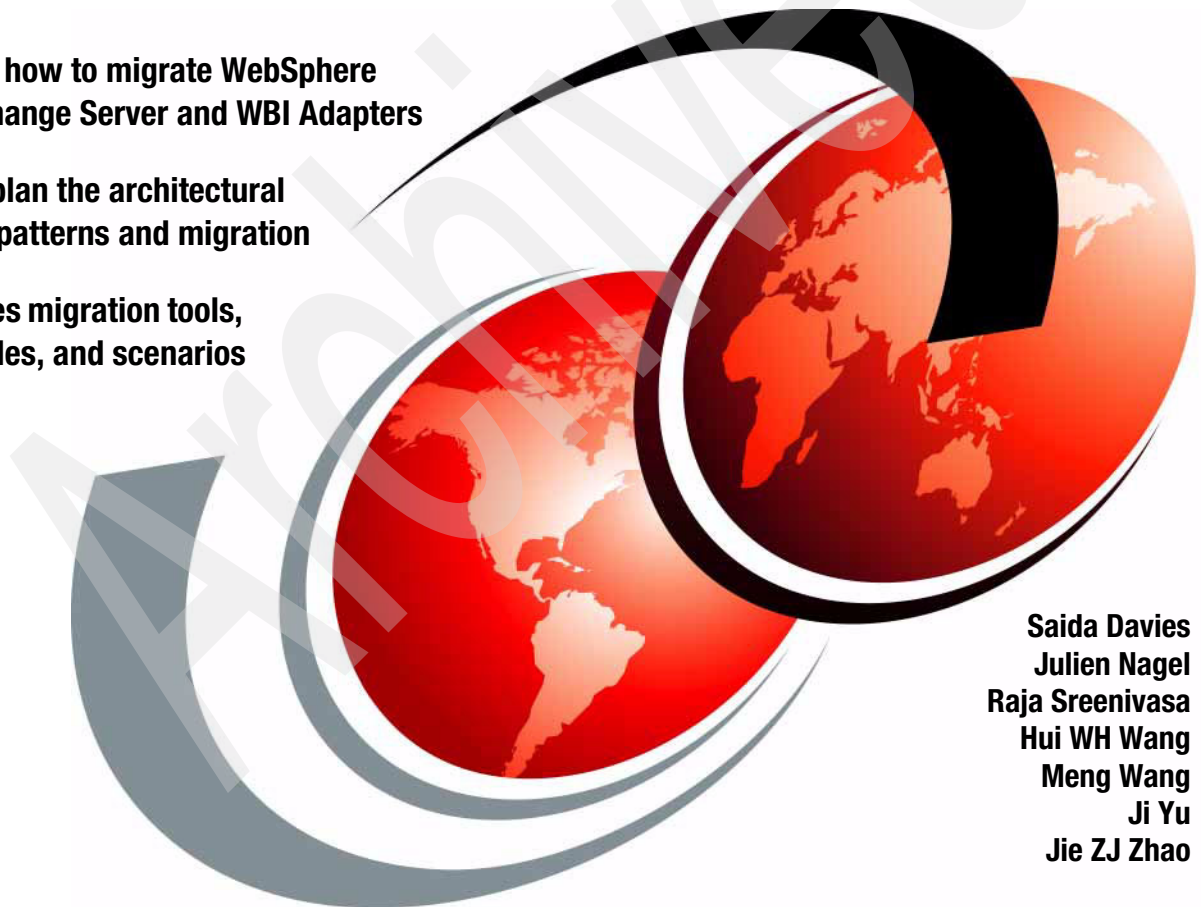


Migrating WebSphere InterChange Server and Adapters to WebSphere Process Server V6.2 and Best Practices

Shows how to migrate WebSphere InterChange Server and WBI Adapters

Helps plan the architectural usage patterns and migration

Includes migration tools, examples, and scenarios



Saida Davies
Julien Nagel
Raja Sreenivasa
Hui WH Wang
Meng Wang
Ji Yu
Jie ZJ Zhao



International Technical Support Organization

**Migrating WebSphere InterChange Server and
Adapters to WebSphere Process Server V6.2**

August 2009

Archived

Second Edition (August 2009)

Version	Release	Modification	Fix pack	Product name
XP			Service Pack 2	Microsoft Windows
5	3	0		IBM WebSphere MQ
5	3	0	CSD 13	IBM WebSphere MQ
6	0	2	2	IBM WebSphere MQ
6	0	2	4	IBM WebSphere MQ Client
6	0	2	4	IBM WebSphere MQ
4	0	9		RFHUtil
8	2	9		IBM DB2 Universal Database
4	3	0	5	IBM WebSphere InterChange Server and Toolkit
2	6	0	11	IBM WebSphere Business Integration Adapter Framework
2	6	0	12	IBM WebSphere Business Integration Adapter Framework
2	7	3		IBM WebSphere Business Integration Data Handler for XML
1	2	0		IBM WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture
5	6	5		IBM WebSphere Business Integration Adapter for JText
2	6	9		IBM WebSphere Business Integration Adapter for JDBC
2	8	2		IBM WebSphere Business Integration Adapter for MQ
6	0	8		IBM WebSphere Business Integration Adapter for mySAP.com
3	0	1		IBM WebSphere Business Integration Adapter for PeopleSoft
3	4	7		IBM WebSphere Business Integration Adapter for Web services
2	7	0		IBM WebSphere Business Integration Adapter for XML Data Handler
6	2			IBM WebSphere Process Server
6	2			IBM WebSphere Integration Developer
6	2			IBM WebSphere JDBC Adapter
6	2			IBM WebSphere Adapter for Flat Files
6	2			IBM WebSphere Adapter for SAP Software
6	2			IBM WebSphere Adapter for PeopleSoft
640			14	SAP Front-End GUI Tool

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	xiii
Tables	xxv
Examples	xxvii
Notices	xxix
Trademarks	xxx
Preface	xxxi
The team who wrote this book	xxxii
Become a published author	xxxvii
Comments welcome	xxxvii
Summary of changes	xxxix
August 2009, Second Edition	xxxix
Part 1. Migration concepts	1
Chapter 1. Introduction to this book	3
1.1 Executive summary	4
1.2 Scope of the book	5
1.3 Intended audience	5
1.4 What is covered	6
1.5 What is not covered	7
1.6 Assumptions	7
Chapter 2. Integration background	9
2.1 Service Integration Maturity Model	10
2.2 SIMM level 1: Silo	13
2.3 SIMM level 2: Integrated (point-to-point)	14
2.4 SIMM level 3: Componentized (hub and spoke)	15
2.5 SIMM level 4: Services	16
2.6 SIMM level 5: Composite services	17
2.7 SIMM level 6: Virtualized services	19
2.8 SIMM level 7: Dynamically reconfigurable services	20
2.9 Summary	21
Chapter 3. Usage patterns	23
3.1 Data synchronization pattern	25

3.2	Data synchronization with dependencies pattern	26
3.3	Data access pattern	27
3.4	Services pattern (accompanying a framework)	29
3.5	Business process or workflow pattern	31
Chapter 4.	Migration planning	33
4.1	Considerations for planning the migration	34
4.1.1	Migration path evaluation	34
4.1.2	Internal IT factors	35
4.1.3	Recommendations for defining the migration path	37
4.2	Assessment phase	39
4.2.1	Current environment	39
4.2.2	Requirements gathering	40
4.2.3	Impact analysis	40
4.2.4	Deliverables	62
4.3	Preparation phase	64
4.3.1	Building skills	64
4.3.2	Using the WebSphere Process Server project experience	64
4.3.3	Refining the assessment	66
4.4	Implementation phase	67
4.4.1	Development	68
4.4.2	Test	68
4.4.3	Deployment	68
4.4.4	Go live	69
Part 2.	Migration implementation concepts	71
Chapter 5.	Product overview	73
5.1	WebSphere InterChange Server	74
5.1.1	Introduction to WebSphere InterChange Server	74
5.1.2	Architectural overview of WebSphere InterChange Server	74
5.1.3	Features of WebSphere InterChange Server	78
5.2	WebSphere Business Integration Adapters	82
5.2.1	WebSphere Business Integration Adapter Framework	82
5.2.2	WebSphere Business Integration Adapters	82
5.3	WebSphere Business Integration data handlers	87
5.3.1	XML data handler	87
5.3.2	Delimited data handler	88
5.3.3	FixedWidth data handler	88
5.3.4	Complex data handler	89
5.3.5	EDI data handler	89
5.3.6	Custom data handler	89
5.4	WebSphere Process Server	90
5.4.1	Introduction to WebSphere Process Server	90

5.4.2 Architectural overview of WebSphere Process Server	90
5.4.3 Features of WebSphere Process Server	97
5.5 WebSphere Adapters	99
5.5.1 WebSphere Adapter for SAP	100
5.5.2 WebSphere Adapter for Flat Files	101
5.6 WebSphere Process Server bindings	101
5.6.1 EIS binding	102
5.6.2 EJB binding	102
5.6.3 Web service binding	103
5.6.4 HTTP binding	103
5.6.5 JMS binding	103
5.6.6 Generic JMS binding	103
5.6.7 WebSphere MQ JMS binding	103
5.6.8 WebSphere MQ binding	104
5.7 WebSphere Process Server data bindings	104
5.7.1 EIS data bindings	105
5.7.2 JMS data bindings	106
5.7.3 WebSphere MQ data bindings	107
5.7.4 HTTP data bindings	107
5.7.5 Custom data bindings and data handlers	108
Chapter 6. Product feature comparison	109
6.1 Architecture and design aspects	110
6.1.1 Server architecture	110
6.1.2 Programing model	110
6.2 Development and testing	111
6.2.1 Source control	111
6.2.2 Testing	111
6.3 Deployment	112
6.3.1 Packaging and deployment model	112
6.3.2 Inter-environment deployment	112
6.3.3 Versioning	113
6.4 Run time	113
6.4.1 High availability and clustering	114
6.4.2 Transactionality	115
6.4.3 Compensation	116
6.4.4 Event Sequencing	117
6.4.5 Flow control	117
6.4.6 Security	118
6.5 Feature comparison summary	119
6.5.1 Functional gaps closed with WebSphere Process Server 6.2	119
6.5.2 WebSphere Process Server features superior to WebSphere InterChange Server features	120

6.5.3 WebSphere InterChange Server features still superior to WebSphere Process Server features	120
Chapter 7. Server component upgrade	123
7.1 Overview of the upgrade	124
7.1.1 Planning	124
7.1.2 Server migration constraints	124
7.1.3 Server artifact migration	125
7.2 Component mapping	125
7.3 Benefits	128
7.4 Limitations	129
Chapter 8. Adapters and data handlers upgrade	131
8.1 Overview of WebSphere Business Integration Adapters	132
8.2 Adapter types	134
8.3 Adapter mapping	134
8.4 Equivalence of adapter functions	136
8.5 Upgrade paths	137
8.5.1 Implementing a WebSphere Business Integration Adapter with WebSphere Process Server	138
8.5.2 Upgrading a WebSphere Business Integration Adapter to a WebSphere Adapter	140
8.5.3 Upgrading a WebSphere Business Integration Adapter with a binding	141
8.5.4 Replacing a WebSphere Business Integration Data Handler	145
8.5.5 Writing a custom adapter using the WebSphere Adapter Toolkit ..	147
8.6 Benefits	148
8.7 Limitations	149
Chapter 9. Migration implementation approach	151
9.1 Integration solutions perspective	152
9.1.1 Architectural design patterns and migration implementation	152
9.1.2 Component model and integration layers	153
9.1.3 Separation of concerns	155
9.2 Migration implementation challenges	160
9.2.1 Upgrade of WebSphere Business Integration Adapters	161
9.2.2 Functional gaps between products	166
9.2.3 Specific Java developments	168
9.3 Selection criteria for migration implementation	169
9.3.1 Technical approaches	169
9.3.2 Technical approaches and the standard migration tools	172
9.3.3 Custom code and standard migration tool constraints	173
9.3.4 Quality of automatically migrated code	173
9.3.5 Performance of automatically migrated code	174

Part 3. Migration tooling	177
Chapter 10. Overview of migration tools	179
10.1 Migration tools	180
10.1.1 Supported versions	180
10.2 WebSphere Integration Developer	181
10.2.1 Editors	183
10.2.2 Module	190
10.2.3 Shared library	190
10.2.4 Integration test client	191
10.3 Migration Wizard	192
10.3.1 Using the Migration Wizard	192
10.4 The reposMigrate utility	205
10.4.1 Migrating by using the reposMigrate utility	205
10.5 Migration tools specific to WebSphere Business Integration Adapters	208
10.5.1 WebSphere Business Integration Adapter Artifact Importer	208
10.5.2 Migrating using the Adapter Migration Wizard	211
Chapter 11. Artifacts migration	215
11.1 Migration prerequisites and constraints	216
11.2 Files generated during migration	216
11.3 Business objects	217
11.3.1 Business objects in WebSphere InterChange Server	217
11.3.2 Service Data Objects in WebSphere Process Server	218
11.3.3 Migration of business objects	220
11.4 Maps	222
11.4.1 Maps in WebSphere InterChange Server	222
11.4.2 Maps in WebSphere Process Server	223
11.4.3 Migration of maps	224
11.5 Collaborations	227
11.5.1 Collaborations in WebSphere InterChange Server	227
11.5.2 Business processes in WebSphere Process Server	228
11.5.3 Migration of collaborations	231
11.6 Relationships	237
11.6.1 Relationships in WebSphere InterChange Server	237
11.6.2 Relationships in WebSphere Process Server	237
11.6.3 Migration of relationship artifacts	239
11.7 WebSphere Business Integration Adapters	240
11.7.1 WebSphere Business Integration Adapters with WebSphere InterChange Server	240
11.7.2 WebSphere Business Integration Adapters in WebSphere Process Server	242
11.7.3 Migration of WebSphere Business Integration Adapter artifacts	244

11.7.4	WebSphere Adapters	245
11.7.5	WebSphere Process Server bindings	248
11.8	WebSphere InterChange Server Access Interface	248
11.8.1	Overview of the Server Access feature in WebSphere InterChange Server	248
11.8.2	Access EJB support in WebSphere Process Server	250
Chapter 12.	Post-migration tasks	253
12.1	Jython script	254
12.1.1	Running the script	254
12.2	Components covered by the Jython script	255
12.2.1	Database connection pools	256
12.2.2	Relationship reuse	256
12.2.3	Scheduler	257
12.3	Bindings	258
12.3.1	Migrating a WebSphere Business Integration Adapter for EJB to a WebSphere Process Server Stateless Session Bean	258
12.3.2	Migrating a WebSphere Business Integration Adapter for HTTP to a WebSphere Process Server HTTP binding	259
12.3.3	Migrating a WebSphere Business Integration Adapter for JMS to a WebSphere Process Server JMS binding or generic JMS binding	260
12.3.4	Migrating a WebSphere Business Integration Adapter for MQ to a WebSphere Process Server MQ binding or MQ JMS binding	261
12.3.5	Migrating a WebSphere Business Integration Adapter for Web services to a WebSphere Process Server HTTP binding	261
Chapter 13.	Best practices	263
13.1	Artifact preparation	264
13.1.1	General development	264
13.1.2	Common code utilities	267
13.1.3	Business objects	267
13.1.4	Database connection pools	268
13.1.5	Collaboration templates	268
13.1.6	Maps	270
13.1.7	Relationships	270
13.1.8	Access framework clients	271
13.1.9	Naming convention	271
13.1.10	Additional information	272
13.2	Post-migration review	272
13.3	Nonfunctional considerations	273
Part 4.	End-to-end technical solutions	277
Chapter 14.	Preparation for the technical solutions	279

14.1	Setting up the testing environment	281
14.1.1	Software environment	281
14.1.2	Setting up the database	281
14.2	Configuring WebSphere MQ	289
14.3	Setting up a file system for the JText Connector	290
Chapter 15.	Data access scenario with technology adapters	293
15.1	Target environment	295
15.1.1	Software environment	295
15.2	Implementation	295
15.2.1	Premigration overview	296
15.2.2	Development	297
15.2.3	Deployment	319
15.2.4	Setting up the testing module	331
15.3	Testing the end-to-end solution	335
15.4	Conclusion	341
Chapter 16.	Data synchronization scenario with technology adapters	343
16.1	Target environment	346
16.1.1	Software environment	346
16.2	Implementation	346
16.2.1	Premigration overview	346
16.2.2	Preparation	348
16.2.3	Migration	351
16.2.4	Work-around for dynamic relationship	388
16.2.5	Running a Jython script and modifying JDBC data sources	401
16.2.6	Deployment	407
16.3	Testing the end-to-end solution	419
16.4	Conclusion	429
Chapter 17.	Data synchronization scenario with application adapters	431
17.1	Target environment	433
17.1.1	Software environment	433
17.2	Implementation	434
17.2.1	Premigration overview	434
17.2.2	Preparation	434
17.2.3	Migration	438
17.2.4	Work-around for dynamic relationship	467
17.2.5	Running a Jython script and modifying JDBC data sources	478
17.2.6	Deployment	485
17.3	Testing the end-to-end solution	497
17.4	Conclusion	509
Chapter 18.	Customize and enhance the solutions	511

