Message: 0
Task Output Message:

bookingOrderForm
- bookingOrderNumber: 9091
- bookingDate: 2/1/07
- credit: true
- customer
  - id: 12012
  - name: Techos
  - address: med
- flight1
  - flight: 8512
  - passengers: 2
- flight2
  - flight: 9090
  - passengers: 2

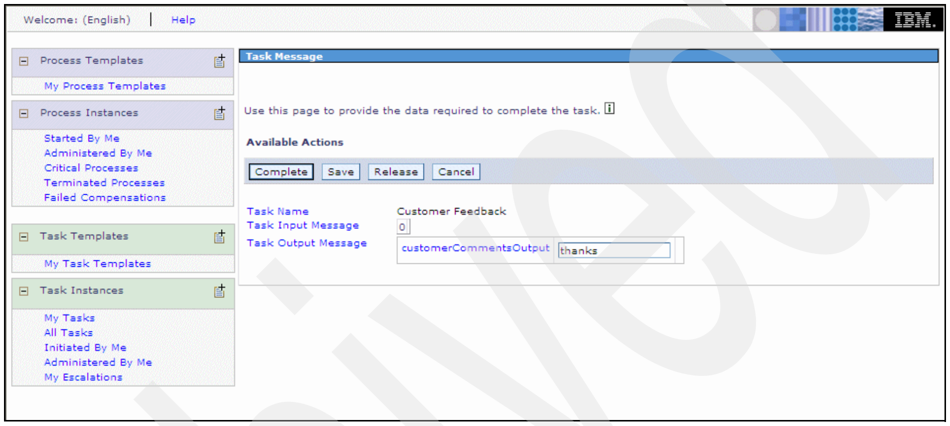
Figure 9-75 Initial form.

5. Invoke the next step (Approval Booking), and click **Complete**. The Java service is invoked. Review the console to view the System.out print. Example 9-12 on page 271 shows an example of the output.

Example 9-12 System.out of the "Registration System" call

```
    [2/1/07 15:38:21:213 PST] 00000079 SystemOut     O the input is the 8082 booking order.
    [2/1/07 15:38:21:213 PST] 00000079 SystemOut     O  > flight 8082 > passengers 2
    [2/1/07 15:38:21:213 PST] 00000079 SystemOut     O  > flight 8280 > passengers 2
    [2/1/07 15:38:21:213 PST] 00000079 SystemOut     O  setting the output BO type =
    BookingOrderId,id = Booking: 8082,
```

6. Finish the process with the human task activity, as shown in Figure 9-76.



Use this page to provide the data required to complete the task.

Task Name	Customer Feedback
Task Input Message	0
Task Output Message	customerCommentsOutput thanks

Figure 9-76 Final human task activity

This technical scenario illustrates the migration from a model in WebSphere Business Integration Modeler to WebSphere Process Server.

10

Troubleshooting

In this chapter, we discuss troubleshooting for migrating WebSphere Business Integration Server Foundation V5.1 to WebSphere Process Server. The chapter is divided into the following sections:

- 10.1, “General” on page 274
- 10.2, “Scenarios” on page 277
- 10.3, “Advanced” on page 282

10.1 General

In this section, we introduce the general troubleshooting methodology of the migration path.

10.1.1 Import into WebSphere Integration Developer

If the source artifact migration fails while importing from WebSphere Studio Application Developer Integration Edition, the following topics are some of the possible sources for troubleshooting

Log File

If you receive the message in Example 10-1while migrating original Integration Edition, check the WebSphere Integration Developer log file in the .metadata folder of the new workspace to see the details of the error. If possible, resolve the cause of the error, delete the module that was created in the new workspace, and try the migration again.

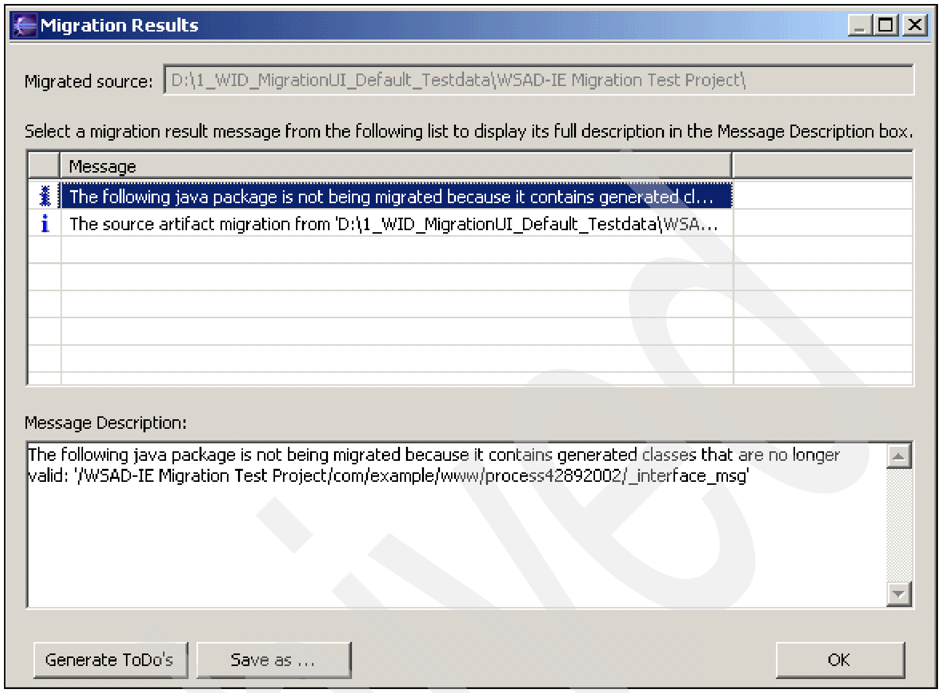
Example 10-1 Migrating original Integration Edition

```
“Migration Error Message”
Reason: Fatal Migration Failure
Message: Contact your IBM Representative
```

Migration Results Windows

If the Migration wizard completes without the message in Example 10-1, 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.

Figure 10-1 on page 275 is displayed if migration messages were generated during the migration process.


Migration Results

Migrated source: D:\1_WID_MigrationUI_Default_Testdata\WSAD-IE Migration Test Project\

Select a migration result message from the following list to display its full description in the Message Description box.

	Message
	The following java package is not being migrated because it contains generated cl...
i	The source artifact migration from 'D:\1_WID_MigrationUI_Default_Testdata\WSA...

Message Description:

The following java package is not being migrated because it contains generated classes that are no longer valid: '/WSAD-IE Migration Test Project/com/example/www/process42892002/_interface_msg'

Figure 10-1 Migration Results window.

In the Migration Results window (Figure 10-1), the migration messages that were generated during the migration process are displayed. By selecting a message from the upper Message list, you can find more information regarding that message in the lower Message Description window.

To keep all messages for future reference:

1. Click the **`Generate ToDo's`** button to create a list of "ToDo" tasks in the task view.
2. Click the **`Save as`** button to save the messages in a text file in the filesystem.
3. Examine each message to see if you need to take any action to immediately fix an artifact that couldn’t be fully migrated.

To verify that a portion of the migration is complete:

1. Switch to the Business Integration perspective, and ensure that all processes and WSDL interfaces from the old service project appear 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, and configure 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 it did on the previous runtime environment.

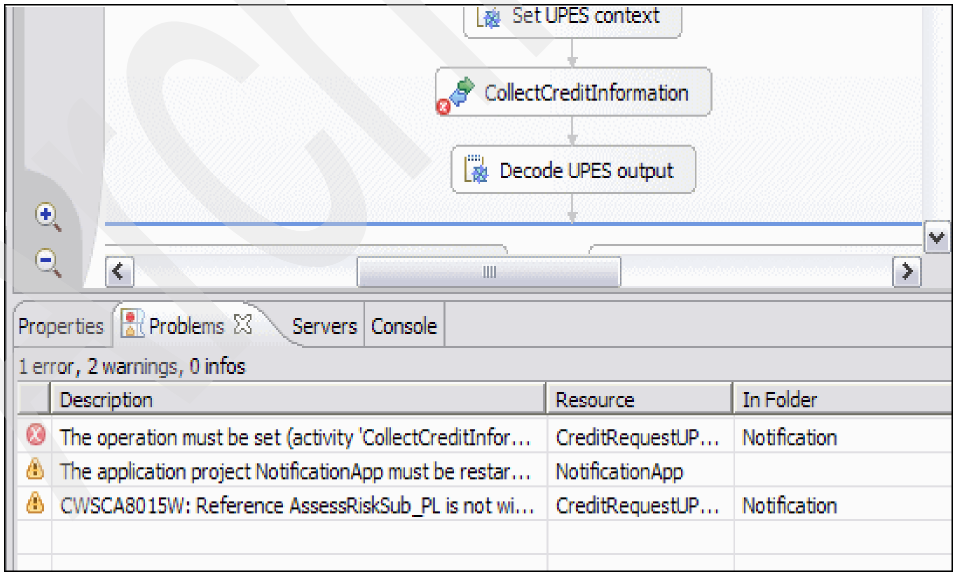
Refer to Chapter 6, “Migrating build time artifacts” on page 91 for the required manual migration steps.

10.1.2 Build the project in WebSphere Integration Developer

After you import the Integration Edition project into WebSphere Integration Developer, a new project is generated.

To build this project:

1. Choose the **Project** → **Build Automatically** option in the menu; otherwise, build the project by clicking **Project** → **Build Project** in the menu. After you build the project, some errors may be reported in the *Problems* view as shown in Figure 10-2.



1 error, 2 warnings, 0 infos

Description	Resource	In Folder
The operation must be set (activity 'CollectCreditInfor...	CreditRequestUP...	Notification
The application project NotificationApp must be restar...	NotificationApp	
CWSCA8015W: Reference AssessRiskSub_PL is not wi...	CreditRequestUP...	Notification

Figure 10-2 WebSphere Integration Developer “Problems” View.

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 there is a red flag for error or a yellow flag for warning on the error item.
3. Read the error message description, and check the properties of the error item.

Sometimes the problems are just caused by the confliction 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 may be resolved.

10.1.3 Deploy and run the process in WebSphere Process Server

While deploying or running the process in WebSphere Process Server, some problems may occur. The logs of WebSphere Process Server are the most pivotal key to diagnose the problem. The logs are located in the following directory:

WPS_install/profiles/*profile_name*/logs/*server_name*

Generally there are eight files in this directory:

- ServerName.pid: The pid of the java process of WebSphere Process Server
- SystemOut.log: Log of all activity within the system
- SystemErr.log: Record system errors
- startServer.log: Log of start server events
- stopServer.log : Log of stop server events
- native_stdout.log: stdout of java process
- native_stderr.log: stderr of java process
- trace.log: Log of all traced events within the system

The SystemOut.log and SystemErr.log are useful for diagnosing the problem of deploying and running the process.

10.2 Scenarios

In this section, we discuss the problems that you might 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 V6.0.2.
- WebSphere Integration Developer V6.0.2.
- WebSphere Process Server V6.0.2.