18.1 Enhanced error handling	512
18.1.1 Target environment	512
18.1.2 Error handling example	513
18.2 Enhanced routing capability	535
18.2.1 Routing capability in the WebSphere InterChange Server	535
18.2.2 Migrating the routing capability in WebSphere Process Server . .	539
18.2.3 Optimization of the routing capability in WebSphere Process Server .	545
18.3 Enhanced migration for artifacts designed visually	551
18.3.1 Map designed with the Activity Editor	551
18.3.2 Migration of the map	553
18.3.3 Enhanced design for the migrated map	557
Chapter 19. Technical solutions: Troubleshooting	559
19.1 General troubleshooting methods	560
19.1.1 Application module errors	560
19.1.2 Deploying and running the scenario in WebSphere Process Server .	560
19.1.3 WebSphere Process Server log and trace	561
19.2 Advanced troubleshooting methods	568
19.2.1 Adapter failed to start in a secured WebSphere Process Server .	568
19.2.2 Failed to run the data synchronization scenario after migrating to	
WebSphere Adapter for JDBC	569
19.2.3 Failed to run the RetrieveCustomer on the data access scenario after	
migration	570
Appendix A. WBI Adapters in a secured WebSphere Process Server	
environment.	571
A.1 Overview	572
A.2 Configuring security	573
A.2.1 Initial assumptions	573
A.2.2 Setting up the connector agent with secured JMS	574
A.2.3 Setting up a security identity qualifier in Message-Driven Beans . .	576
A.2.4 Allowing the MQ chain for the service integration bus	582
A.2.5 Deployment of the connector modules	584
A.3 Conclusion	590
Appendix B. Access EJB migration	591
B.1 Target environment	592
B.2 Access EJB example	593
B.2.1 Preparation	593
B.2.2 Implementation	600
B.2.3 Testing and verification	611
B.3 Conclusion	614

Appendix C. Migrate WBI Adapter for EJB Architecture	615
C.1 Target environment	617
C.2 Running the WebSphere Business Integration Adapter with WebSphere Process Server: EJB	617
C.2.1 Premigration overview	618
C.2.2 Operational model	618
C.2.3 Development	619
C.2.4 Modifying the adapter configuration files	627
C.2.5 Testing and verification	629
C.3 Migrating to an EJB binding	631
C.3.1 Premigration overview	632
C.3.2 Operational model	632
C.3.3 Development	633
C.3.4 Implement the Output skeleton Java component	640
C.3.5 Testing and verification	645
C.4 Conclusion	648
Appendix D. Additional material	651
Locating the Web material	651
Using the Web material	652
How to use the Web material	652
Glossary	653
Abbreviations and acronyms	657
Related publications	659
IBM Redbooks publications	659
Online resources	659
How to get IBM Redbooks publications	660
Help from IBM	661
Index	663

Figures

2-1	Service Integration Maturity Model	12
2-2	SIMM level 1: Siloed applications	13
2-3	SIMM level 2: Point-to-point integration	14
2-4	SIMM level 3: Hub and spoke integration	15
2-5	SIMM level 4: Services	16
2-6	SIMM level 5: Composite services	18
2-7	SIMM level 6: Virtualized services	19
2-8	SIMM level 7: Dynamically reconfigurable services	20
3-1	WebSphere InterChange Server data synchronization pattern	25
3-2	WebSphere InterChange Server data synchronization with dependencies pattern	26
3-3	WebSphere InterChange Server data access pattern	28
3-4	WebSphere InterChange Server services pattern	30
3-5	WebSphere InterChange Server workflow pattern	31
4-1	Components to be clustered in WebSphere Process Server	52
5-1	WebSphere InterChange Server components	75
5-2	WebSphere Business Integration Adapters	82
5-3	Components of WebSphere Process Server	91
5-4	Service-oriented architecture core layer of WebSphere Process Server	92
5-5	Service Component Architecture	93
5-6	Supporting services layer of WebSphere Process Server	94
5-7	Service component layer of WebSphere Process Server	96
5-8	WebSphere Adapters	99
5-9	WebSphere Process Server Binding examples	102
5-10	Diagram of WebSphere Process Server data binding	105
8-1	WebSphere Business Integration Adapters overview	132
8-2	Upgrade paths for WebSphere Business Integration Adapters	133
8-3	WebSphere Business Integration Adapter interface with WebSphere Process Server	139
8-4	WebSphere Adapters with WebSphere Process Server	140
8-5	WebSphere Process Server bindings	141
8-6	WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture used with WebSphere Process Server	142
8-7	EJB Import binding	142
8-8	WebSphere Business Integration Adapter for WebSphere MQ or for JMS used in conjunction with WebSphere Process Server	143
8-9	MQ or JMS binding after adapter replacement	143
8-10	WebSphere Business Integration Adapter for Web Services before	

migrating to WebSphere Process Server	144
8-11 WebSphere Business Integration Adapter for Web Services upgraded to Web services binding	145
8-12 WebSphere Process Server support to reuse WebSphere Business Integration data handlers	147
9-1 Integration layers and the components in WebSphere InterChange Server and WebSphere Process Server	155
9-2 Integration layers and business logic layer	156
9-3 Separation of concerns in WebSphere Process Server: Integration logic example	158
9-4 Separation of concerns in WebSphere Process Server: Business logic example	159
9-5 WebSphere Business Integration Adapter and WebSphere InterChange Server	162
9-6 WebSphere Adapter or WebSphere Process Server binding	162
9-7 WebSphere Business Integration Adapter reused with WebSphere Process Server	164
9-8 Reusing migrated maps	165
9-9 Approaches comparison	170
10-1 Welcome page	182
10-2 Workbench	183
10-3 Assembly Editor	184
10-4 Business Object Editor	184
10-5 Process Editor	185
10-6 Interface Editor	186
10-7 interface Mapping Editor	186
10-8 Mediation Flow Editor	187
10-9 Business Object Mapping Editor	188
10-10 Relationship Editor	189
10-11 Visual Snippet Editor	189
10-12 Dependencies Editor	190
10-13 Integration test client	191
10-14 Migration Wizard	193
10-15 Returning users icon	194
10-16 Returning users page	195
10-17 Migration page	196
10-18 Select WebSphere InterChange Repository Details	197
10-19 Configure Connector Migration	198
10-20 Conversion Options	202
10-21 Migration Summary	203
10-22 Migration Results	204
10-23 Artifact conversion	206
10-24 External Service	209

10-25	Select a Messaging Agent	210
10-26	Discovery Configuration	211
10-27	Adapter Migration Wizard	212
10-28	Review Changes	213
11-1	WebSphere Process Server components architecture	218
11-2	WebSphere Process Server business objects	219
11-3	Business objects that are listed under Data Types	220
11-4	Employee business object	221
11-5	Employee business graph	221
11-6	Map in WebSphere InterChange Server	225
11-7	Business object map in WebSphere Process Server	226
11-8	Business graph map in WebSphere Process Server	227
11-9	Sample collaboration template in WebSphere InterChange Server ..	228
11-10	Sample collaboration object in WebSphere InterChange Server ..	228
11-11	Sample business process in WebSphere Process Server	230
11-12	WebSphere InterChange Server collaboration with a decision node .	231
11-13	Decision node criteria	232
11-14	Business process with a condition on two of the links in WebSphere Process Server	233
11-15	Collaboration with a break node	234
11-16	Collaboration with asynchronous inbound service call	235
11-17	Receive choice activity in WebSphere Process Server	235
11-18	Example of a relationship	239
11-19	ContactR relationship	239
11-20	WebSphere Business Integration Adapter architecture	241
11-21	WebSphere Business Integration Adapter deployment in WebSphere InterChange Server	242
11-22	Using WebSphere Business Integration Adapter in WebSphere Process Server	243
11-23	Generated project after migration	244
11-24	JCA adapter architecture	246
11-25	WebSphere JCA Adapter process	247
11-26	Access EJB communication with WebSphere InterChange Server ..	250
12-1	Jython script	255
12-2	Enterprise JavaBean Binding	258
12-3	EJBRequestHandlerImpl.java	259
14-1	Downloading and extracting the files to the setup temp folder	283
14-2	Creating a new standard database in DB2 Control Center	284
14-3	Database Configuration Advisor DB2 Message window	284
14-4	Configuration Advisor confirmation DB2 Message window	285
14-5	ICSREPOS database information view	286
14-6	DB2 Command Editor	287
14-7	Successful creation of the JDBCCustomer database table	288

14-8	CUST.QUEUE.MANAGER in WebSphere MQ Explorer	290
14-9	JText Connector directory setup	291
15-1	Create Customer Collaboration flow diagram	296
15-2	Retrieve Customer Collaboration flow diagram	297
15-3	New data access scenario library	298
15-4	WebSphere InterChange Server JAR file Import wizard	299
15-5	Migration Import wizard	300
15-6	Configure Connector Migration	301
15-7	Conversion Options	302
15-8	Migration Summary	303
15-9	Migration Results	304
15-10	Module created	305
15-11	Select Projects	306
15-12	Review changes	307
15-13	Create Customer Process Component	308
15-14	JCA JDBC Adapter	309
15-15	MQ binding	310
15-16	HTTP binding	311
15-17	Starting WebSphere Process Server	320
15-18	Running the administrative console	321
15-19	Disabling administrative security	322
15-20	Disabling bus security	323
15-21	Allowing permitted transports for bus security	324
15-22	Opening the profile configuration	325
15-23	Disabling profile security	326
15-24	Restarting the WebSphere Process Server	327
15-25	Adding projects to WebSphere Process Server	328
15-26	Deploying to WebSphere Process Server	329
15-27	New Web services utility library	331
15-28	Importing the WSDL interface files	332
15-29	Importing the WSDL interface selection	334
15-30	Importing the WSDL generated artifacts	335
15-31	Testing createCustomer	337
15-32	Result of testing CreateCustomer	338
15-33	Testing RetrieveCustomer	339
15-34	Result of testing RetrieveCustomer	340
16-1	MQDL_To_JDBC_CustomerSync collaboration object	347
16-2	MQDL_To_JTextXML_CustomerSync collaboration object	347
16-3	DataSyncScenario project as viewed in System Manager	349
16-4	Relationships successfully deployed to WebSphere InterChange Server .	350
16-5	New data synchronization scenario library	352
16-6	Import wizard: WebSphere InterChange Server JAR File	353

16-7 Import Wizard: Select WebSphere Interchange Repository Details . . .	354
16-8 Import Wizard: Configure Connector Wizard JTextConnector	355
16-9 Import Wizard: Configure Connector Wizard MQDLConnector	356
16-10 Import Wizard: Configure Connector Migration JDBCConnector	357
16-11 Import Wizard: Conversion Options	358
16-12 Import Wizard Migration Summary	359
16-13 Import Wizard Migration Results	360
16-14 Module created	361
16-15 Update WebSphere Business Integration Adapter for JText	362
16-16 Upgrade WebSphere Business Integration Adapter for JText: Select Projects.	363
16-17 Upgrade WebSphere Business Integration Adapter for JText: Warning. . 364	
16-18 Upgrade WebSphere Business Integration Adapter for JText: Review Changes	365
16-19 Upgrade WebSphere Business Integration Adapter for JDBC	366
16-20 Upgrade WebSphere Business Integration Adapter for JDBC: Select Projects.	367
16-21 Upgrade WebSphere Business Integration Adapter for JDBC: Warning . 368	
16-22 Upgrade WebSphere Business Integration Adapter for JDBC: Review Changes	369
16-23 Modify process: Open process properties.	370
16-24 Modify process: Modify Java Imports	371
16-25 Modify process: Select node	372
16-26 Open Perspective: Java	374
16-27 Select .mon file of process	374
16-28 Modify .mon file for process	375
16-29 Connect MQDL_To_JTextXML_CustomerSync: Remove node 1 in .medflow file	376
16-30 Connect MQDL_To_JTextXML_CustomerSync: Remove node 2 in .medflow file	376
16-31 Connect MQDL_To_JTextXML_CustomerSync: Remove the node in the .mfc file	377
16-32 Connect MQDL_To_JTextXML_CustomerSync: Add reference for mediation flow component	378
16-33 Connect MQDL_To_JTextXML_CustomerSync: Select reference . . .	379
16-34 Connect MQDL_To_JTextXML_CustomerSync: Connect reference in component	380
16-35 Connect MQDL_To_JTextXML_CustomerSync: Connect reference in mediation flow.	381
16-36 MQDLConnector: Set Destination Queue for MQ Binding Export. . . .	382
16-37 MQDLConnector: Set Destination Queue for MQ Binding Import. . . .	382

16-38	JTextConnector Assembly Diagram	383
16-39	Modify Output_Processing.java file	384
16-40	Specify Sequence file path for WebSphere Adapter for Flat Files . . .	385
16-41	Open Perspective: Java	386
16-42	Update JDBC Driver Path: Select file	387
16-43	DB2 Development Center: Launchpad for the stored procedure . . .	389
16-44	DB2 Development Center: Creating a project	390
16-45	DB2 Development Center: Add Connection	391
16-46	DB2 Development Center: Connection Type	392
16-47	DB2 Development Center: Connection	393
16-48	DB2 Development Center: Project created	394
16-49	DB2 Development Center: Importing stored procedures	395
16-50	DB2 Development Center: Specifying the source database	396
16-51	DB2 Development Center: Providing the filter details	397
16-52	DB2 Development Center: Selecting the two stored procedures that you want to modify	398
16-53	DB2 Development Center: Opening a stored procedure that you want to modify in the Editor View	400
16-54	Switching perspectives	401
16-55	Jython script	402
16-56	JDBC provider created in the cell scope	403
16-57	Data source created	404
16-58	Driver type	404
16-59	Helper class	405
16-60	Setting the correct password for authentication	405
16-61	XA DataSource properties	406
16-62	Data source connection testing	407
16-63	Starting WebSphere Process Server	408
16-64	Running the administrative console	409
16-65	Disabling administrative security	410
16-66	Disabling bus security	411
16-67	Opening the profile configuration	412
16-68	Disabling profile security	413
16-69	Restarting the WebSphere Process Server	414
16-70	Adding projects to WebSphere Process Server	415
16-71	Deploying to WebSphere Process Server	416
16-72	Verifying that all modules started	417
16-73	Verifying the migrated relationships	417
16-74	Static relationship instance data in Relationship Manager	418
16-75	WebSphere MQ Explorer	420
16-76	RfhUtil	421
16-77	Set Message Type	422
16-78	JDBCCUSTOMER table in DB2 Control Center	423

16-79	JDBCCUSTOMER table data	423
16-80	Created JTextXMLCustomer output file	424
16-81	Created JTextXMLCustomer output data	424
16-82	Selecting Relationship Manager in the administrative console	425
16-83	Querying the customer relationship	426
16-84	Querying all existing customer relationship instances	426
16-85	Selecting the relationship instance ID created in the test run	427
16-86	Dynamic relationship instance data in Relationship Manager.	428
17-1	SAPJMS_To_PSFT_CustomerSync collaboration object.	434
17-2	DataSyncScenario project as viewed in System Manager	436
17-3	Relationships successfully deployed to WebSphere InterChange Server	437
17-4	New data synchronization scenario library	439
17-5	Import wizard: Selecting the WebSphere InterChange Server Repository JAR File	440
17-6	Import Wizard: Select WebSphere InterChange Repository Details.	441
17-7	Import Wizard Configure Connector Migration: PeopleSoftConnector	442
17-8	Import Wizard Configure Connector Migration: SAPConnector	443
17-9	Import Wizard: Conversion Options	444
17-10	Import Wizard: Migration Summary.	445
17-11	Import Wizard: Migration Results	446
17-12	Module created	447
17-13	Copy Customer BO.	448
17-14	Upgrade WebSphere Business Integration Adapter for mySAP.com	449
17-15	Upgrade WebSphere Business Integration Adapter for mySAP.com: Select Projects	450
17-16	Upgrade WebSphere Business Integration Adapter for mySAP.com: Warning	451
17-17	Upgrade WebSphere Business Integration Adapter for mySAP.com_Review Changes.	452
17-18	Upgrade WebSphere Business Integration Adapter for mySAP.com: Copy the BO and BG files	453
17-19	Upgrade WebSphere Business Integration Adapter for PeopleSoft	454
17-20	Upgrade WebSphere Business Integration Adapter for PeopleSoft: Select Projects.	455
17-21	Upgrade WebSphere Business Integration Adapter for PeopleSoft: Warning	456
17-22	Upgrade WebSphere Business Integration Adapter for PeopleSoft: Review Changes	457
17-23	Upgrade WebSphere Business Integration Adapter for PeopleSoft: Copy back BO and BG files	458
17-24	In the Business View, opening the assembly module of WebSphere Adapter for SAP Software	459

17-25	Deleting the synchronized EIS binding of WebSphere Adapter for SAP Software	459
17-26	Open the Input_Async_Processing Java component	460
17-27	Modify BPEL: Select process	462
17-28	Modify BPEL: Modify Java Imports	462
17-29	Modify BPEL: Select node	463
17-30	Open Perspective: Java	465
17-31	Select the CustomerSync_From_bpel.mon file	466
17-32	Modify the CustomerSync_From_bpel.mon file	467
17-33	DB2 Development Center: Launchpad for the stored procedure	468
17-34	DB2 Development Center: Creating a project	469
17-35	DB2 Development Center: Add Connection	470
17-36	DB2 Development Center: Connection Type	471
17-37	DB2 Development Center: Connection	472
17-38	DB2 Development Center: Project created	473
17-39	DB2 Development Center: Importing stored procedures	474
17-40	DB2 Development Center: Specifying the source database	475
17-41	DB2 Development Center: Providing the filter details	476
17-42	DB2 Development Center: Selecting the stored procedures that you want to modify	477
17-43	DB2 Development Center: Opening a stored procedure that you want to modify in the Editor View	478
17-44	Switching perspectives	479
17-45	Jython script	480
17-46	JDBC provider created in cell scope	481
17-47	Data source created	482
17-48	Driver type	482
17-49	Helper class	483
17-50	Setting the correct password for authentication	483
17-51	XA DataSource properties	484
17-52	Data source connection testing	485
17-53	Starting WebSphere Process Server	486
17-54	Running the administrative console	487
17-55	Disabling administrative security	488
17-56	Disabling bus security	489
17-57	Opening the profile configuration	490
17-58	Disabling profile security	491
17-59	Restarting the WebSphere Process Server	492
17-60	Adding projects to WebSphere Process Server	493
17-61	Deploying to WebSphere Process Server	494
17-62	Verifying that all modules started	494
17-63	Verifying the migrated relationships	495
17-64	Static relationship instance data in Relationship Manager	496