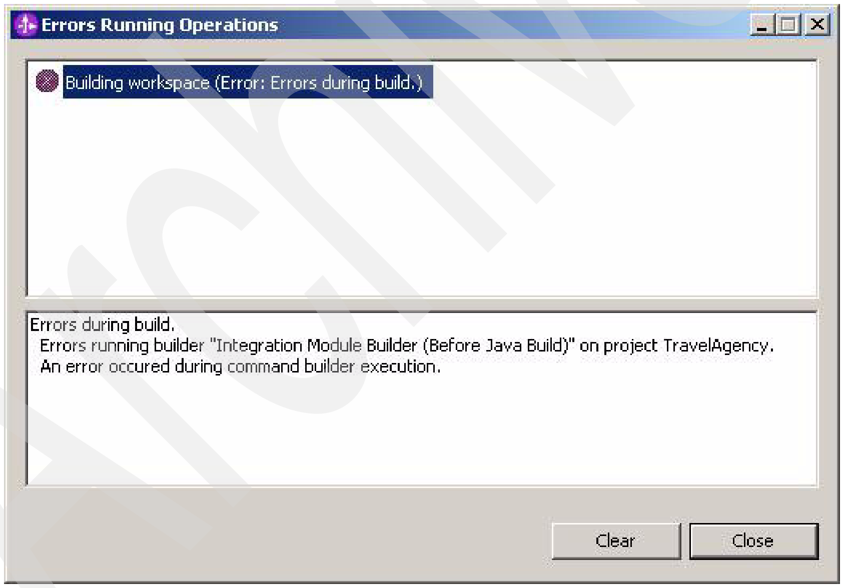
Note: It is very import to follow the same product version to replay the scenarios. So when you encounter any problems, check the product version first.

Build errors in Integration Developer

We explain the solution to this problem in three topics: problem, cause, and solution.

Problem

After the Integration Developer Migration Wizard ran successfully, unresolvable build errors occurred, as shown in Figure 10-3. Integration Developer cannot resolve a WSDL message.


Errors Running Operations
Building workspace (Error: Errors during build.)
Clear
Close

```
Errors during build.
  Errors running builder "Integration Module Builder (Before Java Build)" on project TravelAgency.
  An error occured during command builder execution.
```

Figure 10-3 Error message in Integration Developer

You can also find the error in the workspace directory in the .metadata/.log file.

Example 10-2 on page 279 shows an exception in the .log file of the Integration Developer's workspace.

Example 10-2 Example exception in the .log file of the Integration Developer workspace

```
!MESSAGE Cannot resolve WSDL message
{http://email.process.scb.com/MailSMTP/}sendEmailRequest
!STACK 0
java.lang.RuntimeException: Cannot resolve WSDL message
{http://email.process.scb.com/MailSMTP/}sendEmailRequest
   at
com.ibm.ws.sca.internal.scdl.wsdl.impl.WSDLEPackageImpl$WSDLEOperationImpl.getEClassi
fier(WSDLEPackageImpl.java:296)
   at
com.ibm.ws.sca.internal.scdl.wsdl.impl.WSDLEPackageImpl$WSDLEOperationImpl.getEParame
ters(WSDLEPackageImpl.java:371)
   at
com.ibm.wsspi.sca.scdl.impl.OperationTypeAdapterImpl.getInputType(OperationTypeAdapte
rImpl.java:92)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.getInputTypes(Unkn
own Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.getMatchingMethod(
Unknown Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.getMatchingMethod(
Unknown Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.validateInterfaceM
ethodImplementation(Unknown Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.validateJavaImplem
entation(Unknown Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.validate(Unknown
Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand.access$0(Unknown
Source)
   at
com.ibm.wbit.internal.java.validator.JavaComponentValidatorCommand$1.run(Unknown
Source)
```

Cause

There is a known issue in Integration Developer 6.0.2 that causes this build error.

Solution

Install APAR JR25914 to solve this problem and rebuild the workspace.

Exception CWWBE0085E observed

We explain the solution to this problem in three topics: Problem, Cause, and Solution.

Problem

Exception CWWBE0085E is observed during the execution of the migrated process in Process Server. Example 10-3 shows an example of the runtime exception.

Example 10-3 Exception while running TravelOperationsProcess

```
CWWBE0085E: Incompatible types encountered during assignment to a
variable or partner link 'ApproveRequest' in activity
'PrepareApprovalRequest'.com.ibm.bpe.api.EngineIncompatibleTypesExcepti
on: CWWBE0088E: An incompatible object was encountered during
assignment to part 'expectedCost'.     java.lang.ClassCastException:
The value of type 'class java.lang.Double' must be of type 'class
java.lang.Long'
```

The process instance finishes in a failed state after that exception occurred.

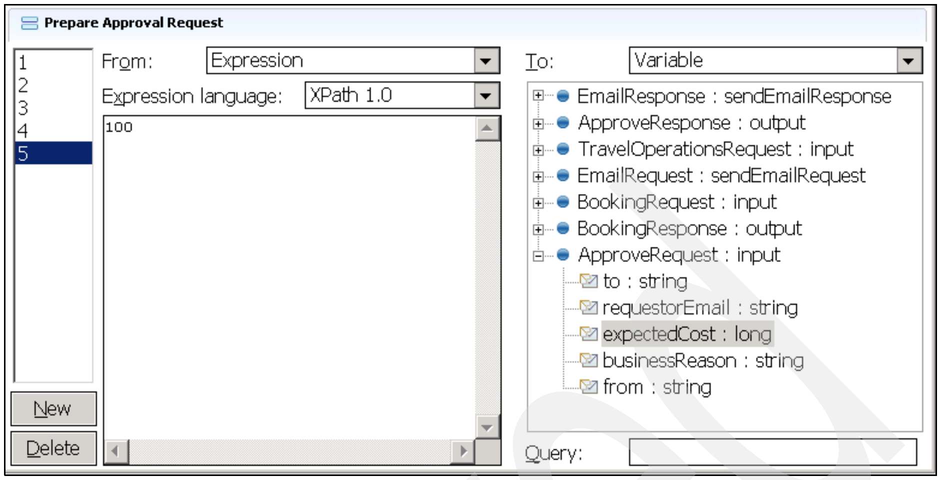
Cause

Assigns of fixed values are migrated to fixed values within XPath expressions by the Integration Developer Migration Wizard.

There is a known problem regarding fixed values within XPath expressions that are within the Process Server. Fixed values that are represented by XPath expressions in Assign activities are not cast properly in Process Server, which causes this exception.