17-65	SAP Front-End GUI Tool: Login	497
17-66	SAP Front-End GUI work panel	498
17-67	IDoc List	499
17-68	IDoc List: Search result	500
17-69	SAP Front-End GUI Tool work panel: Navigate	500
17-70	Search IDoc	501
17-71	Edit the IDoc	501
17-72	Send the IDoc	502
17-73	SAP Front-End GUI Tool work panel: Navigate	502
17-74	Execute Test Event	503
17-75	Opening and viewing the PS_WBI_CUSTOMER, PS_WBI_ADDRESS, and PS_WBI_PHONE data	504
17-76	Selecting Relationship Manager in the administrative console	505
17-77	Querying the customer relationship	506
17-78	Querying all of the existing customer relationship instances	506
17-79	Selecting the Relationship instance ID created in the test run	507
17-80	Dynamic relationship instance data in Relationship Manager	508
18-1	Create Customer Collaboration flow diagram	515
18-2	Retrieve Customer Collaboration flow diagram	515
18-3	RetrieveCustomerCollab_FromRetrieve business process diagram ..	517
18-4	Fault handler on TOJDBC.Retrieve	519
18-5	Adding the JDBCFault variable	520
18-6	Adding the WhileLoop activity	521
18-7	Question window prompting to change the type of snippet	521
18-8	Adding a fault handler	523
18-9	HTMLInterface	524
18-10	Changing the Assign From and Assign To properties	524
18-11	Assigning from HTMLRetrieveCustomerBG to ToJDBCBusObj_var CustomerRetrieveBG	525
18-12	RetrieveCustomerCollab_FromRetrieve process interface	526
18-13	Execute_PortType interface	526
18-14	Modified part of the RetrieveCustomer_FromRetrieve process	527
18-15	Importing the HashMap class	528
18-16	Importing the BOFactory class	528
18-17	Importing the ServiceBusinessException class	530
18-18	JDBCCustomer table data	531
18-19	RetrieveCustomer test client	531
18-20	Exception handling HumanTask	532
18-21	HumanTask Input and Output message	533
18-22	RetrieveCustomer result	534
18-23	Example of a routing logic implementation in WebSphere InterChange Server	536
18-24	The first collaboration object has the TargetFilter property set to the value	

of TARGETAPP1	537
18-25 The second collaboration object has the TargetFilter property set to the value of TARGETAPP2	538
18-26 Filtering logic implemented in the collaboration template CustomerSync.	539
18-27 Global view of the migration of the WebSphere InterChange Server routing example within WebSphere Integration Developer	540
18-28 Migration of the collaboration objects with WebSphere Integration Developer	542
18-29 Migration of the inbound connector module	542
18-30 Migration of the filtering logic at the collaboration level.	544
18-31 Custom multicast in the inbound adapter module: New reference ...	545
18-32 Custom multicast in the inbound adapter module: Original implementation in the MFC.	546
18-33 Custom multicast in the inbound adapter module: New implementation in the MFC	547
18-34 Optimized migrated solution for maintainability and dynamicity	549
18-35 The TestActEdBasic map	552
18-36 Custom transformation in a map that is implemented with Activity Editor.	552
18-37 Generated Java code corresponding to the custom transformation designed with the Activity Editor.	553
18-38 Business graph map that is generated by the migration tools	554
18-39 Business object map that is generated by the migration tools	555
18-40 Java code is migrated but the visual design of the Activity Editor is lost	556
18-41 Transform has been rewritten using WebSphere Integration Developer Visual Editor	558
19-1 Logging and Tracing pane of the administrative console	561
19-2 Changing the log detail levels	562
19-3 Configuration and Runtime tabs	565
19-4 Diagnostic trace	566
19-5 SystemOut and SystemErr logs	567
A-1 Customer synchronization example from Clarify to SAP	574
A-2 Connector agent configuration for secured JMS communication	575
A-3 Connector EJB modules in the Java EE perspective	576
A-4 Modifying the deployment descriptor and security roles.	577
A-5 Setting up a name for the Security Role	578
A-6 Setting up the Security Identity	579
A-7 Setting up the Role name for Security Identity	580
A-8 Selecting the correct EJB components for the Clarify connector	581
A-9 Selecting the correct EJB components for the SAP connector.	582
A-10 Go to the Buses view in the administrative console	583

A-11	Selecting Allow the use of all defined transport channel chains	584
A-12	Selecting Show me all installation options and parameters	585
A-13	An example of setting up the authentication alias for the activation spec of the MDBs	586
A-14	Setting up the authentication alias for the connection factory	588
A-15	Mapping the security role with an actual user.	589
A-16	Mapping the RunAs role to an actual user	590
B-1	Access EJB example overview.	593
B-2	Importing the AccessEJB project interchange	596
B-3	Adding the data handler JAR files to the build path	597
B-4	AccessEJB assembly diagram	598
B-5	Assembly diagram of the DynamicRouting module	599
B-6	Generated projects after migration	601
B-7	End-to-end flow of events through the examples	602
B-8	Adding a dependent library.	603
B-9	Modified assembly diagram of the DynamicRouting module	604
B-10	Selector file in the Java perspective	605
B-11	Additions to the selector file	606
B-12	Creating a new Java implementation file	607
B-13	Specifying the Java implementation file in the component.	608
B-14	Changes in the implementation file.	609
B-15	Changes made in the AccessServlet.java file.	610
B-16	Loading the AccessServlet client	612
B-17	Reply from the VTC	612
B-18	Response received on AccessServlet	613
B-19	Output in the SystemOut.log file	613
C-1	WebSphere Business Integration Adapters with WebSphere Process Server.	617
C-2	Prebuilt inbound example on WebSphere InterChange Server	618
C-3	WebSphere Business Integration Adapter with WebSphere Process Server: EJB	619
C-4	Creating a new workspace	620
C-5	Selecting the Import option.	621
C-6	Importing the WebSphere InterChange Server Repository file	622
C-7	Select WebSphere InterChange Server Repository Detail.	623
C-8	Configure Connector Migration.	624
C-9	Conversion Options	625
C-10	Migration Summary	626
C-11	Migrated projects	627
C-12	Modifications on the connector configuration file	628
C-13	Visual Test Connector to send the BIA_SampleMusicCartBO.bo	630
C-14	WebSphere Process Server binding interacting directly with a transport or protocol.	631

C-15	CustomerRetrieve collaboration object for outbound processing	632
C-16	Operational model for the EJB binding example	633
C-17	Creating a new workspace	634
C-18	Select WebSphere InterChange Repository Details.	635
C-19	Configure Connector Migration.	636
C-20	Migration Summary	637
C-21	Migration Results	638
C-22	Migrated projects	639
C-23	Enterprise JavaBean Binding	640
C-24	EJBRequestHandlerImpl	641
C-25	Start WebSphere Process Server in WebSphere Integration Developer . . 645	
C-26	Test setup.	646
C-27	Execution trace.	647
C-28	Return object mapped to business graph.	648

Tables

2-1	SIMM level comparison for WebSphere InterChange Server and WebSphere Process Server	21
4-1	Basic migration project considerations	63
7-1	Mapping the most usual WebSphere InterChange Server components .	126
7-2	Artifact mapping for other less common WebSphere InterChange Server components	127
7-3	Other WebSphere InterChange Server features compared to WebSphere Process Server	128
8-1	Adapter to adapter correlation	135
8-2	Adapter to binding correlation	135
8-3	Other cases	136
8-4	Upgrade path examples	138
9-1	Integration logic and business logic implementations	157
9-2	Comparison of the migration approaches	171
10-1	Overview of migration pathways	180
10-2	Migration options for WebSphere Business Integration Adapters	200
10-3	Command-line options of the reposMigrate utility	207
11-1	Generated artifacts after migrating	217
11-2	SDO business object key concepts	222
11-3	Mapping rules in WebSphere InterChange Server and their descriptions .	223
11-4	Mapping rules in WebSphere Process Server and their description . .	224
11-5	WebSphere Process Server relationship key concepts	238
14-1	Software versions used	281
15-1	Software versions used	295
16-1	Software versions used	346
17-1	Software versions used	433
18-1	Software versions used	513
19-1	Components and packages	563
B-1	Products installed in the software environment	592
C-1	Software versions used	617

Examples

10-1 Exporting WebSphere InterChange Server artifacts by using the repos_copy command	205
12-1 URL generated by the Migration Wizard	261
12-2 Modified URL	261
14-1 MQ_Setup.txt	289
14-2 MQ command	289
15-1 Old BO import	312
15-2 New BO import	312
15-3 Old copy code in Call JDBC java snippet activity	312
15-4 New copy code in Call JDBC java snippet activity	312
15-5 Old copy code in Copy Result to triggeringBO java snippet activity	313
15-6 New copy code in Copy Result to triggeringBO java snippet activity	313
15-7 Old copy code in Call MQ java snippet activity	313
15-8 New copy code in Call MQ java snippet activity	313
15-9 Old BO import	314
15-10 New BO import	314
15-11 Old copy code in UNNAMED_ACTIVITY_6 java snippet activity	314
15-12 New copy code in UNNAMED_ACTIVITY_6 java snippet activity	314
15-13 Old copy code in UNNAMED_ACTIVITY_8 java snippet activity	315
15-14 New copy code in UNNAMED_ACTIVITY_8 java snippet activity	315
15-15 Old response BG	315
15-16 New response BG	316
15-17 Old response BG verb	316
15-18 New Response BG verb	316
15-19 Old response BO	316
15-20 New response BO	316
15-21 Old response BG	317
15-22 New response BG	317
15-23 Old response BG verb	317
15-24 New Response BG verb	317
15-25 Old response BO	317
15-26 New response BO	318
15-27 JDBC driver classpath	330
15-28 Old WebService address	336
15-29 New WebService address	336
15-30 Old WebService address	336
15-31 New WebService address	336
16-1 Modifying the CustomerSync_From _Modify node Initialize_parameters	

Java snippet	373
16-2 BusinessObjectBG_Sync in Output_Processing.java	383
16-3 Output_Processing.java	383
17-1 Input_Async_Processing Java component	461
17-2 Modifying CustomerSync_From _Modify node Initialize_parameters Java snippet	464
18-1 WhileLoop snippet.	521
18-2 setJDBCFault snippet	522
18-3 clearJDBCFault snippet	522
18-4 setJDBCFault1 snippet	525
18-5 Adding HashMap and BOFactory to a WBI2WSA Java component	528
18-6 CustomerRetrieveBG_Sync() method in the Output_Processing Java component	529
18-7 Add createWICSFault() method in the Output_Processing Java component	529
18-8 Input data of CreateCustomer.	530
19-1 WebSphere Process Server SystemOut.log	568
19-2 Authentication error	569
19-3 RetrieveCustomer errors.	570
B-1 Input XML file	599
C-1 Expected result in SystemOut.log	630
C-2 Sample for EJBCustomerRepoSessionBeanBG_Async method	642
C-3 Sample for EJBCustomerRepoSessionBeanBG_Sync method	642
C-4 Sample to convert from a Java object to SDO	643
C-5 Sample to convert from SDO to a Java object	644

Notices

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

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

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

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

AIX®	IBM®	Redbooks (logo)  ®
ClearCase®	OMEGAMON®	System z®
DB2 Universal Database™	Parallel Sysplex®	Tivoli®
DB2®	Rational®	WebSphere®
developerWorks®	Redbooks®	z/OS®
HACMP™	Redpapers™	

The following terms are trademarks of other companies:

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

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

ABAP, BAPI, mySAP, mySAP.com, SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

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

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

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

Preface

IBM® WebSphere® Process Server is the next generation business process integration server that has evolved from proven business integration concepts, application server technologies, and the latest open standards. In this IBM Redbooks® publication, we provide guidance about how to migrate IBM WebSphere InterChange Server and WebSphere Business Integration Adapters to WebSphere Process Server. We provide this guidance for WebSphere InterChange Server users in general, and particularly for project managers, architects, and developers.

In this book, we discuss the critical concepts that are related to integration solution architecture, migration project planning, and the technical implementation approach. We also discuss the capabilities of the migration tools. We include various migration examples that show how you can upgrade WebSphere InterChange Server and WebSphere Business Integration Adapters to WebSphere Process Server and WebSphere Adapters.

We divide this IBM Redbooks publication into four parts:

First, in Part 1, “Migration concepts” on page 1, we introduce the high-level concepts that are required to comprehend the migration roadmap. We begin with a software architecture perspective. We categorize the various design patterns that compose an integration solution and explain how they are implemented in WebSphere InterChange Server and WebSphere Process Server. Then we take a project perspective and provide a recommended approach for migration planning.

Next, in Part 2, “Migration implementation concepts” on page 71, we discuss relevant concepts to start the migration implementation. We begin with a product overview, introduce key technologies and capabilities, and explain how they relate to each other between the products. Then we provide recommendations to select the appropriate path with regard to the main technical challenges of the migration implementation.

Then in Part 3, “Migration tooling” on page 177, we cover the standard migration tools that are available for upgrading from WebSphere InterChange Server to WebSphere Process Server. We discuss the migration tools, including WebSphere Integration Developer, the Migration Wizard, the reposMigrate utility, and the tools that are dedicated to adapter migration. We explain the post migration tasks to execute for certain artifacts, such as database connection

pools and relationships. We conclude with the best practices to ease the usage of the migration tools and to optimize the generated artifacts

Finally, in Part 4, “End-to-end technical solutions” on page 277, we provide step-by-step instructions to migrate three end-to-end integration solutions that are based on the commonly used data access and data synchronization interaction patterns. We follow a phased approach to migration by first migrating the WebSphere InterChange Server components as is by using the migration tool. Then we upgrade the WebSphere Business Integration Adapters to either WebSphere Adapters or native bindings as appropriate. We further enhance and optimize the migrated artifacts based on WebSphere Process Server implementation best practices when applicable.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Hursley Center.



Saida Davies is a Project Leader for the ITSO and has published several IBM Redbooks and Redpapers™ publications on WebSphere Business Integration, Web services, and WebSphere Service Oriented Middleware using multiple platforms. Saida has experience in the architecture and design of WebSphere MQ solutions, extensive knowledge of the z/OS® operating system, and a detailed working knowledge of both IBM and independent software vendors' operating system software. Prior to joining the ITSO, she was a Senior IT Specialist who was responsible for the development of services for WebSphere MQ within the z/OS and Microsoft® Windows® platform. This position 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 has received Bravo awards for her project contributions. Saida has a degree in computer studies, and her background includes IBM System z programming. She supports, contributes, and participates in Women in Technology activities and meetings.



Julien Nagel is a Business Integration Consultant for IBM Software Group. Since joining IBM ten years ago, he has worked on various programming languages and software products, including COBOL on System z®, Java™, ATG Dynamo, WebSphere Application Server, WebSphere InterChange Server, and WebSphere Process Server. In the last seven years, his contribution has been in Business Integration Software services. His assignments at clients have ranged from full life-cycle, long-term, large-scale integration projects from early architecture design to go-live assistance and audits. During these engagements, Julien has acquired deep knowledge of Enterprise Application Integration (EAI) and service-oriented architecture (SOA). He takes pride in having assisted many clients in implementing and going live successfully with WebSphere InterChange Server and WebSphere Process Server. His current focus is on providing enablement to ease migration for WebSphere InterChange Server clients to WebSphere Process Server in SouthWest Europe. Julien has an Engineering School Diploma in Fluid Mechanics.



Raja Sreenivasa is a Software Engineer working in the IBM WebSphere Adapters development team. He joined IBM five years ago. He has worked on various IBM products, including WebSphere MQ, WebSphere Message Broker, WebSphere Application Server, WebSphere Adapters, WebSphere InterChange Server and WebSphere Process Server. He contributed mainly on Business Integration Software Services. He holds a Masters degree in Business Applications from SriVenkateswara University, in Tirupati.



Hui WH Wang joined IBM China Development Lab five years ago and is a Staff Software Engineer. Her experience has been in verification testing of WebSphere InterChange Server and WebSphere Process Server. In 2007, Hui passed the Project Management Professional (PMP) certification from the Project Management Institute (PMI). Currently, she is the team lead and project manager for WebSphere Business Integration Heritage support team, responsible for WebSphere InterChange Server and WebSphere Business Integration Server Express L3 support. Hui got a Bachelor degree in Computer Science from Beijing University of Technology.



Meng Wang is a Software Engineer from IBM China Development Lab, who has worked on WebSphere Process Server for more than four years. Currently, his main responsibility is the development for Heritage products Migration to WebSphere Process Server. Before that, he had three years of WebSphere Process Server function verification testing experience, and one year of WebSphere Process Server runtime development experience. He has skills in WebSphere Process Server problem determination. He is an IBM Certified System Administrator of WebSphere Application Server Network Deployment V6.1. He has a Bachelor degree in Computer Science from Beijing University of Technology.



Ji Yu is a Staff Software Engineer for IBM China Development Lab. She joined IBM six years ago and is currently working in the WebSphere Process Server system and functional test team. As the team lead, her main responsibility is performing system and functional testing for migrating WebSphere InterChange Server to WebSphere Process Server and migrating WebSphere Business Integration Server Foundation to WebSphere Process Server. Ji Yu has been working on the WebSphere Process Server product for four years. She is an IBM Certified System Administrator of WebSphere Application Server Network Deployment V6.0 and WebSphere Process Server Network Deployment V6.0. She has written two developerWorks® articles and coauthored three IBM Redbooks publications; *IBM WebSphere InterChange Server Migration to WebSphere Process Server*, SG24-7415, *Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server V6.1 and Best Practices*, SG24-7673-00, and *Migrating WebSphere Business Integration Server Foundation to WebSphere Process Server V6.2 and Best Practices*, SG24-7673-01. She holds a Bachelor of Software Engineering degree from Tsinghua University of China.



Jie ZJ Zhao joined IBM China Development Lab about three years ago. His experience mainly focused on verification testing of WebSphere Adapters and the SOA feature pack of WebSphere Application Server. He also has one year of experience in WebSphere Adapter Foundation Class development. He is an IBM Certified System Administrator of WebSphere Application Server Network Deployment V6.1. He has published two papers in China developerWorks related to Java EE Connector Architecture (JCA) binding fault handling and customizing the event handling of WebSphere Adapter. Currently, he is the team lead for the WebSphere Adapter FVT team, responsible for test plan, test resource arrangement, and technical support.

The ITSO and the team express special thanks to the following individuals for providing the venue and resources for this project.

BPM Runtime Development & WebSphere Enterprise Service Bus team
IBM China Development Lab, IBM Beijing, China

The team thanks the following people for their assistance and contributions to this project:

Bu Feng Hou
Manager, IBM China Development Lab, IBM Beijing, China

Yan Zhou
Senior Manager, IBM China Development Lab, IBM Beijing, China

Jun Jie Lu
Manager, IBM China Development Lab, IBM Beijing, China

Wen Chao Li
Dept Manager, IBM China Development Lab, IBM Beijing, China

Shuo Zhang
Staff Software Engineer, IBM China Development Lab, IBM Beijing, China

Jie Zhang
Software Engineer, IBM China Development Lab, IBM Beijing, China

Mahesh Sharma
IT SOA Architect, Solution Design, WebSphere, WebSphere Business
Integration Integration Architect
IBM Software Group, Application and Integration Middleware Software, IBM
Burlingame, USA

Gopalakrishnan Balasubramanian
Manager, IBM Software Group, Application and Integration Middleware
Software, IBM Bangalore, India

Jerome Carrasco
Manager, IBM Sales & Distribution, Software Sales, IBM Courbevoie, France

Ping Hao
Dept Manager, IBM China Development Lab, IBM Beijing, China

Yun Diao
Dept Manager, IBM China Development Lab, IBM Beijing, China

Qiang Xu
Dept Manager, IBM China Development Lab, IBM Beijing, China

Brian Petrini
Senior I/T Architect, WebSphere Business Integration, IBM Software Group,
Application and Integration Middleware Software, IBM San Diego, USA

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

Authors of the first edition, *IBM WebSphere InterChange Server Migration to
WebSphere Process Server*, SG24-7415-00, published in December 2008, are:

- Saida Davies
Gopal Balasubramanian, IBM India
Pamela H. Fong, IBM USA
Susan Herrmann, IBM Germany
Jeremy Kong, IBM USA
Kevin McKenney, IBM USA
Julien Nagel, IBM France
Frank M. Neumann, IBM Germany
Jayasaikumar Padimiti, Miracle Software Systems USA
Pravesh Patel, IBM USA
Ashish Rahrurkar, IBM USA
Vishnu Selvaraj, IBM USA
Sascha Schefenacker, IBM Germany
Iftekhhar Siddiqui
Sandeep Sinha, IBM USA
Ratan Siripurapu, Sarasu IT Solutions, LTD USA

Kiran S. Sundar
Tassanee Supakkul, IBM USA
Chang Liang Xing, IBM China
Ji Yu/China, IBM China
Qu Ji Zhang, IBM China

Become a published author

Join us for a two- to six-week residency program. Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, IBM 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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

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

ibm.com/redbooks

- Send your comments in an e-mail to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Summary of changes

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

Summary of Changes
for SG24-7415-01
for Migrating WebSphere InterChange Server and Adapters to WebSphere
Process Server V6.2
as created or updated on August 17, 2009.

The previous edition is *IBM WebSphere InterChange Server Migration to WebSphere Process Server*, SG24-7415-00:

<http://www.redbooks.ibm.com/abstracts/sg247415.html?Open>

August 2009, Second Edition

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

New information

► Enhanced Migration Wizard

The Migration Wizard available with WebSphere Integration Developer V6.2 brings significant enhancements, especially for adapter migration. The new Migration Wizard supports three choices for adapter migration:

- Adapters can still be migrated to Java Message Service (JMS) bindings, which allow reusing the original WebSphere Business Integration Adapters.
- Technical adapters can be directly migrated to native bindings if they are supported as native bindings (MQ, JMS, HTTP, Web services, Enterprise JavaBeans™ (EJB™)).
- Also, if an adapter is detected as one that has an equivalent Java EE Connector Architecture (JCA) Adapter, it can be directly migrated by the Adapter Migration Wizard in WebSphere Integration Developer (JText, EMail, Java Database Connectivity (JDBC™), SAP®, and PeopleSoft® adapters, for example).

Changed information

All the content of the book has been updated to take into account new WebSphere Process Server V6.2 features that are relevant to migration.

The structure of the book has been modified and simplified:

- ▶ A new Chapter 6, “Product feature comparison” on page 109 has been added in Part 2, “Migration implementation concepts” on page 71.
- ▶ Technical examples (Part 4) and end-to-end technical solutions (Part 5) in the first edition of the book have been merged into a new Part 4, “End-to-end technical solutions”:
 - Most of the scenarios described in the old Part 4 , Technical examples, have been removed or rewritten, because they were not accurate any longer with WebSphere Integration Developer V6.2.
 - When and where necessary, those technical scenarios have been rewritten to fit in the new end-to-end solutions covered in Part 4, “End-to-end technical solutions”.
 - Additionally, we added a new solution to demonstrate Data Synchronization with application adapters to integrate SAP and PeopleSoft to the two technical end-to-end solutions demonstrated in the the first edition of book.
- ▶ Several specific technical scenarios that are not covered in the new Part 4, “End-to-end technical solutions” have been updated for WebSphere Integration Developer V6.2 and moved into the appendixes:
 - Appendix A, “WBI Adapters in a secured WebSphere Process Server environment” on page 571
 - Appendix B, “Access EJB migration” on page 591
 - Appendix C, “Migrate WBI Adapter for EJB Architecture” on page 615



Part 1

Migration concepts

In Part 1, we introduce the high-level concepts that are required to comprehend the migration roadmap from the IBM WebSphere InterChange Server to the WebSphere Process Server. This part contains the following chapters:

- ▶ Chapter 1, “Introduction to this book” on page 3, provides preliminary information, such as the executive summary, the scope, and the intended audience of the book.
- ▶ Chapter 2, “Integration background” on page 9, takes a software solution evolution perspective and explains how an enterprise application integration solution, such as the WebSphere InterChange Server, evolves naturally toward a service-oriented architecture (SOA) or Business Process Management (BPM) platform, such as WebSphere Process Server.
- ▶ Chapter 3, “Usage patterns” on page 23, provides a categorization of architectural integration concepts and explains how WebSphere InterChange Server and WebSphere Process Server realize implementations of these concepts.
- ▶ Chapter 4, “Migration planning” on page 33, provides a recommended approach to plan the migration from a project perspective.

Archived

Introduction to this book

In this chapter, we discuss the following topics:

- ▶ 1.1, “Executive summary” on page 4
- ▶ 1.2, “Scope of the book” on page 5
- ▶ 1.3, “Intended audience” on page 5
- ▶ 1.4, “What is covered” on page 6
- ▶ 1.5, “What is not covered” on page 7
- ▶ 1.6, “Assumptions” on page 7

1.1 Executive summary

With the introduction of new integration technology and new products, such as IBM WebSphere Process Server and WebSphere Enterprise Service Bus, users are given standards-compliant and state-of-the-art tools to integrate enterprise business applications. The new technology, based on open standards, is well documented and familiar to developers who are hired from the marketplace or universities. Products based on such standards reduce the cost of skill development. Only the new product skill set needs to be built up and reconditioned. This approach contrasts an environment where existing products and new products have to run in conjunction to provide business integration solutions. In this case, the IT department must be skilled in both existing and new applications, products, and technologies.

The cost of hardware is another important aspect to consider for product upgrades. Migrating to new products can allow the consolidation of the hardware and software landscape. Developers need only one set of tools and computers for their development environment. A consolidated landscape also offers lower cost in terms of hardware maintenance, software licenses, and energy management.

Overall, the upgrade decision needs to be determined by various aspects of the total cost of ownership (TCO). Today, enterprises use IBM WebSphere Business Integration WebSphere InterChange Server and Adapters to interface with various Enterprise Information Systems (EISs) in complex and heterogeneous environments. With the introduction of new products, such as WebSphere Process Server and WebSphere Enterprise Service Bus, consider the benefits of an upgrade.

In this book, we provide guidance and upgrade considerations for WebSphere Business Integration in regard to WebSphere InterChange Server and Adapters. This book is written for WebSphere InterChange Server users in general, and particularly for project managers, architects, and developers.

The recommended migration roadmap involves the following preliminary tasks:

- ▶ Study up on WebSphere Process Server as soon as possible.
- ▶ If possible, use WebSphere Process Server for new projects before migrating an existing project.
- ▶ Investigate the new functionalities of WebSphere Process Server. The migration needs to be used as an opportunity to add value to the current IT environment and the business that it supports.
- ▶ Invest in an assessment and preparation phase to prepare for migration implementation.

Then begin with the migration implementation.

1.2 Scope of the book

In this book, we provide guidance about how to migrate WebSphere InterChange Server and WebSphere Business Integration Adapters to WebSphere Process Server and WebSphere Adapters. We begin by discussing the critical concepts that are related to an integration solution architecture, migration project planning, and a technical implementation approach. Then we go into more detail and describe the capabilities of the migration tools. Finally, we show various examples of how you can upgrade WebSphere InterChange Server and WebSphere Business Integration Adapters to WebSphere Process Server and WebSphere Adapters.

1.3 Intended audience

This book targets a broad audience beginning with managers, IT architects, solution designers, and integration developers who have been involved with WebSphere InterChange Server projects.

The following chapters are intended for architects and solution designers, and are also useful to developers:

- ▶ Chapter 2, “Integration background” on page 9 provides the integration history.
- ▶ Chapter 3, “Usage patterns” on page 23 examines the usage patterns for WebSphere InterChange Server.
- ▶ Chapter 4, “Migration planning” on page 33, is intended for project management.
- ▶ Chapter 5, “Product overview” on page 73, through Chapter 19, “Technical solutions: Troubleshooting” on page 559, are intended for developers. The content requires an intermediate technical skill level with regard to WebSphere Adapters, WebSphere Integration Developer, and WebSphere Process Server.

1.4 What is covered

This book is divided into the following parts:

- ▶ Part 1, “Migration concepts” on page 1

In this part, we introduce the high-level concepts that are required to comprehend the migration roadmap. We begin with a software architecture perspective. We categorize the various design patterns that compose an integration solution and explain how they are implemented in WebSphere InterChange Server and WebSphere Process Server. Then we take a project perspective and provide a recommended approach for migration planning.

- ▶ Part 2, “Migration implementation concepts” on page 71

In this part, we discuss relevant concepts to start the migration implementation. We begin with a product overview, introduce key technologies and capabilities, and explain how they relate to each other between the products. Then we provide recommendations to select the appropriate path with regard to the main technical challenges of the migration implementation.

- ▶ Part 3, “Migration tooling” on page 177

In this part, we discuss the standard migration tools that are available for upgrading from WebSphere InterChange Server to WebSphere Process Server. We cover the migration tools, including WebSphere Integration Developer, the Migration Wizard, the reposMigrate utility, and the tools that are dedicated to adapter migration. We explain the post migration tasks to execute for certain artifacts, such as database connection pools and relationships. We conclude with the best practices to ease the usage of the migration tools and to optimize the generated artifacts.

- ▶ Part 4, “End-to-end technical solutions” on page 277

In this part, we provide step-by-step instructions to migrate three end-to-end integration solutions that are based on the commonly used data access and data synchronization interaction patterns. We follow a staged approach to migrate WebSphere InterChange Server components to WebSphere Process Server components, and WebSphere Business Integration Adapters to native bindings or WebSphere Adapters. We further enhance and optimize the migrated artifacts based on WebSphere Process Server implementation best practices when applicable.

We demonstrate these solutions through the following activities:

- We perform premigration preparation of the hardware and software, as well as setup steps that you must perform before continuing with the migration examples.

- We migrate a data access integration scenario to WebSphere Process Server that was originally designed for and implemented on WebSphere InterChange Server.
- We migrate two data synchronization integration scenarios to WebSphere Process Server that were originally designed for and implemented on WebSphere InterChange Server. The first scenario focuses on technology adapters, and the second scenario focuses on application adapters.
- We use WebSphere Process Server best practices to enhance migrated integration solutions.
- We troubleshoot the issues that are encountered when running the end-to-end integration scenarios.

Also, we provide additional technical information through appendixes to cover the following specific topics:

- Using WebSphere Business Integration Adapters within a secured WebSphere Process Server environment
- Migration of the Access Enterprise JavaBeans (EJB)
- Upgrading WebSphere InterChange Server Access EJBs to EJB binding

1.5 What is not covered

This book does not cover details about the installation of software products, including WebSphere InterChange Server, IBM DB2® Universal Database™, WebSphere Business Integration Adapters, WebSphere Integration Developer, and WebSphere Process Server. Therefore, no step-by-step installation instructions are included.

1.6 Assumptions

In this book, we make the following assumptions:

- ▶ You are familiar with WebSphere InterChange Server and its related products:
 - WebSphere InterChange Server toolkit
 - WebSphere Business Integration Adapters framework
 - WebSphere Business Integration Adapters
 - WebSphere MQ

- ▶ You have a clear understanding of the target platform and technologies:
 - WebSphere Process Server concepts
 - WebSphere Integration Developer

To follow the step-by-step instructions of the technical examples described in this book, you must have a development environment up and running for both WebSphere InterChange Server and WebSphere Process Server.

Integration background

To clearly understand the differences between IBM WebSphere InterChange Server and WebSphere Process Server, you must understand the integration history. One model that you can use to understand integration history, and where both WebSphere InterChange Server and WebSphere Process Server fit into the current spectrum of integration, is the Service Integration Maturity Model (SIMM).

In this chapter, we discuss SIMM and the various SIMM levels. We include the following sections:

- ▶ 2.1, “Service Integration Maturity Model” on page 10
- ▶ 2.2, “SIMM level 1: Silo” on page 13
- ▶ 2.3, “SIMM level 2: Integrated (point-to-point)” on page 14
- ▶ 2.4, “SIMM level 3: Componentized (hub and spoke)” on page 15
- ▶ 2.5, “SIMM level 4: Services” on page 16
- ▶ 2.6, “SIMM level 5: Composite services” on page 17
- ▶ 2.7, “SIMM level 6: Virtualized services” on page 19
- ▶ 2.8, “SIMM level 7: Dynamically reconfigurable services” on page 20
- ▶ 2.9, “Summary” on page 21

2.1 Service Integration Maturity Model

The Service Integration Maturity Model (SIMM) helps to create an incremental transformation roadmap toward higher levels of service integration maturity. SIMM is used to determine which characteristics are desirable to achieve by attaining a new level of maturity. It determines whether problems encountered at a given level of maturity can be solved by evolving to the next level of service integration maturity.

The SIMM has the following purpose:

- ▶ Assess a client's current state in service integration and flexibility (including services orientation) and their desired or future state, for a line of business or enterprise
- ▶ Provide a model to assist a client in determining its architectural strategy when adopting service orientation for the following reasons:
 - Improve pain points in areas of flexibility or integration
 - Provide an architectural roadmap for one or more initiatives in existing transformation, package implementation or integration, application renovation, or systems integration
- ▶ Provide a model for determining scope, focus, and incremental steps (that is, an architectural roadmap) for a transformation toward service orientation with increasing integration maturity with defined business benefits
- ▶ Provide a framework to surface insights and identify IT improvements in areas of component development, service-oriented architecture (SOA), services integration, and IT processes (for example, governance) improvements

SIMM originates from IBM Global Business Services¹. Further detail is available from the following IBM developerWorks Web page:

<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-simm/>

The model has been donated to The Open Group (TOG) and is known as the *Open Group Service Integration Maturity Model*. For further detail, refer to the following Web address:

<http://www.opengroup.org/projects/osimm/>

¹ The SIMM description is from a presentation called "Service Integration Maturity Model (SIMM): Introduction" that was presented at The Open Group IT Architect Practitioners Conference in Miami, Florida, in 2006. The authors of the presentation are Ali Arsanjani, Ph.D., the Chief Architect, SOA Center of Excellence, and Jorge Diaz, Executive IT Architect, SCITA, both with IBM. For details, refer to <http://www.openinnovations.us/events/q306/arsanjani-diaz.htm>

With SIMM, companies can plan their current and strategic future positions in terms of business in the following domains:

► Business view

The business view domain looks at the maturity of the business architecture, the relationship between business and IT, and how value can be achieved by moving to a service-oriented paradigm for the business.

► Organization

The organization domain looks at the maturity of the enterprise, business units, or both in the context of organizational structure, processes, mechanisms, learning and knowledge enablement, and governance in support of service orientation.

► Methods

The methods domain looks at the maturity of the enterprise, business units, or both in their use of system development methods, processes, and related development tools to support the SOA life cycle. This domain includes project management and project estimation considerations.

► Application

The application domain looks at the maturity of the application portfolio to use service orientation. It focuses on the use of services for sharing and reuse of business functionality across business units and the ability to flexibly interchange functionality to meet changing business needs.

► Architecture

The architecture domain looks at the maturity of various views of the architecture, specifically enterprise and application architecture to support service orientation.

► Information

The information domain looks at the maturity of the information, data architecture, and management to support service orientation. It includes the notion of information as a service where service orientation combined with information architecture provides improved value.

► Infrastructure

The infrastructure domain provides a view of the maturity of the IT environment to support a service-oriented ecosystem. Maturity addresses service monitoring and management, service security, and the technologies and tools to support the nonfunctional and operational requirements that are needed to operate.

SIMM also has the following integration service levels:

1. Silo
2. Integrated
3. Componentized
4. Services
5. Composite services
6. Virtualized services
7. Dynamically reconfigurable services

In this section, we focus on the service levels to ascertain how they reflect within WebSphere InterChange Server and WebSphere Process Server. Figure 2-1 represents SIMM.