Solution

This issue is going to be fixed in the next release of WebSphere Process Server. As a workaround, you can change the assigns that are in questions from being represented by fixed value within XPath expressions to being represented directly by fixed value expressions. See Figure 10-4 on page 281.


Prepare Approval Request

1
2
3
4
5

From: Expression
Expression language: XPath 1.0

```
100
```

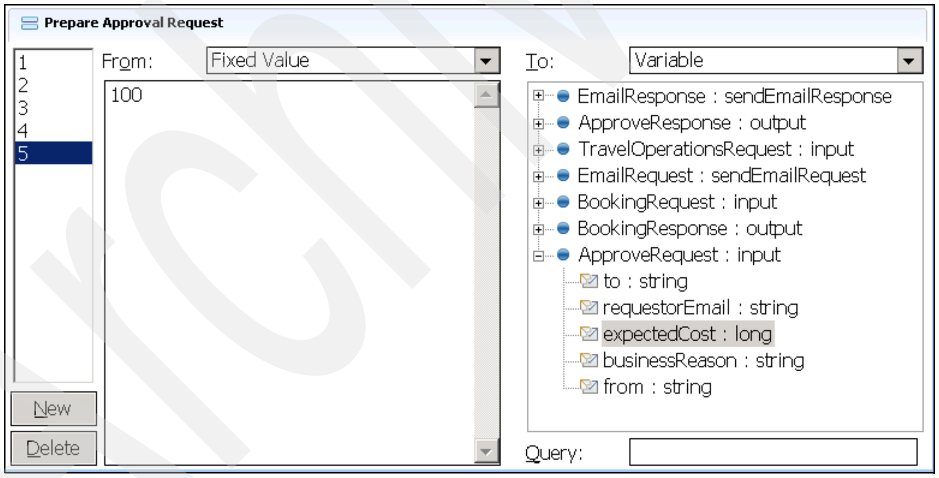
To: Variable

- EmailResponse : sendEmailResponse
- ApproveResponse : output
- TravelOperationsRequest : input
- EmailRequest : sendEmailRequest
- BookingRequest : input
- BookingResponse : output
- ApproveRequest : input
  - to : string
  - requestorEmail : string
  - expectedCost : long
  - businessReason : string
  - from : string

Query:

Figure 10-4 Migrated Prepare Approval Request Assign activity: fixed value within XPath

Clean up the questionable Assign activities by changing the style to Fixed Value, as shown Figure 10-5, for assign item 5 in the Prepare Approval Request activity.


Prepare Approval Request

1
2
3
4
5

From: Fixed Value

```
100
```

To: Variable

- EmailResponse : sendEmailResponse
- ApproveResponse : output
- TravelOperationsRequest : input
- EmailRequest : sendEmailRequest
- BookingRequest : input
- BookingResponse : output
- ApproveRequest : input
  - to : string
  - requestorEmail : string
  - expectedCost : long
  - businessReason : string
  - from : string

Query:

Figure 10-5 Migrated Prepare Approval Request Assign activity with fixed value

Repeat the same step for assign item 5 in the Prepare Booking activity.

10.3 Advanced

For unpredictable problems, follow these steps to resolve 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.

10.3.1 Search in information center or manuals

Sometimes the explanation of the error message can help you to understand the extra meanings of the error 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 Process Server V6 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp

WebSphere Integration Developer v6 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp

WebSphere Business Integration Server Foundation V5.1 information center:

http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp

WebSphere Business Modeler V6 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp

10.3.2 Search in product support Web site

IBM has a support Web site for all products. On the support Web site, fix packs, technotes, Authorized Program Analysis Reports (APARs) and some other useful resources are available.

When you encounter problems:

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 yes, try the solution provided in the APAR or the technote.

The following links are to the product's information center or manuals:

WebSphere Process Server V6 support Web site:

http://www-306.ibm.com/software/integration/wps/support/

WebSphere Integration Developer v6 support Web site:

http://www-306.ibm.com/software/integration/wid/support/

WebSphere Business Integration Server Foundation support Web site:

http://www-306.ibm.com/software/integration/wbisf/support/

WebSphere Studio Application Developer Integration Edition support Web site:

http://www-306.ibm.com/software/integration/wsadie/support/

WebSphere Business Modeler V6 support Web site:

http://www-306.ibm.com/software/integration/wbimodeler/support/

10.3.3 Contact IBM support for help

If you still cannot resolve the problem, contact IBM support for help. Access 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 for you to prepare and consider before you contact IBM support:

1. Make a clear problem description, including:
   – What software versions were running when the problem occurred?
   – Were there any logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
   – Can you recreate the problem? If so, what steps led to the failure?
   – Have any changes been made to the system? (for example, hardware, operating system, networking software, and so on)
   – Are you using any workaround for this problem? If so, be prepared to explain it while you report the problem.

2. Collect necessary document for the problem analysis:
   - For the development toolkit, such as WebSphere Integration Developer, WebSphere Business Modeler, and WebSphere Studio Application Developer Integration Edition V5.1, 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. Record the operations you did to reproduce the problem, for example, some window capture software is also a good choice.
   - For some runtime products, such as WebSphere Process Server and WebSphere Application Server, there are error logs and trace to record the problem. Also, the output of some special commands can help to analyze the problem. On the support Web sites of these products, MUSTGATHERs are often provided. MUSTGATHERs are some technotes that provide a detailed document list that is required for analyzing different problems. The following links are the MUSTGATHERs for WebSphere Process Server and WebSphere Application Server.

     WebSphere Process Server MUSTGATHER:

     `http://www-1.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&dc=DB520&dc=D600&dc=DB530&dc=D700&dc=DB500&dc=DB540&dc=DB510&dc=DB550&q1=Mustgather&uid=swg21239895&loc=en_US&cs=utf-8&lang=en`

     WebSphere Application Server MUSTGATHER:

     `http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599`

A

Additional material

In this book, we refer to additional material that you can download from the Internet.