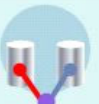





						
Silo	Integrated	Componentized	Services	Composite Services	Virtualized Services	Dynamically Re-Configurable Services
Function Oriented	Function Oriented	Function Oriented	Service Oriented	Service Oriented	Service Oriented	Service Oriented
Ad hoc IT Governance	Ad hoc IT Governance	Ad hoc IT Governance	Emerging SOA Governance	SOA and IT Governance Alignment	SOA and IT Governance Alignment	SOA and IT Governance Alignment
Structured Analysis & Design	Object Oriented Modeling	Component Based Development	Service Oriented Modeling	Service Oriented Modeling	Service Oriented Modeling	Grammar Oriented Modeling
Modules	Objects	Components	Services	Process Integration via Services	Process Integration via Services	Dynamic Application Assembly
Monolithic Architecture	Layered Architecture	Component Architecture	Emerging SOA	SOA	Grid Enabled SOA	Dynamically Re-Configurable Architecture
Platform Specific	Platform Specific	Platform Specific	Platform Specific	Platform Specific	Platform Neutral	Dynamic Sense & Respond
Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7

Figure 2-1 Service Integration Maturity Model

2.2 SIMM level 1: Silo

In SIMM level 1, separate *siloed* applications have no knowledge of one another. No data passes from one system to another, other than by being re-typed by users as shown in Figure 2-2.

No direct integration exists at this level. Therefore, no product is applicable at this level. Because both WebSphere InterChange Server and WebSphere Process Server are products used for integration, they are not relevant.

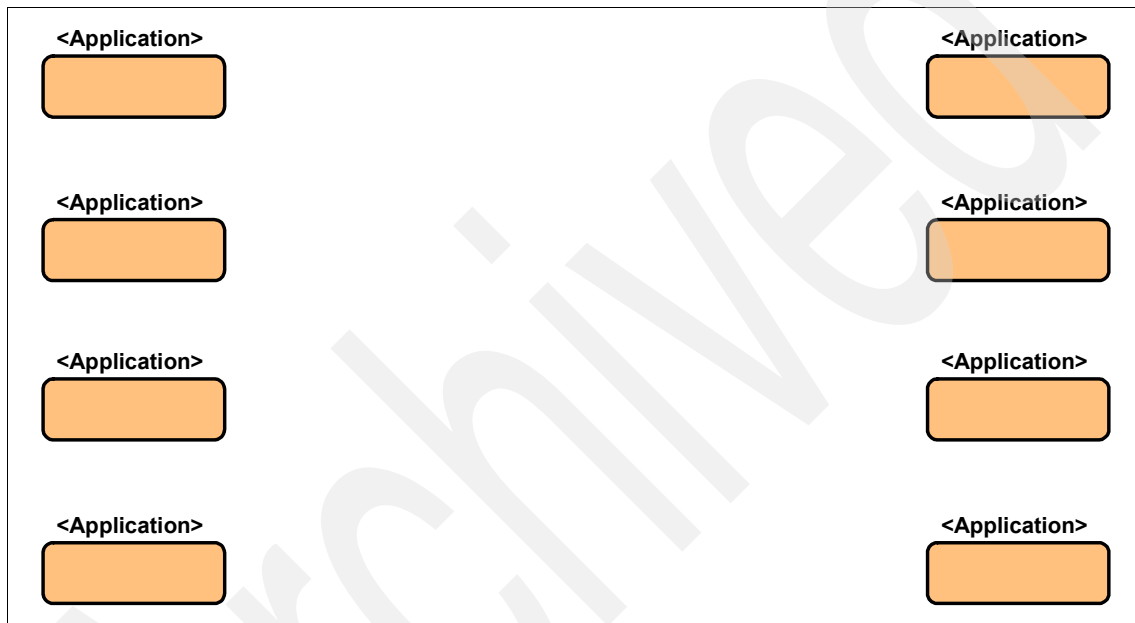


Figure 2-2 SIMM level 1: Siloed applications

2.3 SIMM level 2: Integrated (point-to-point)

Point-to-point integration directly connects one application to another. One application acts as the consumer, and the other application acts as the provider as shown in Figure 2-3. A provider in applications can also be a consumer; however for simplification, we label them as one or the other. Each additional consumer or provider requires a growing amount of *connector code* or *configuration*. The resultant number of connections increases geometrically, quickly becoming unmanageable.

Both WebSphere InterChange Server and WebSphere Process Server have connectors and, therefore, can provide the connectivity that is required for point-to-point integration.

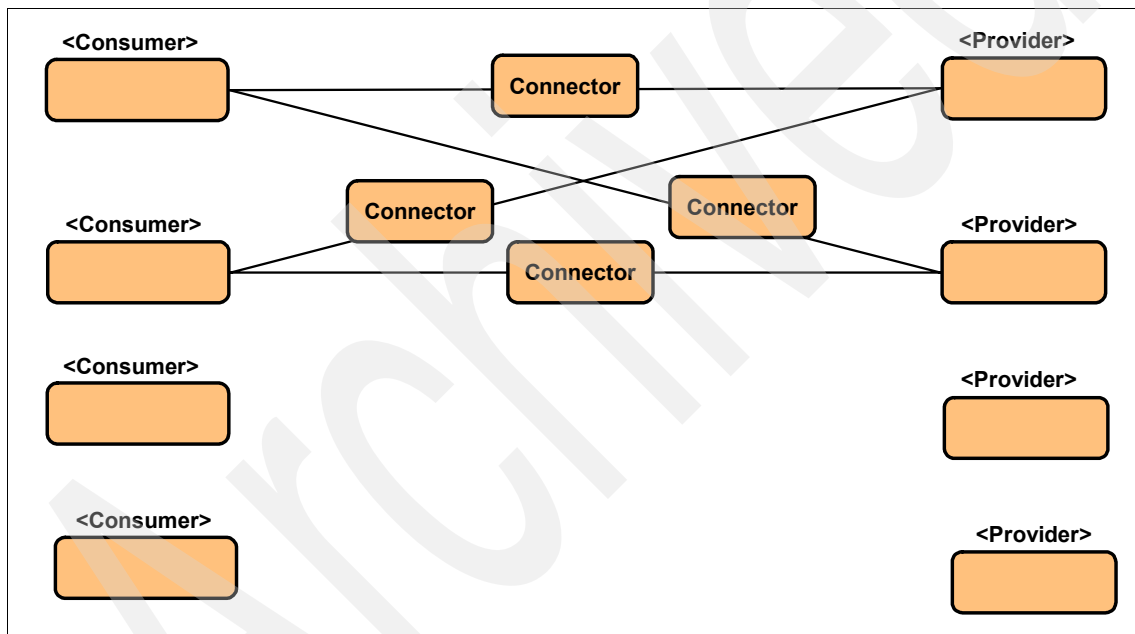


Figure 2-3 SIMM level 2: Point-to-point integration

2.4 SIMM level 3: Componentized (hub and spoke)

Figure 2-4 illustrates the flow of hub and spoke integration. Neither consumers nor providers can talk in a common protocol. Adapters or connectors are used to bridge the protocol or data format gap and allow communication to the hub. Adding a new consumer requires preparation of new maps and possibly the addition of a new connector or adapter. The hub treats consumers and providers similarly, providing routing, transformation, and integration logic to enable them to communicate with one another.

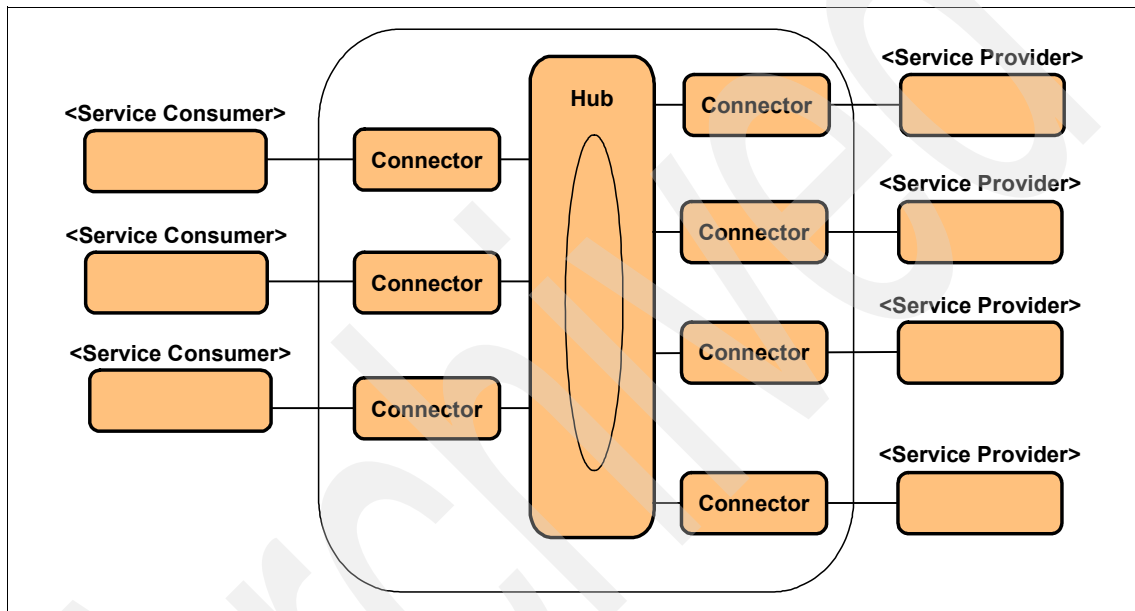


Figure 2-4 SIMM level 3: Hub and spoke integration

WebSphere InterChange Server with its adapters implements the hub and spoke principle. WebSphere Business Integration Adapters act as the connectors, and WebSphere InterChange Server acts as the hub. Most of WebSphere InterChange Server implementations are at this level.

WebSphere Process Server can also act as a hub and spoke. The WebSphere Adapters act as the connectors, and the WebSphere Process Server acts as the hub.

2.5 SIMM level 4: Services

Figure 2-5 illustrates the services level, which is the first level where organizations begin to benefit from an SOA. Here the enterprise service bus (ESB) gateway provides controlled access to enterprise services over standardized protocols, as well as operational and business metrics.

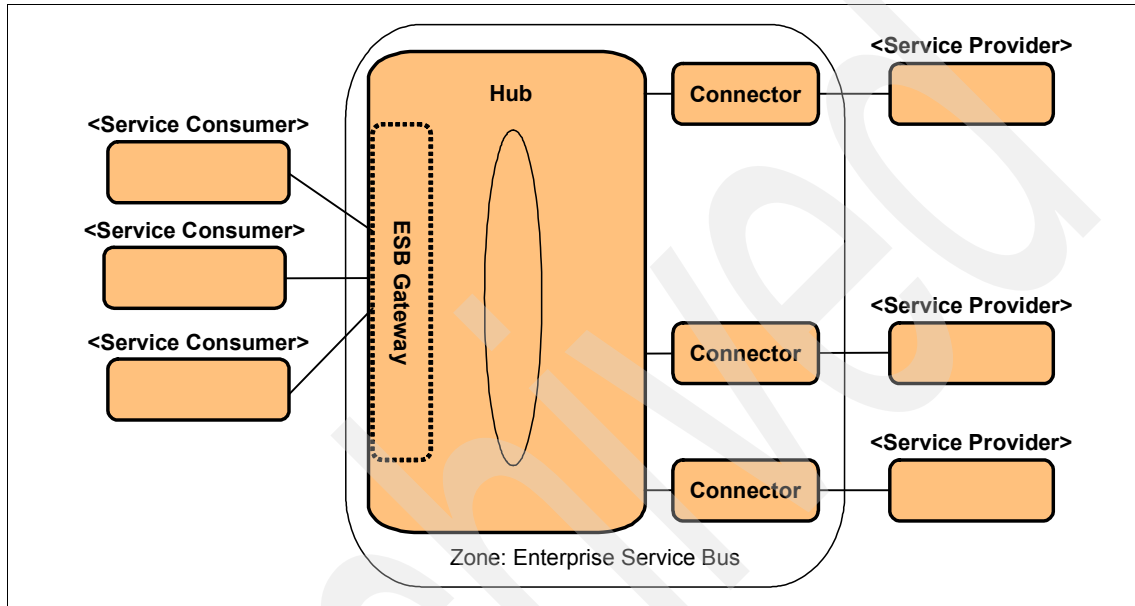


Figure 2-5 SIMM level 4: Services

Implementers of services can publish service interface definitions to a registry. Consumers can then retrieve the details from the registry for use at development time. The services themselves can either be atomic or aggregates of atomic services.

Consumers no longer need connectors to communicate to the hub due to standardized SOA protocols. New consumers can be added easily, because the contract with the hub is represented in a standard way.

In theory, as the SOA matures, more service providers make standardized protocols available. However, older systems might require deeper integration and, therefore, need connectors or adapters.

The hub continues to provide mediation functions, such as mapping, data formatting, and aggregation. Emphasis is placed on qualities of service (QoS), such as store-forward, retry, error handling, and so on, to ensure service level

agreement (SLA) compliance with the more unpredictable workload from potentially unknown consumers.

It is possible to build services with WebSphere InterChange Server, although it is generally less standard and less efficient than with WebSphere Process Server. The tools and the run time are less optimized for standards.

The approach to expose standard services with WebSphere InterChange Server relies on WebSphere Business Integration Adapters. Of these adapters, the Java Message Service (JMS) and Web services adapters expose standard services. WebSphere Business Integration Adapters use their own Java virtual machine (JVM™). When invoking a service through the WebSphere InterChange Server, at least three run times can be used, because adapters are required on the source side and the destination side, independent of the back-end application that provides much of the logic. Therefore, this type of service invocation does not perform as well as WebSphere Process Server, because it takes longer to expose the service at build time and to use it at run time.

Another possible approach with the WebSphere InterChange Server is to expose nonstandard services. This approach is possible by using the Server Access Interface to directly call services that are exposed as collaborations. With such an approach, the client code must be customized to use the nonstandard APIs of the Server Access Interface.

WebSphere Process Server is designed from the ground up to build and enable services. The services can be easily exposed as service bindings. WebSphere Process Server includes WebSphere Enterprise Service Bus (ESB), which excels at acting as an ESB gateway. Services can be mediated and act as both a logical and physical layer between service consumer and service implementation.

2.6 SIMM level 5: Composite services

Figure 2-6 on page 18 illustrates the flow of composite services, where a service calls another service. Often this is a choreography of services. WebSphere Process Server has a choreography engine, which is based on Business Process Execution Language (BPEL). This engine is the *business flow manager*. WebSphere Process Server also has a state machine representation called the *business state machine*, which uses the same engine. Both the business flow manager and the business state machine are meant for choreographing business processes. In the case where the business process is also exposed as a service, then it creates a composite service.

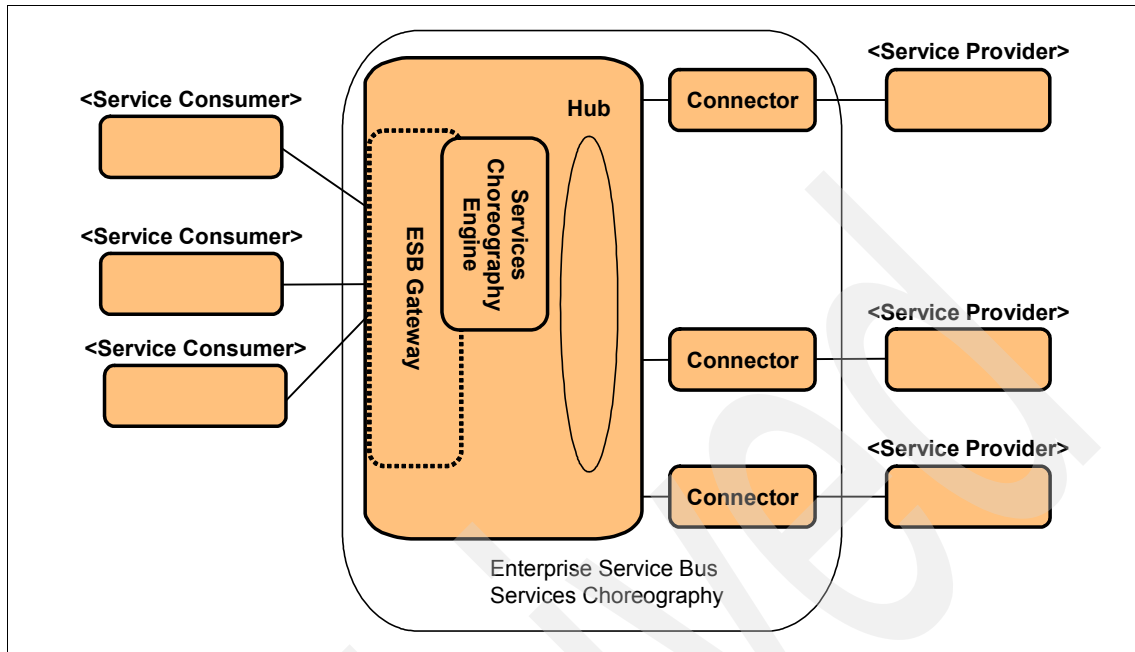


Figure 2-6 SIMM level 5: Composite services

A service registry is optional and recommended at this level; however, it is not mandatory. A Dynamic Assembler is also optional and recommended at this level. In a Dynamic Assembler, the atomic services can be assembled at run time based on business policies, roles, and channels. IBM WebSphere Business Services Fabric has a Dynamic Assembler component.

WebSphere InterChange Server presents possibilities to orchestrate services, based on the collaboration templates and collaboration groups; however, it lacks the support of standards and has no dedicated artifact to orchestrate standard services. Therefore, the WebSphere InterChange Server is less efficient than WebSphere Process Server from this perspective.

2.7 SIMM level 6: Virtualized services

Figure 2-7 illustrates the flow of virtualized services. At this level, the service consumer calls a virtual service provider, and the ESB calls the actual service provider. The calls require a Service Registry, such as WebSphere Registry and Repository. Service virtualization can be used, for example, to dynamically select the best service implementation for a specific consumer, depending on the current context, which allows respect of the SLA in a flexible way. WebSphere InterChange Server has no standard integration with a registry. Therefore, it is incapable of providing virtualized services.

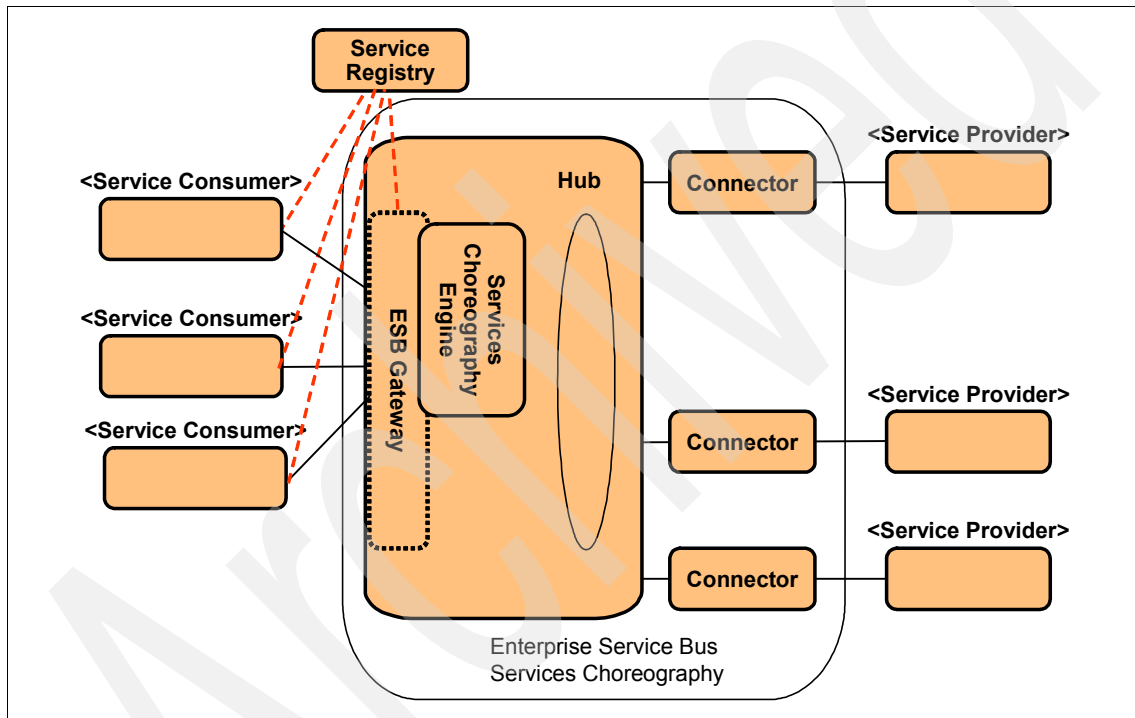


Figure 2-7 SIMM level 6: Virtualized services

2.8 SIMM level 7: Dynamically reconfigurable services

By using dynamically reconfigurable services as shown in Figure 2-8, a business user can dynamically call, change, and even choreograph the service providers that are used to implement a service. WebSphere Business Services Fabric can assist WebSphere Process Server with this task. With the Dynamic Assembler component, the business user can dynamically change the service implementation called by the consumer. The business user can also match the service consumer's requirements and the service provider's capabilities.

WebSphere InterChange Server does not work with WebSphere Business Services Fabric and has no equivalent component to the Dynamic Assembler. Therefore, WebSphere InterChange Server cannot be used for dynamically reconfigurable services.

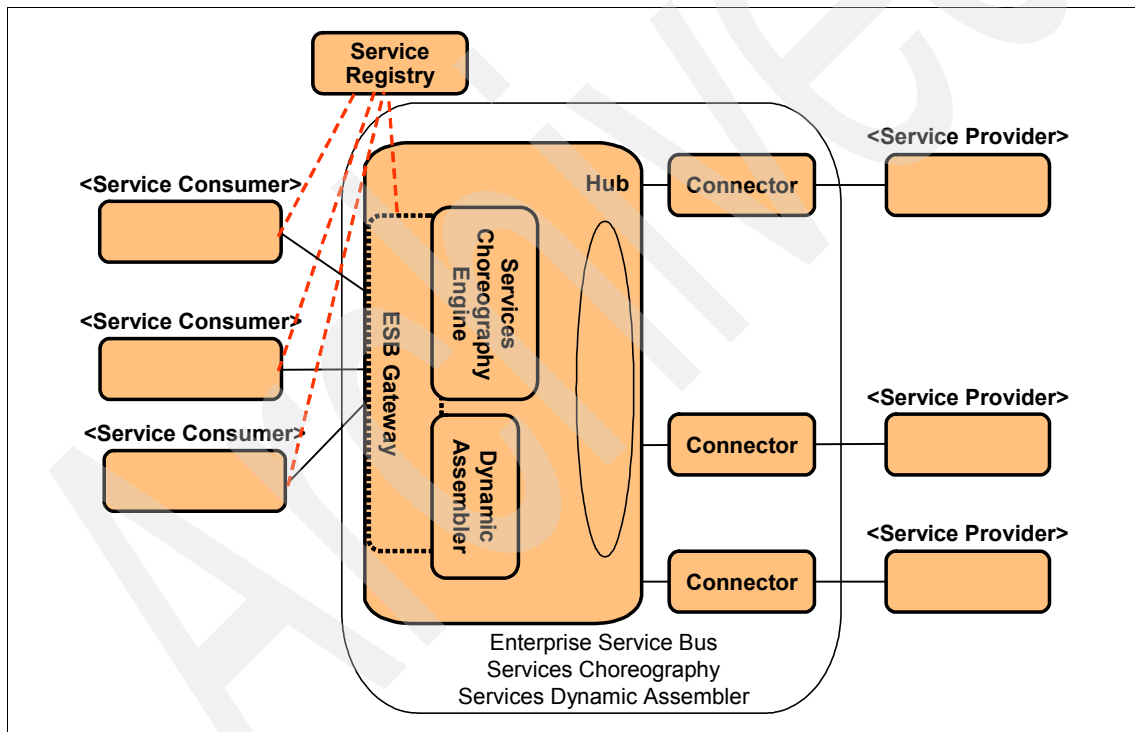


Figure 2-8 SIMM level 7: Dynamically reconfigurable services

2.9 Summary

WebSphere InterChange Server can be used in SIMM levels 2 to 5, although it is primarily used at SIMM level 3. WebSphere Process Server can be used at SIMM levels 2 to 7, although it excels at levels 4 and higher, especially when used in conjunction with other WebSphere products. Refer to Table 2-1 in which we compare the SIMM levels for WebSphere InterChange Server and WebSphere Process Server.

Table 2-1 SIMM level comparison for WebSphere InterChange Server and WebSphere Process Server

SIMM level	WebSphere InterChange Server	WebSphere Process Server
Silo	Not applicable	Not applicable
Integrated	Full support	Full support
Componentized	Full support	Full support
Services	Partial support	Full support
Composite services	Partial support	Full support
Virtualized services	No support	Full support with WebSphere Registry and Repository
Dynamically reconfigurable services	No support	Full support with Services Fabric

Usage patterns

To understand how to best migrate from WebSphere InterChange Server to WebSphere Process Server, you must understand how WebSphere InterChange Server is used. Primarily, WebSphere InterChange Server is used for integration, where WebSphere Process Server is used for integration, process choreography, and service exposition.

WebSphere InterChange Server has five major usage patterns, although the first three are the most widely used. By using patterns, you can categorize a problem, the context of the problem, and the solution. By using patterns, you can classify the major applications of a technology. After you know the classification, guidance is clearer on how you can reapply the usage pattern on a different technology.

In this chapter, we examine the usage patterns for WebSphere InterChange Server. We also provide general migration guidance about how to implement the usage patterns with WebSphere Process Server.

Important: We do not provide specific migration recommendations in this chapter, because each organization has unique migration requirements.

In this chapter, we include the following sections:

- ▶ 3.1, “Data synchronization pattern” on page 25
- ▶ 3.2, “Data synchronization with dependencies pattern” on page 26

- ▶ 3.3, “Data access pattern” on page 27
- ▶ 3.4, “Services pattern (accompanying a framework)” on page 29
- ▶ 3.5, “Business process or workflow pattern” on page 31

3.1 Data synchronization pattern

In the data synchronization pattern (Figure 3-1), a change in one Enterprise Information System (EIS) results in updating one or more EISs with the same logical data. In this pattern, WebSphere InterChange Server is used solely for integration to provide data synchronization between source and target applications. We cannot trivialize the collaborations as “moving data.” Here, the collaborations are used for transaction control, batch, routing, validation, transformation, enhancement of the adapter logic, and so on. For the most part, this pattern accounts for between 60% and 80% of all collaborations that were ever implemented on WebSphere InterChange Server.

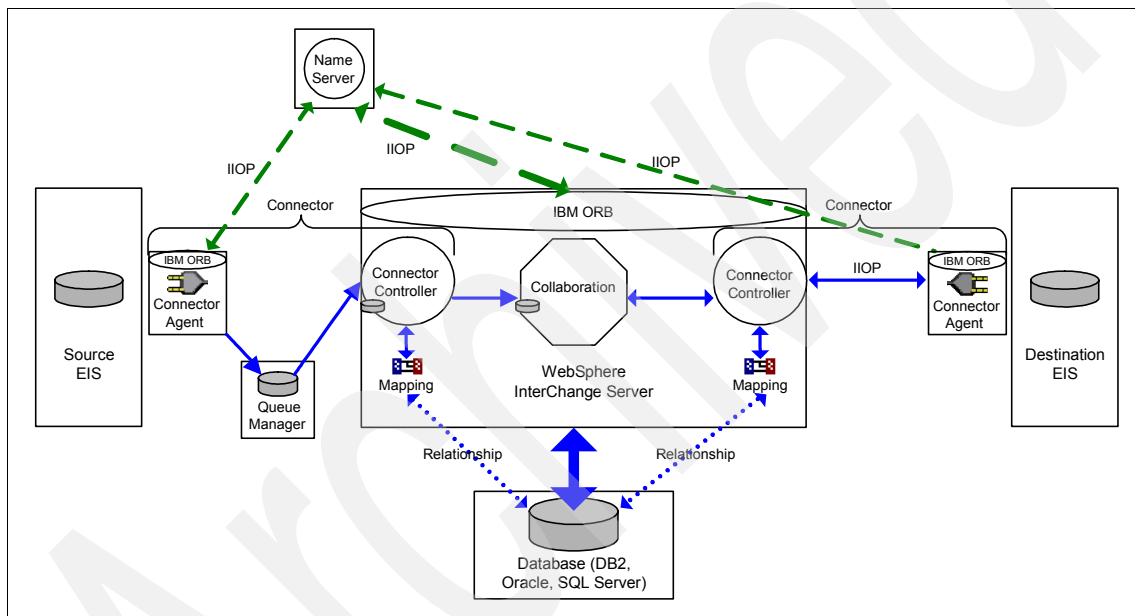


Figure 3-1 WebSphere InterChange Server data synchronization pattern

Migration recommendation

The usage of a data synchronization pattern depends on the specific details of the tasks that are being performed in the collaboration. Typically, most of the tasks are related to integration logic, not business logic. It is often better to manually redo these collaborations on WebSphere Process Server with the best applicable technology.

Most often, you use the following WebSphere Process Server components:

- The mediation flow components for simple and medium integration logic

- The Business Process Execution Language (BPEL) components if more complex integration logic is required (for example, if complex error handling with compensation is required)

If business logic represents a non-negligible part of the collaboration tasks, refer to 3.5, “Business process or workflow pattern” on page 31.

3.2 Data synchronization with dependencies pattern

Most of the WebSphere InterChange Server collaboration groups fall into the category of data synchronization with dependencies as illustrated in Figure 3-2. For example, in a sales order processing (SOP) collaboration group, you must synchronize order information. However, to send an order to an EIS, the EIS must have current customer and item information. If the EIS is not the system of record for customer and item, that synchronization triggers other synchronizations for those data dependencies.

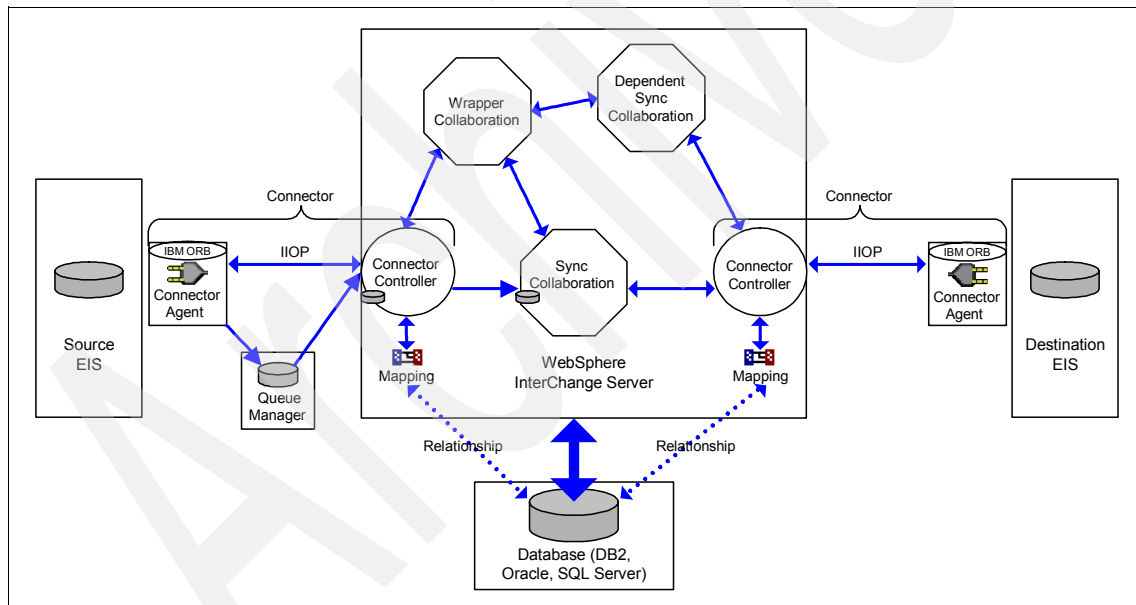


Figure 3-2 WebSphere InterChange Server data synchronization with dependencies pattern

In the case of an SOP, the following collaborations exist:

- SOP collaboration
- Customer wrapper collaboration
- Customer sync collaboration

- ▶ Item wrapper collaboration
- ▶ Item Sync collaboration
- ▶ Other wrapper and sync collaborations for each amount of dependent data

The wrapper collaborations are responsible for retrieving the data from the system of record and sending it to the synchronization collaboration, which sends it to the destination EIS.

Migration recommendation

The usage of the data synchronization with dependencies pattern depends on the specific details of the tasks that are being performed in the collaboration group. Generally, the collaboration group migration requires a dedicated master process to orchestrate the group interaction, for which BPEL is often the best choice for this implementation. Your integration layer, which includes the wrapper and synchronization collaborations, does not differ from the data synchronization pattern except that it responds to the control BPEL.

3.3 Data access pattern

The data access pattern, which is illustrated in Figure 3-3 on page 28, exposes data from the back-end EIS in a near real-time manner by using technology adapters on the front end (MQ/Java Message Service (JMS) or Web services) with primarily pass-through collaborations. The primary characteristics of this pattern are a fast response time and high availability. Most often this pattern is used by a Web front end that services a request from a person who is waiting for the response.

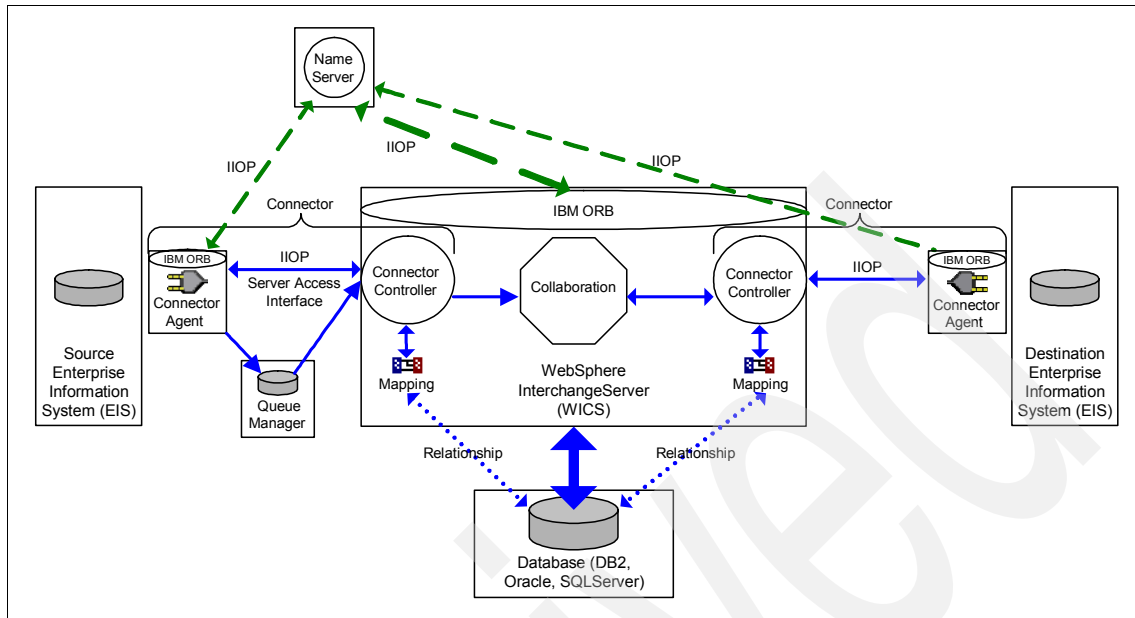


Figure 3-3 WebSphere InterChange Server data access pattern

Migration recommendation

For the data access pattern, a BPEL component is often not required, because the integration logic is simple (retrieving data from a back-end system typically). Therefore, the migration generally consists of implementing mediation flow components.

Because response time is paramount, replace the source technology WebSphere Business Integration Adapter with a native WebSphere Process Server Binding. By replacing this technology, you can reduce the interaction by two hops and open possibilities for native clustering and load balancing. Keeping a WebSphere Business Integration Adapter means using a separate Java virtual machine (JVM) from the process server, passing the data through JMS communication, and having extra serializations and deserializations between the server and the adapter. Also, WebSphere Business Integration Adapters cannot be easily used with a WebSphere Process Server cluster. However, it is fully supported with a native binding, such as the Web Services binding.