Locating the Web material

The Web material that is associated with this IBM Redbook publication is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG2416

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, SG247216.

Using the Web material

The additional Web material that accompanies this book includes the files in the following table.

File name	Description
chapter9.zip	All zipped code samples that we used in the Chapter 9 scenarios.
travel booking.zip	Zipped code samples.
Travel Operations.zip	Zipped code samples.
WSADIEV511_PI_StartingPointForTechnicalScenario1.zip	

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Glossary

administration console The IBM administrative console is the Web interface to the Application Server, Process Server, or Enterprise Service Bus. It offers powerful options for configuration and deployment.

API Application Programming Interface is 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.

binding Defines the kind of interaction, the caller, and sender. You can use it to connect different SCA modules together.

BO Business Objects are representations of data structure that are used in business processes.

BPC Business Process Choreographer is component of WebSphere Process Server runtime architecture and provides support for business processes and human tasks. It offers a way to model your business process based on the WS-BPEL specification and to model interactions that involve humans, such as human-to-human, human-to-machine, and machine-to-human interactions. Both business processes and human tasks are exposed as services in a service-oriented architecture.

BPEL The Business Process Execution Language is an open standard to store process definitions in a file.

BPEL4WS Business Process Execution Language for Web services.

Breakpoint Used as a point to stop the flow of a message in a message flow when the flow debugger is attached.

Broker A broker is a set of execution processes that host and run message flows.

bus A bus provides the possibility to transfer data between different nodes that are connected to this bus.

Business Process A business Process defines a flow of actions to solve a special task. It can be stored in a BPEL file.

CEI Common Event Infrastructure is part of the SOA core. You can use CEI to capture events 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 SCA service component.

J2EE Java Platform, Enterprise Edition or Java EE (formerly known as Java 2 Platform, Enterprise Edition or J2EE up to version 1.5), is a programming platform—part of the Java Platform—for developing and running distributed multitier architecture Java applications, based largely on modular software components that are running on an application server. The Java EE platform is defined by a specification. Similar to other Java Community Process specifications, Java EE is also considered informally to be a standard because providers must agree to certain conformance requirements to declare their products as Java EE compliant; albeit with no ISO or ECMA standard.
Java EE includes several API specifications, such as JDBC, RMI, e-mail, JMS, Web services, XML, and so on, and defines how to coordinate them. Java EE also features some specifications that are unique to Java EE for components, which include Enterprise Java Beans, servlets, portlets (following the Java Portlet specification), JavaServer™ Pages™, and several Web service technologies. This allows the developer to create an enterprise application that is portable between platforms and scalable, while integrating with older technologies. Other added bonuses are, for example, that the application server can handle the transactions, security, scalability, concurrency and management of the components that are deployed to it, meaning that the developers can concentrate more on the business logic of the components rather than the lower level maintenance tasks.

Java Snippet A small part of a program code that is written in JAVA.

JCA The J2EE Connector Architecture is a Java-based technology solution for connecting application servers and enterprise information systems.

JMS Java Message Service, which is part of the J2EE (Java 2 Enterprise Edition) suite, provides standard APIs that Java developers can use to access the common features of enterprise message systems. JMS supports the publish/subscribe and point-to-point models and allows the creation of message types that consist of arbitrary Java objects.

JNDI Java Naming and Directory Interface™ is 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 using Java's object model. Using JNDI, applications that are based on Java technology can store and retrieve named Java objects of any type. In addition, JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.

LDAP Lightweight Directory Access Protocol (LDAP), is a networking protocol for querying and modifying directory services that run over TCP/IP. A directory is 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.
An 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 might appear entries that represent people, organizational units, printers, documents, groups of people, or anything else that represents a given tree entry (or multiple entries).

LDIF LDAP Data Interchange Format is a standard data interchange format for representing (LDAP) directory content and directory update (Add, Modify, Delete, Rename) requests. It 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.

MDB message driven bean is an Enterprise JavaBean (EJB) that is similar to a session bean, except it responds to a JMS message rather than an RMI event. MDBs were introduced in the EJB 2.0 specification, which is supported by Java 2 Platform, Enterprise Edition 1.3 or higher. The message bean represents the integration of JMS (Java Message Service) with EJB to create an entirely new type of bean that is designed to handle asynchronous JMS messages.

Queue manager A queue manager is a system program that provides queuing services to applications. It enables communication between the WebSphere Message Broker components, and each component requires access to a Queue Manager.

SCA The Service Component Architecture is the base for a service-oriented architecture because the different services can be easily connected to solutions.

SCA component A part of an SCA module that has a specific functionality. They can be assembled to a meaningful service.

SCA Export A possibility to provide an interface that can be called by other SCA modules using an import.

SCA Import A possibility to call an interface of another SCA module that is provided using an export.

SCA Module A logical unit of different SCA components that build a service.

SDO The Service Data Object is a data representation mostly used to connect to a enterprise information system. It caches the data so that the consumer do not have to care about interacting with the EIS directly.

SOA Service-oriented architecture is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks or services. SOA helps you build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes.

SOAP SOAP (originally Simple Object Access Protocol) is a protocol for exchanging XML-based messages over computer networks, normally using HTTP. SOAP forms the foundation layer of the Web services stack and provides a basic messaging framework that more abstract layers can build on. The original acronym was dropped with Version 1.2 of the standard, which became a W3C Recommendation on June 24, 2003 because it was considered to be misleading.

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.

WebSphere Application Server A very powerful application server that IBM offers. It is also included as a powerful foundation in other products, such as WebSphere Portal and WebSphere Process Server.

WebSphere MQ A messaging application that 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 MQ Explorer A graphical user interface for WebSphere MQ for administering WebSphere MQ components, such as queue managers, channels, and queue.

WebSphere Process Server A state of the art Business Process execution environment.