Note: The data synchronization, data synchronization with dependencies, and data access patterns represent the vast majority of WebSphere InterChange Server implementations that are in existence.

3.4 Services pattern (accompanying a framework)

Several implementations use collaborations as either coarse-grained or fine-grained services. Mostly these implementations are large implementations in which a lot is invested into building custom frameworks for logging, auditing, and notification. They normally do a good job at service definition; however, they are often customized due to collaboration limitations. Figure 3-4 on page 30 illustrates the flow of the services pattern.

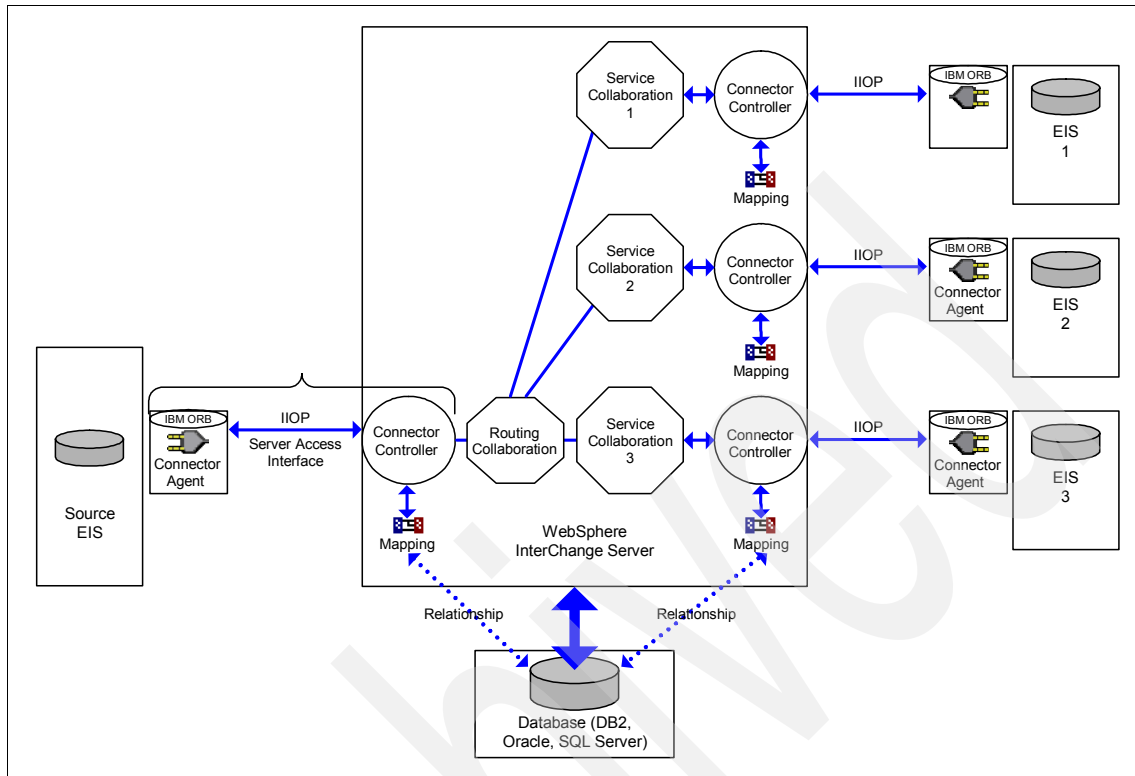


Figure 3-4 WebSphere InterChange Server services pattern

Migration recommendation

Any real framework based on the service collaboration pattern must be ported to the WebSphere Process Server platform. In general, a complete rewrite is necessary by using the best available components in WebSphere Process Server. Because WebSphere Process Server is based on the WebSphere Application Server platform, it offers a wide range of J2EE™ and Web services components that can be useful to re-implement the old framework in a more standard way. Standard migration tools cannot help with this process, because they are unable to migrate custom code.

3.5 Business process or workflow pattern

Certain WebSphere InterChange Server implementations focus on business logic to distinguish themselves from typical integration logic implementations. *Business logic* means that business users can take more control of the process. Generally this control requires advanced functionalities, such as human tasks, business rules, and workflow capability.

The WebSphere InterChange Server does not provide specific components to implement business logic, except the long-lived business process (LLBP) collaborations. With LLBPs, typically you can implement a workflow pattern based on a stateful process as illustrated in Figure 3-5. In such a process, the solution can communicate asynchronously with the back-end system, allowing long response times. The solution can also persist data that represents the process state. However, with the WebSphere InterChange Server, a comprehensive solution implementation generally requires custom code to store the state of the business process that is accessed, for example, across multiple collaborations.

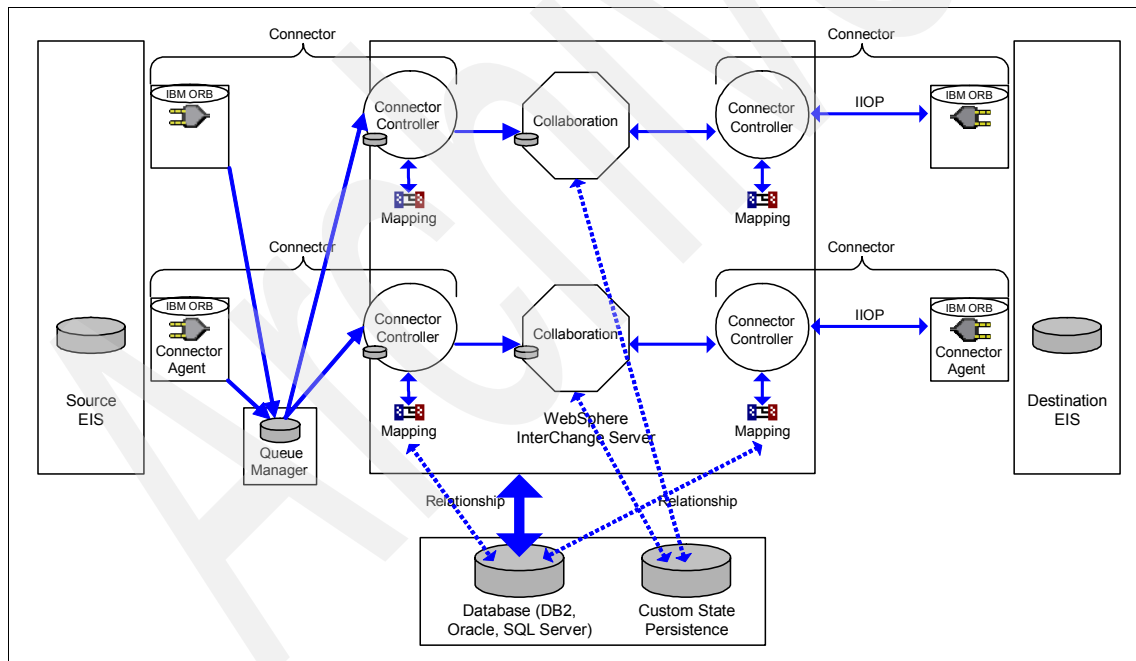


Figure 3-5 WebSphere InterChange Server workflow pattern

Migration recommendation

The WebSphere InterChange Server implementation is generally specific and customized for that pattern. Moreover, the standard migration tool cannot differentiate between process logic and integration logic that is contained in the collaboration code. In general, the best way to migrate this type of solution is to rewrite it from scratch in WebSphere Process Server. You can rewrite it by using long running BPEL component capabilities and additional new native components, such as business rules, human tasks, and business state machines.

Migrating long-lived processes might also require a parallel run, because the instance data cannot be directly migrated from WebSphere InterChange Server to WebSphere Process Server.

Migration planning

In this chapter, we discuss the considerations for migration planning. This chapter is written to help the project management team to plan the migration. The output is a recommended approach that can be driven through three major project phases:

- ▶ The assessment phase, which is the first phase, leads to an assessment of the impacts and risks, as well as high-level planning in accordance with the requirements.
- ▶ The preparation phase, which is the second phase, ensures that the necessary level of skills and experience is reached to secure the migration project.
- ▶ The implementation phase, which is the third phase, regroups the tasks that are needed to deliver the migration.

We include the following sections:

- ▶ 4.1, “Considerations for planning the migration” on page 34
- ▶ 4.2, “Assessment phase” on page 39
- ▶ 4.3, “Preparation phase” on page 64
- ▶ 4.4, “Implementation phase” on page 67

4.1 Considerations for planning the migration

In the following section, we explain why the migration must be driven generally through a dedicated project and the key items that you must consider.

4.1.1 Migration path evaluation

WebSphere InterChange Server projects are large scale projects that deal with an inherent complexity related to Enterprise Application Integration (EAI) concepts. As explained in Chapter 2, “Integration background” on page 9, WebSphere Process Server is far more than a new version of the WebSphere InterChange Server. WebSphere Process Server includes many new concepts related to service-oriented architecture (SOA) and Business Process Management (BPM).

WebSphere InterChange Server and WebSphere Process Server are different products. Even if most of the WebSphere InterChange Server concepts have been integrated into WebSphere Process Server, the runtime aspects differ:

- ▶ The WebSphere InterChange Server runtime data, such as long-lived business process (LLBP) instances and unresolved flows, cannot be migrated directly. There is no “in-place” migration for runtime data. Therefore, from this perspective, the supported path is to drain the WebSphere InterChange Server flows and to enable the WebSphere Process Server flows in parallel.
- ▶ Even if there is no runtime data, a given WebSphere InterChange Server environment cannot be migrated directly. There is no “in-place” migration for runtime components. Therefore, from this perspective, the supported path is to perform the following tasks:
 - Start from the development environment to migrate the build-time components.
 - Follow the usual project life cycle to build, test, and deploy the migrated code into the production environment.

Clearly, the migration involves more effort than a simple software upgrade. The effort can be analyzed this way:

- ▶ You might decide to stay at the same technical level and try to migrate “as is,” keeping the exact same functionalities. For such an approach, the return on investment (ROI) might be poor or even unacceptable.
- ▶ A second idea is to consider a global solution level. The migration is about moving from an EAI solution to an SOA or BPM solution. Then the migration

is seen as an opportunity to add value to the current business by using the new functionalities that are provided by WebSphere Process Server.

For all of these reasons, adopt a project approach for the migration. With such an approach, you can assess, optimize, budget, prepare, and deliver the migration properly and get as much ROI as possible.

4.1.2 Internal IT factors

In this section, we look at the major IT factors that are related to the implementation of a WebSphere InterChange Server migration project. Such factors are called *internal*, because they focus only on the following aspects:

- ▶ The current technical implementation scope of the WebSphere InterChange Server project
- ▶ The corresponding technical implementation capabilities of WebSphere Process Server

Important:

- ▶ The migration project must not focus only on internal IT factors.
- ▶ Consider carefully the external IT factors, such as the external applications roadmap and the global IT infrastructure roadmap.
- ▶ Because WebSphere Process Server offers many new features that are related to SOA and BPM, investigate how these features can enhance the current solution and add business value.

Beneficial internal IT factors

The following major internal IT factors are beneficial:

- ▶ WebSphere InterChange Server implementations that primarily use the data access pattern and expose data from the back-end Enterprise Information System (EIS) in a near real-time manner by using technology adapters on the front end with primarily pass-through collaborations

The client typically accesses the WebSphere InterChange Server through the Server Access Interface or technology adapters with synchronous capabilities, such as WebSphere MQ, Java Message Service (JMS), and Web services adapters.

- ▶ WebSphere InterChange Server implementations that use LLBP collaborations, providing long running process capabilities

- ▶ WebSphere InterChange Server implementations where XML/SOAP is the main data format to interact with the EIS and where there are only a few custom data handlers
- ▶ WebSphere InterChange Server implementations where Technology adapters such as Java Database Connectivity (JDBC) are used
- ▶ WebSphere InterChange Server implementations where maps are designed using mostly move, split, or join transformations and all custom coding is done in the Visual Java™ Snippet Editor
- ▶ Small WebSphere InterChange Server implementations with fewer than a dozen interfaces
- ▶ WebSphere InterChange Server implementations where there is scarce use of the system manager for administration (neither fat nor thin client), but rather use of instrumentation at the source to time Service Level Agreements (SLAs) and statistics
- ▶ Customers having a strong experience with WebSphere Application Server and J2EE, for example, running WebSphere Portal or WebSphere Application Server Network Deployment and also wanting to consolidate run times to simplify administration
- ▶ Customers having high availability (HA) requirements

Non-beneficial internal IT factors

The following internal IT factors are non-beneficial:

- ▶ Customers used to relying on the flow control capabilities of the WebSphere InterChange Server. For example, there are no equivalent capabilities in WebSphere Process Server with J2EE Connector Architecture (J2C) adapters regarding Store and Forward. Another example is the recovery concept, which is no longer available with WebSphere Process Server. In summary, the flow control mechanism is not the same in both products, because the engines are different.
- ▶ WebSphere InterChange Server implementations where many collaboration groups are used.
- ▶ WebSphere InterChange Server implementations where many custom Java libraries are used.
- ▶ WebSphere InterChange Server implementations where batch processing is used.
- ▶ WebSphere InterChange Server implementations where custom frameworks are heavily used, for example, built-in frameworks for error handling, auditing, and monitoring.

- ▶ WebSphere InterChange Server implementations where extensive custom coding was done in collaborations and maps.
- ▶ WebSphere InterChange Server implementations where the collaborations and maps are already the result of a migration (coming from older versions of the WebSphere InterChange Server product).
- ▶ WebSphere InterChange Server projects where administration teams heavily rely on the system manager (fat and thin clients), for example, to follow up flow statistics, to monitor the health of the components, and to start and stop individual fine-grained components.
- ▶ Clients having no experience with WebSphere Application Server and J2EE.

4.1.3 Recommendations for defining the migration path

In this section, we summarize the guidelines to define the migration path. These guidelines lead to a project approach that consists of three main phases that are described in 4.2, “Assessment phase” on page 39, 4.3, “Preparation phase” on page 64, and 4.4, “Implementation phase” on page 67.

Know your target

Generally speaking, because the intent is to move from a known source environment (WebSphere InterChange Server) to a new target environment (WebSphere Process Server), know the target before you start the migration implementation. Use a project approach for this purpose:

- ▶ Start using WebSphere Process Server for new development as soon as possible. Because you need comprehensive experience with the new platform, from development to production system, a real project is ideal for this purpose.
- ▶ Choose a new project to begin acquiring and building skills on the SOA or BPM platform. A pilot project is generally the best compromise to reach this goal with reduced risks.
- ▶ Start building WebSphere Process Server skills *now*, along with WebSphere Application Server skills if needed. By running a pilot project, with a well-defined and tailored technical scope, you can select the points of interest and build relevant key skills.
- ▶ Avoid starting the WebSphere Process Server experience with a pure migration project. By using a brand new pilot project, you can identify new and interesting capabilities that will bring value to your organization and to your business. A first pilot project also helps you get used to the new platform, before you focus on the migration implementation.

In summary, go through a full life cycle from development to going live with WebSphere Process Server so that you get a proven production infrastructure before migrating. This approach smooths out the learning curve and optimizes the migration effort.

Get prepared

To get prepared, sooner is often better, but the tasks must also be executed in the correct order:

- ▶ Start planning your migration now.
That does not mean that you start the migration implementation now.
- ▶ Execute the migration path through a dedicated migration project. Take the migration through the main phases as presented in the following sections:
 - 4.2, “Assessment phase” on page 39
 - 4.3, “Preparation phase” on page 64
 - 4.4, “Implementation phase” on page 67
- ▶ Invest in the assessment and preparation phases, because they help to secure the migration implementation phase.

Optimize your migration

Consider the migration as an opportunity and not a constraint:

- ▶ Identify your business objective and the value of the modified solution:
 - Enhance the solution with capabilities that previously were not possible.
 - Use the new functions.
- ▶ Generally, avoid an “all or nothing” approach:
 - Unless your WebSphere InterChange Server project is small, do not migrate everything at one time. Doing so is generally considered too risky.
 - Migrate by viable technical stages, for example, by first migrating the server and the supported adapters, and then the unsupported adapters. You can also rely on interoperability capabilities between WebSphere InterChange Server and WebSphere Process Server.
 - Migrate by functional stages. For example, first migrate a pilot project, then start migrating noncritical flows, and then progressively migrate functional and coherent sets of flows.
 - Compensate parallel infrastructure costs by implementing new functionalities that bring added value, such as Human Tasks, for example.
- ▶ As a cost-effective measure, combine migration with the next application or infrastructure change the next time that you need to upgrade either your applications or the infrastructure.

4.2 Assessment phase

The migration project must begin with an assessment phase. In general, because the intent is to move from a known source environment (WebSphere InterChange Server) to a new target environment (WebSphere Process Server), follow this path:

1. Provide a precise assessment of the existing WebSphere InterChange Server environment. That is, define where you stand.
2. Proceed to requirements gathering and analysis. Define where you are going.
3. Drive impact analysis and risk assessment. Define the right path.

4.2.1 Current environment

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

Current IT environment context

Before you address the project scope of WebSphere InterChange Server, collect input that is related to the global IT environment. The following examples are related to this task:

- ▶ What are the identified dependencies between the WebSphere InterChange Server project and other domains?
What is the project roadmap for the connected applications?
- ▶ What is the SOA strategy and its maturity?
Is there an ongoing task about providing reusable services?
- ▶ What governance model is currently in place?
How are the projects and the common assets funded?

Current WebSphere InterChange Server project context

The following examples are input to collect in order to assess the current WebSphere InterChange Server solution:

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

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

4.2.2 Requirements gathering

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

It is important to assess whether you intend to migrate the current solution “as is” or to add value to the current solution. You must determine if the migration is driven by constraint, opportunity, or both, because that determination influences the ROI.