Wire A connection between SCA components in a SCA module. The wire always connects Reference to interface.

WSDL Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services.
WSDL is an XML-based service description on how to communicate using Web services. The WSDL defines services as collections of network endpoints, or ports. WSDL specification provides an XML format for documents for this purpose.

XSD An XML Schema, published as a W3C Recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve recommendation status by the W3C.
Like all XML schema languages, you can use XML Schema to express a schema, which is a set of rules to which an XML document must conform to be considered 'valid' according to that schema. However, unlike most other schema languages, XML Schema was also designed with the intent of validation that results in a collection of information that adheres to specific datatypes, which can be useful in the development of XML document processing software.
An XML Schema instance is an XML Schema Definition (XSD) and typically has the filename extension ".xsd". The language itself is sometimes informally referenced as XSD.

XSL The eXtensible Stylesheet Language (XSL) is a family of transformation languages that allows you to describe how files that are encoded in the XML standard are to be formatted or transformed. There are three languages in the family:
 * XSL Transformations (XSLT): an XML language for transforming XML documents
 * XSL Formatting Objects (XSL-FO): an XML language for specifying the visual formatting of an XML document
 * the XML Path Language (XPath): a non-XML language used by XSLT and also available for use in non-XSLT contexts for addressing the parts of an XML document.

Abbreviations and acronyms

ARM	application response monitor
BFM	Business Flow Manager
BPC	Business Process Choreographer
BPE	Business Process Engine
BPEDB	Business Process Choreographer database
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
CBE	Common Base Events
CEI	Common Event Infrastructure
CPU	Central Processing Unit
EAR	Enterprise Application
EIS	Enterprise Information Systems
EJB	Enterprise JavaBeans
ESB	Enterprise Service Bus
GUI	Graphical User Interface
HHTP	Hypertext Transfer Protocol
HTM	Human Task Manager
HTML	Hypertext Markup Language
IBM	International Business Machines Corporation
IE	Integration Edition (WebSphere Studio)
IT	Information Technology
ITSO	International Technical Support Organization
J2EE	Java 2 Enterprise Edition
JAR	Java Archive
JCA	J2EE Connector Architecture
JDBC	Java Database Connection
JMS	Java Message Service
JSP	JavaServer Pages
LDAP	Lightweight Directory Access Protocol
MDB	message driven bean
ODA	Object Discovery Agent
ORB	Object Request Broker
PMEs	Programming Model Extensions
PMI	performance monitoring infrastructure
SCA	Service Component Architecture
SCA	Service component architecture
SDO	Service Data Object
SDOs	Service Data Objects
SDOs	stored as Service Data Objects
SLA	Service Level Agreement
SOA	Service-oriented architecture
SQL	Structured Query Language
TEL	Task Execution Language
UDDI	Universal Description, Discovery, and Integration
UTE	Unit Test Environment
WAR	Web Archive
WID	WebSphere Integration Developer
WSDL	Web Services Description Language
WS-Security	Web Services Security
XML	Extended Markup Language

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 296. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *WebSphere Business Integration V6 Performance Tuning*, REDP-4195

Online resources

These Web sites are also relevant as further information sources:

- Installation and system requirements documentation on WebSphere Integration Developer V6.0.2 and WebSphere Process Server V6.0.2:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp
- Installation and system requirements documentation and further details about WebSphere Business Integration Server Foundation V5.1:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp
- Migrating applications using Programming Model Extensions:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rins_migratepme.html
- Basic steps for clustering WebSphere Process Server:

  http://www-128.ibm.com/developerworks/websphere/techjournal/0604_chilanti/0604_chilanti.html
- WebSphere Application Server Network Deployment V6: High Availability Solutions:

  http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=SG246688

- WebSphere Process Server information center for more comprehensive information about creating a secure WebSphere Process Server environment:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp

- The developer works article provides information about requirements for security in WebSphere Process Server and how to leverage security features of WebSphere Application Server V6:

  http://www-128.ibm.com/developerworks/websphere/library/techarticles/0602_khangaonkar/0602_khangaonkar.html

- WebSphere Business Integration V6 Performance Tuning:

  http://www.redbooks.ibm.com/cgi-bin/searchsite.cgi?query=REDP-4195

- A Server Foundation and Process Server installation can run on the same machine. In that event, ensure that they are not running in port conflicts. The documentation is available at:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/iins_coexist_top.html

- Information about Eclipse and tutorials:

  http://www.eclipse.org

- Information about using the AuditLog:

  http://www-128.ibm.com/developerworks/websphere/library/techarticles/0503_neumann/0503_neumann.html

- Information about integrating EJBs with WebSphere Process Server:

  http://www.ibm.com/developerworks/websphere/library/techarticles/0602_xu2/0602_xu2.html

- Migrating Business Rule Beans applications:

  http://www-1.ibm.com/support/docview.wss?uid=swg21217937

- WebSphere Integration Developer Information center:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tejbproj.html

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/rlimsrcart.html

- Description of the event contents for Server Foundation:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.websphere.wbifz.doc/info/wbifz/workflow/tasks/tmonitor.html

- Description of the event contents for WebSphere Process Server:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.mon.doc/doc/bpc/cmonitor_process.html

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.migration.ui.doc/topics/tsrcartifacts.html

- A complete list of system requirements for using Process Server:

  http://www-1.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&uid=swg27006205

- Information about WebSphere Process Server database:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.602.ins.doc/doc/cins_db_specs.html

- Information about Business Process Choreographer:

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

- WebSphere MQ as a JMS provider and WebSphere MQ Cluster Information center:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/bpc/t2creq.html

- Information about creating the BPEDB in WebSphere Process Server:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/bpc/t2data.html

- Basic aspects of using the staff service in the WebSphere Process Server:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.ins.doc/doc/bpc/c4staff.html

- Microflows and long running processes:

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

- Repair compensation actions in error in the Business Process Choreographer:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.wasee.doc/info/ee/wfclient/tasks/t7comp.html

- Failed messages:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.wasee.doc/info/ee/workflow/concepts/c7replay.html

- WebSphere Process Server V6 support Web site:

  http://www-306.ibm.com/software/integration/wps/support/

- WebSphere Integration Developer v6 support Web site:

  http://www-306.ibm.com/software/integration/wid/support/
- WebSphere Business Integration Server Foundation support Web site:

  http://www-306.ibm.com/software/integration/wbisf/support/
- WebSphere Studio Application Developer Integration Edition support Web site:

  http://www-306.ibm.com/software/integration/wsadie/support/
- WebSphere Business Modeler V6 support Web site:

  http://www-306.ibm.com/software/integration/wbimodeler/support/
- WebSphere Process Server MUSTGATHER:

  http://www-1.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&dc=DB520&dc=D600&dc=DB530&dc=D700&dc=DB500&dc=DB540&dc=DB510&dc=DB550&q1=Mustgather&uid=swg21239895&loc=en_US&cs=utf-8&lang=en
- WebSphere Application Server MUSTGATHER:

  http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A
APIs
  See also Application programming interfaces
Application
  Response monitor 24

B
BPEL 5, 10–12, 15, 21–24, 30, 35, 50, 56–58, 64, 71, 85, 87, 89, 93, 101, 105–106, 110, 112–113, 132, 136, 139, 151–154, 157–158, 160, 163, 170
  processes 49, 57, 91–92, 97, 106
  See also Business Process Execution Language
BPEL4WS 11, 13, 30, 71, 93, 101, 106
Bsiness Process Execution Explorer
  See also BPC Explorer 250
Business object 15–21, 30, 35, 60, 62–63, 65–66, 118, 134–135, 140–146, 182
Business objects 15–21, 30, 35, 60, 62–63, 65–66, 118, 134–135, 140–146, 182
Business Process
  Business Processes 10–13, 15–16, 21–24, 30, 35, 46–48, 56–58, 62, 84, 93, 95, 107, 110–114, 117, 124–125, 127–128, 130–133, 139, 147–148, 150–160, 163, 165, 167, 174–175, 179–180, 183, 198, 228–229, 240, 242, 250, 258
Business process 5, 10–12, 15, 21–24, 30, 35, 50, 56–58, 64, 71, 85, 87, 89, 93, 101, 105–106, 110, 112–113, 132, 136, 139, 151–154, 157–158, 160, 163, 170
Business Process Choreographer 10–13, 46–48, 56, 107, 110, 113, 125, 127–128, 130, 132–133, 139, 147–148, 150–158, 163, 165, 167, 175, 198, 228, 240
Business Process Engine
  See also BPE
Business Process Execution Language
  See also BPEL
Business processes 5, 10–12, 15, 21–24, 30, 35, 50, 56–58, 64, 71, 85, 87, 89, 93, 101, 105–106, 110, 112–113, 132, 136, 139, 151–154, 157–158, 160, 163, 170
Business Rule Beans 10, 30–31, 91–93, 137, 147–148, 165
Business Rules 13–15, 21, 23–24, 30, 35, 93, 114, 138, 151–152, 165, 180
Business Rules Service 14
Business state machine 23–24, 180

C
CEI 15, 35, 91–92, 139, 148, 151, 166–168
  See also Common Event Infrastructure
Clients 20, 29, 88, 91–92, 111–113, 125–128, 131, 134, 150, 174, 199, 221
CMM
  Extended Messaging 31–32, 92, 138, 148, 166
Coexistence 37, 44, 51, 79, 82, 85–89, 134–135
Common Event Infrastructure 10, 13, 15–17, 19, 24, 29, 35, 92, 139, 147–148, 152, 166
  See also CEI
Components 46

D
Data object 15, 18, 135–136, 223–224
DB2
  See also DB2 server

E
EAR
  files 243
Eclipse and tutorials 60
Editor
  Assembly Diagram editor 62, 226, 238
  business object 62–63
  business object mapping 65–66
  Relationship 66
  relationship 66
  visual snippet 67
EJB
  See also Enterprise JavaBeans
Enterprise Information Systems
  Enterprise Information Systems 15, 21, 24, 68, 140
Enterprise Service Bus 25, 30, 35, 152
  See also ESB
Enterprise service discovery wizard 60, 68–69, 143

ESB 35, 201–202
See also Enterprise Service Bus
extended data element 139

F
Functional Components 16

H
Hhuman task
editor
236
HTTP
See also Hypertext Transfer Protocol
Human task
activities 58, 71, 110, 129, 247
Human tasks 15, 21–22, 24, 88–89, 110, 125, 129, 132, 151–152, 157–158, 161, 163

I
installation
wizard 160–161
Interface
mapping editor 65
interface
editor 64–65, 105
Interface maps
Interface maps 20, 30, 35, 65

J
J2EE
Connector Architecture 31, 140
See also Java 2 Platform Enterprise Edition
Java Build path 191–192, 196, 209–210
Java component 35, 114, 117–118, 120–122, 133, 220, 232
Java Message Service 47, 141
Java snippets 94, 103, 105–107, 109, 133, 136, 174, 215, 230
JSP 6, 32, 48, 51, 58, 92–94, 104, 110, 112–113, 117, 123, 126–127, 138–140, 152, 157–158, 160, 164–165, 282

M
matching reference 120
Mediation
services 17
Mediation flows 15, 19–20
Mediations
pre-built 25
messaging
applications 91, 138, 166
Migration
approaches 79, 134
FDML processes 57
plan 2, 4
planning 1, 37, 83, 86
with dependencies 79
wizard 15, 33–34, 56–57, 59, 70–71, 91–93, 100–102, 104–106, 110, 123, 136, 171, 173, 205–209, 213–215, 217–219, 222–225, 229–230, 232, 234, 274, 278, 280
Migration Workspace 74
Model Extensions 10, 13, 31–32
Module
Name 101–103, 215, 230

P
Performance monitoring infrastructure 24
Process
Editor 64, 221–223, 234, 236
Process Choreographer Database
See also BPEDB 12, 82–83, 152, 157
Processes
long running 44, 49–51, 79, 82–83, 113, 164, 183, 234
migrate FDML processes 57
Products overview 1
Programming 10, 13, 31–32
Programming Model Extensions 10, 13, 31–32
Project Name 192

Q
Quality of Service 29, 170
See also QoS

R
Rational
Application Developer 100, 122, 205
Redbooks Web site 296
Contact us xxiv
Relationships 19, 21, 30, 66, 151–152, 182
Response
message 142
routing

Rule
  group 176
runtime environment 11, 40, 148, 154, 164, 183, 276

S
SCA 14–16, 19, 21, 23, 29–30, 33–35, 49, 58, 62, 64, 67, 71, 95, 101, 105–106, 110–114, 117–118, 120–126, 135–137, 141–143, 146, 150–151, 157, 174, 176, 180, 183–184, 217, 219–220, 223, 225–226, 232, 237, 279
  components 15, 19, 21, 23, 62, 64, 117, 150, 176, 180, 219
  proposes 95
SDO 15, 18–19, 29–30, 35, 89, 106, 114, 118, 134–135, 137, 140, 145, 217, 219–220, 223–224, 232
See also EIS
Selector 21, 176
Server Foundation 1–2, 4–6, 10–11, 13–14, 23, 31, 53–54, 79, 101, 106–107, 110, 125, 128, 133, 136, 153, 164, 170
  Overview 10
Server Foundation artifacts 10, 20, 29–31, 88, 91–93, 111–113, 125–128, 131, 134, 137, 147–148, 150, 165, 173–174, 199, 221
  Adapter usage 92
  BPEL processes 49, 57, 91–92, 97, 106
  Common Event Infrastructure 10, 13, 15–17, 19, 24, 29, 35, 92, 139, 147–148, 152, 166
  Container Managed Persistence 31–32, 92, 138
  EJB services 91–92, 117, 174
  Extended Messaging 31, 91–92, 138, 147–148, 166
  Java services 91–92, 106, 114, 174, 189–190, 193
  Web services 125, 173
  WSIFMessage API 91–92, 135, 137
  WSIFMessage Interface 133
Service Component Architecture 15–16, 21, 23, 30, 62, 135, 141
  See also SCA
Service Data Object 15, 18, 135–136, 223
  See also SDO
Service-oriented architecture 10, 13, 15, 17, 23, 29
  core layer consists 16
  See also SOA
SIMPLE 20, 107, 114, 118, 137, 179–180, 186, 188, 246
SIP method
  INFO 32, 48, 139, 164–165, 261
  OPTIONS 28, 55, 59, 67, 72, 82, 98, 101, 112–114, 116–117, 125–126, 144, 157, 207–208
  REFER 6, 36, 47–48, 51, 56–57, 59, 106, 126, 149, 153, 193, 202, 209, 214, 218–219, 235, 276
  SUBSCRIBE 150, 167
  UPDATE 103, 120, 135, 181, 215
SOA 10, 16, 23, 35
  See also Service Oriented Architecture

T
topology 42, 44, 47, 54, 147, 149, 152, 170, 186–187

U
UPDATE 103, 120, 135, 181, 215

W
WAS
  See also WebSphere Application Server
Web Client
  Launch Business Process 198, 258
Web Service
  Invocation 135–136
  Invocation Framework 135–136
Web services 125, 173
WebSphere
  Enterprise Service Bus 30
WebSphere Application Server
  Network Deployment 16
WebSphere Business Integration
  Modeler 53
WebSphere Business Integration Server Foundation 1–2, 4–6, 10–11, 13–14, 23, 31–32, 53–54, 79, 101, 106–107, 110, 125, 128, 133, 136, 153, 164, 170
  capabilities 5
  functionality 5
  Java services 91–92, 106, 114, 174, 189–190, 193
  Overview 10
WebSphere Integration Developer 24, 53, 60, 62, 68–69, 93, 97–98, 100–103, 105–106, 114, 116,

118, 120, 137–141, 143, 174, 180–181, 193, 204, 211, 214–215, 274, 276
Business Integration Library project 96, 98, 194, 211
Business Integration Module Project 93, 96, 102
Business Integration perspective 61, 96, 105, 212, 275
integration test client
60, 69–70
See also WID
WebSphere Modeler 15, 23
WebSphere Monitor 15, 23
WebSphere Process Server
advanced business process integration server 15
Business Rules Service 14
Business state machine 23–24, 180
concepts and architecture 10, 15
configuration parameters 49
WID 283
See also WebSphere Integration Developer
Windows
XP 187–188, 277
XP Professional 187–188, 277
WS-BPEL 30, 71, 93, 101, 106
WSDL
files 97, 101, 105, 123, 173, 191, 193–195, 211–212
See also Web Services Description Language

IBM

Redbooks

Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server and Best Practices

(0.5" spine)
0.475"<->0.875"
250 <-> 459 pages

Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server and Best Practices

Migration concepts and planning

Migration tools and best practices

Migration technical scenarios

In this IBM® Redbooks publication, we discuss the concepts, differences, and migration paths that you must understand before you attempt to migrate the artifacts that you created using the IBM WebSphere® Studio Application Developer Integration Edition 5.1 product to the IBM WebSphere Integration Developer 6.0.2. We also include a discussion on how to migrate models that are developed in WebSphere Business Integration Modeler 5.1 to WebSphere Business Modeler 6.0.2.

In this book, we provide guidance on how to migrate the processes that are installed and running in WebSphere Business Integration Server Foundation to the new integration platform. We also tell you how to bring your components and artifacts to this new generation of integration paradigm.

SG24-7416-00 ISBN 0738485632