4.2.3 Impact analysis

In addition to migrating a product, you are also moving from an Enterprise Application Integration solution to an SOA solution. Therefore, you must tackle the project with a comprehensive vision. Each project has its unique set of specific constraints and priorities. In this section, we provide a list for you to use as input to build your own impact assessment.

We discuss the following topics:

- ▶ “Organizational aspects” on page 41
- ▶ “Project delivery” on page 42
- ▶ “Methodology” on page 45
- ▶ “Architecture and design” on page 47
- ▶ “Development” on page 47
- ▶ “Testing” on page 49
- ▶ “Deployment” on page 50
- ▶ “Run time, quality of service, and security” on page 54
- ▶ “Operations” on page 58

Organizational aspects

In this section, we identify possible relationships between the migration project and organizational aspects.

SOA and IT governance

SOA governance is an extension of IT governance that focuses on the life cycle of services and composite applications in an organization's SOA. For an introduction to SOA governance, refer to the following Web address:

<http://www-306.ibm.com/software/solutions/soa/gov/>

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

BPM enabled by SOA, business, and IT alignment

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

<ftp://ftp.software.ibm.com/software/bigplays/AP-BPMSOA-BTW-00.pdf>

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

Center of Excellence

For large-scale Integration Solution implementations, it is common to have a Center of Excellence (CoE) in place. The same concept applies to SOA. The migration project can either use or be an opportunity to initiate such an organizational structure.

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

A CoE has the following goals:

- ▶ Communicate the architectural framework, best practices, assets, patterns, and methods.
- ▶ Conduct independent architecture reviews.
- ▶ Provide architecture vitality.
Continuously assess and refine the architecture framework and support assets based on internal and external influences.
- ▶ Provide project support (direct project assistance to drive the architecture).

- ▶ Provide skills transfer and early proof of concept.
Identify skill gaps, create development road maps, and drive the use of new technologies.
- ▶ Promote asset adoption.
Manage component, pattern, and data reuse processes to reduce project risk and accelerate delivery.
- ▶ Provide best practice policy and procedures.
Provide expert resources to accelerate the delivery of critical architecture practices.
- ▶ Support production.
Enable infrastructure teams to execute on build and deploy, tuning, and metrics reporting.

Project delivery

In the following sections, we present topics that are related to project delivery and that might be impacted by the migration.

Project costs and funding

Because WebSphere Process Server offers new SOA concepts, its deployment during the migration project can impact the usual project costs and funding model:

- ▶ Hardware
For example, WebSphere Process Server is supported on new platforms. Therefore, it can use the new hardware. The migration project generally implies a transition phase that can increase the license and hardware costs in the case of a parallel run.
- ▶ Data management
For example, because the internal data model is richer in WebSphere Process Server than in WebSphere InterChange Server, data management can be impacted by archiving and warehousing costs to maintain the long running instances of business processes.
- ▶ License procurement and maintenance
Investigate what you can do with the Extended Entitlement, which is described at the following Web address:

<http://www.ibm.com/software/integration/wps/library/entitlement.html>

► Funding

Because WebSphere Process Server participates in the SOA vision, it brings new concepts around project funding. A typical example is the way in which SOA intends to identify and reference services to ensure that they are shared by multiple entities or projects in an optimal way:

- Capacity to reuse an existing service
- Capacity for the service to scale up properly with growing consumers

This act of sharing can impact the current funding model by raising the following types of questions:

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

► The migration project

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

- The parallel infrastructure cost can be mitigated through the enablement of a means of interoperability between WebSphere InterChange Server and WebSphere Process Server. For example, WebSphere Process Server can reuse existing WebSphere InterChange Server developments through a Web service call, which spares the corresponding new development costs. Another example is that WebSphere Process Server and WebSphere InterChange Server can use the same existing relationships, which spares certain database-related costs.
- Use extended license entitlements for migration. Installed base licensees with an active subscription can activate entitlement to the WebSphere Process Server and receive support for both old and new products during the migration.
- Identify new functions that add value to the migrated solution.

Project resources and skills enablement

The people involved in the migration can have different skill sets:

- Typically, WebSphere InterChange Server users have a good understanding of the following concepts:
 - EAI concepts with a hub and spoke architecture
 - Java and stand-alone server management, because WebSphere InterChange Server is a proprietary product based on a Java 2 Platform, Standard Edition (J2SE™)-based single server

- ▶ WebSphere Process Server users typically have a background that includes the following concepts:
 - J2EE, SOA, and BPM
 - Service Component Architecture (SCA), Business Process Execution Language (BPEL), and natively clustered environments, because WebSphere Process Server is based on WebSphere Application Server Network Deployment and new open standards

Many WebSphere InterChange Server concepts, such as adapters, business objects, maps, and relationships, are used in WebSphere Process Server. Therefore, consider that WebSphere InterChange Server is a subset of WebSphere Process Server. The details of the products, including the runtime architecture, component granularity, quality of service (QoS), and so on, are quite different. Also, WebSphere Process Server addresses concepts, such as SCA and BPEL.

Therefore, the learning curve is expected to be relatively steep when implementing the first WebSphere Process Server project, regardless of whether the team has a background in WebSphere InterChange Server or J2EE/Web services. Skills enablement must not be underestimated, because this learning curve applies to both the development and operations teams.

Project timeline and product roadmap

Among the factors that impact the project timeline is the *product roadmap*. During the transition phase of going from WebSphere InterChange Server to WebSphere Process Server, the migration project must manage both product road maps:

- ▶ For the WebSphere InterChange Server, the main factor is the “end of support” date:
 - The end of support has not yet been announced for WebSphere InterChange Server 4.2.2 and 4.3 (or WebSphere Business Integration Server). IBM intends to support these products until at least September 2013, with extended support availability planned until at least April 2016.
 - To the extent that these products include third-party prerequisite software, such as WebSphere Business Integration Adapters for enterprise applications, continued service for such software during this period can be determined on a case-by-case basis and in the sole discretion of IBM.
- ▶ For WebSphere Process Server, the main factor is the way that the new releases of the product fill in possible functional gaps, easing technical migration, for example, and add new functionalities, providing business differentiators, for example.

Service level agreement

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

An SLA might be more complex with an SOA solution than with an EAI solution, because WebSphere Process Server offers a richer set of integration patterns. For example, WebSphere InterChange Server is rarely used as a Web services gateway, where WebSphere Process Server does it well within a cluster deployment. Therefore, performances and QoS might have to be measured differently depending on the implementation of the new integration patterns. Also, combined or aggregated SLAs might appear in the context of a global SOA solution and are generally quite complex to handle.

For all of these reasons, the way in which SLAs are defined might be impacted by the migration.

More information: We discuss the technical aspects of the SLA, such as QoS and performance, in “Run time, quality of service, and security” on page 54.

Methodology

The word *methodology* can have various meanings depending on the context. In this section, we use it in a generic manner.

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

The following examples are about methodologies that might be needed to support the migration implementation, but also about existing methodologies that might be impacted by the migration:

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

- ▶ A methodology for regression testing might have to be defined to secure the migrations phase. If a methodology is pre-existing, it might be impacted by the migration. Refer to “Testing” on page 49 for more details.
- ▶ From a system life-cycle perspective, a *Transition for Operations* methodology might have to be defined to secure skill building for this team. Refer to “Operations” on page 58 for more details.

The following considerations relate to the WebSphere Process Server architecture and open standards:

- ▶ The WebSphere InterChange Server has a simple and efficient architectural model:
 - The hub-and-spoke model has a canonical format (application-specific business object (ASBO)-generic business object (GBO) model).
 - Connectivity is done by using adapters (protocol and data translation).
 - Structured data transformation occurs in maps.
 - Integration logic is done by using collaborations.
 - With such a constrained model, it is almost impossible to get an incorrect implementation. The architectural model is so simple that you cannot deviate from it. WebSphere InterChange Server implements a single general integration pattern (hub and spokes).
- ▶ In comparison, WebSphere Process Server is more open, because it already relies on rich and open standards, such as J2EE, Web services, Service Data Objects (SDO), and the SCA. WebSphere Process Server also implements BPM concepts to extend the architectural possibilities around the integration field.
- ▶ Openness can have a paradoxical side effect. It might raise the risk of mistakes and misuse. How do you mitigate this risk?

You mitigate the risk by applying proven design methodologies and best practices. Today, the concept of a design pattern is widely spread over the IT community. The following examples can be of great interest for the migration:

- Object design patterns
- J2EE and SOA design patterns
- BPM and workflow design patterns as described at the following address: <http://www.workflowpatterns.com>

For more information about integration design patterns, refer to Chapter 3, “Usage patterns” on page 23. In this book, we encourage you to use these patterns as a general design approach for the architectural aspects of the migration project.

The following considerations are related to project life-cycle management:

- ▶ From the general project life-cycle perspective, the major project phases are the same as when implementing an SOA or an EAI project.
- ▶ SOA can bring new concepts, such as an extensive focus on external environment dependencies at the enterprise level. For instance, activities that are related to common services identification and exposure might need to be added in the current project activities.
- ▶ If you look for well-proven methodologies and tools related to SOA project life-cycle management, you might investigate Service-oriented Modeling and Architecture (SOMA) and Rational® Unified Process (RUP) for SOA. The IBM SOMA methodology helps address SOA implementation through dedicated phases, such as services identification, specification, and realization. It uses a holistic approach that combines top-down and bottom-up analysis.

Architecture and design

Design patterns provide a way to describe the current WebSphere InterChange Server solution and the target WebSphere Process Server solution from the architecture design perspective. Such a common architecture language is useful to categorize the concepts that are in both EAI and SOA solutions and to define a migration path between the product specific implementations. Refer to Chapter 3, “Usage patterns” on page 23, for relevant information.

Development

In this section, we discuss points that are related to the development phase of the usual project life cycle. The WebSphere InterChange Server development team is impacted by the migration. The WebSphere Process Server development tool (WebSphere Integration Developer) is a new tool that is completely independent from the WebSphere InterChange Server development tool (WebSphere InterChange Server toolkit).

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

WebSphere Integration Developer is built on the Rational Software Development Platform. Rational Software Development Platform is based on Eclipse 3 technology. Each IBM product that is built on this platform can coexist and share plug-ins with other Rational Software Development Platform-based products.

Supported platforms for build time

Investigate precisely the supported platforms for the build time (WebSphere Integration Developer) in terms of hardware and operating system. You can obtain this information in the product's support pages and in the WebSphere Integration Developer information center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

Note: The main part of the standard migration tool is in WebSphere Integration Developer. During the transition phase, you might have to use the WebSphere InterChange Server and WebSphere Process Server development tools in parallel. Therefore, you must also verify the feasibility of their coexistence on a common platform, such as the compatibility of products to run on a single machine, the required memory, and so on.

Software upgrade process for build time

Fix packs are provided regularly to fix programming errors and provide enhancements through minor product upgrades. Because WebSphere Integration Developer differs from the WebSphere InterChange Server and Adapters toolkit, investigate the supported software upgrade process for WebSphere Integration Developer. You can find such information on the support pages of the product.

New development concepts and skills building

WebSphere Process Server provides a new programming model that is based on concepts, such as SCA, BPEL, Mediation Flow Component, and business rules.

Developing for WebSphere Process Server in the most efficient way also implies that you must investigate more advanced concepts, such as the following examples:

- ▶ Versioning and dynamicity
- ▶ Error handling
- ▶ Quality of service

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

Constraints of migration tools: The development team must investigate the constraints of the standard migration tools. For example, the migration tools can automatically migrate maps and collaboration templates into WebSphere Process Server artifacts. However, this use requires that you follow best practices, which can impact the original WebSphere InterChange Server artifacts. Refer to Chapter 13, “Best practices” on page 263, for input to this task.

Team development

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

Verify the supported repositories and their versions to anticipate any impact if a Version Control System (VCS) is already used with the WebSphere InterChange Server toolset.

Further information

For further information about the development tasks, refer to the product documentation and the WebSphere Integration Developer information center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.nav.doc/topics/welcome.html>

Testing

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

Unit testing

Unit testing with WebSphere Integration Developer is far richer than with the WebSphere InterChange Server toolset. It is now possible to create test suites that contain variable data and to fully automate testing and results analysis. These capabilities definitely add value by reducing the testing time and improving the test quality. We therefore advise you to investigate such new capabilities for WebSphere Process Server developments.

Regression testing

When possible, to simplify migration, a best practice is to have regression test tools that are located on the edges, external to the WebSphere InterChange Server solution, or in the applications, and not to use WebSphere InterChange

Server APIs or tools. This way, the regression test tools are less impacted or even not impacted by the migration.

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

- ▶ QoS (global resilience of the solution, flow control and throttling capabilities, for example)
- ▶ The Operations team's ability to monitor and administer the new platform

Performance testing and benchmarking

Because the WebSphere InterChange Server and WebSphere Process Server run times and deployment configurations differ, run dedicated performance tests as early as possible to assess the capacity of the migrated solution.

Deployment

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

Packaging

Investigate the capabilities of the WebSphere Process Server platform regarding the packaging process by using a GUI and automation tools. WebSphere InterChange Server and WebSphere Process Server differ in this perspective, which impacts the following areas:

- ▶ Generation of software packages to be deployed
- ▶ Inter-environment adaptation procedures
- ▶ Procedures for deployment to the run time environment

Component architecture of the solution

The component architecture of the solution differs between WebSphere Process Server and WebSphere InterChange Server. WebSphere InterChange Server needs at least a separate Object Request Broker (ORB) process, a database, and generally a WebSphere MQ queue manager. WebSphere Process Server is based on WebSphere Application Server architecture and requires a database with a richer set of tables than the WebSphere InterChange Server database.

The solution can also interact with other third-party products, such as the following examples:

- ▶ A Lightweight Directory Access Protocol (LDAP) directory, such as IBM Tivoli® Directory Server
- ▶ A security system, such as Tivoli Access Manager
- ▶ Monitoring tools, such as Tivoli monitoring tools

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

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

Topology for high availability and scalability

The deployment topology of WebSphere InterChange Server differs from WebSphere Process Server. Therefore, your current environment must be assessed from an availability and scalability point of view. In addition, the corresponding deployment topology of WebSphere Process Server must be selected to ensure a desired level of business continuity. The overall availability of your environment is closely related to costs, and your decision will trade off between cost and availability factors. The topology has the following levels of availability:

- ▶ Basic systems
There is protection of backups only, not of data and services.
- ▶ Redundant data
Disk redundancy or disk mirroring is used to protect against data loss.
- ▶ Component failover
Hardware and software components are doubled, which means that there are at least two instances of the same component in the system. In case of the failure of one instance, the second instance takes over the load. Differentiate *vertical clustering*, such as running two or more instances of WebSphere Process Server on one hardware node, from *horizontal clustering*, such as running two or more hardware nodes with WebSphere Process Server.
- ▶ System failover
A standby or backup system is used to take over for the primary system if the primary system fails. Usually, clustering software monitors the health of the system and automatically fails over the load to a secondary system.
- ▶ Disaster recovery
Disaster recovery is similar to a system failover, but the primary system and the secondary system are at separate geographical locations (primary site and backup site). The secondary system takes over the complete load in case of primary site loss.

The High Availability (HA) of the WebSphere InterChange Server system, in general, is performed by using a hardware clustering solution. For correct failover and recovery behavior, the broker depends on the following two items:

- ▶ Message resilience and message availability provided by the underlying transport layer
- ▶ Event and transaction information provided by a highly available database

For WebSphere InterChange Server, the smallest unit of failover is a server, ORB, and the WebSphere InterChange Server queue manager. To make the WebSphere InterChange Server highly available, the ORB must be included as part of the WebSphere InterChange Server HA cluster resource group, because it is not possible to start WebSphere InterChange Server without restarting the ORB.

For more in-depth information about how to make the WebSphere InterChange Server highly available by using hardware clustering software and for details about the resource groups and grouping of components, refer to *Highly Available WebSphere Business Integration Solutions*, SG24-6328.

The High Availability of the WebSphere Process Server involves components other than the WebSphere InterChange Server. At a high level, every WebSphere Process Server environment can be broken down into three fundamental layers as shown in Figure 4-1.

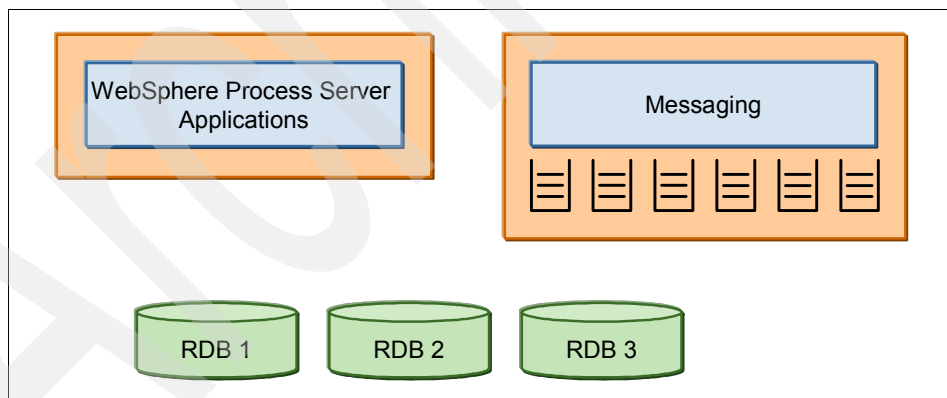


Figure 4-1 Components to be clustered in WebSphere Process Server

We explain each layer:

- ▶ WebSphere Process Server applications

A clustering of WebSphere Process Server applications does not significantly differ from clustering a plain J2EE application in a 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. Part of the messaging infrastructure must use WebSphere Application Server Service Integration Bus (SIBus) and the WebSphere Default Messaging JMS provider. Because the SIBus requires a WebSphere Application Server to run, the messaging infrastructure can also be clustered by using the WebSphere clustering techniques.

In conclusion, you must consider a number of aspects when selecting the topology to adopt. For more information about key production topologies of WebSphere Process Server clustering, see the following references:

- *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413
- *WebSphere Business Process Management V6.1.2 Production Topologies*, SG24-7665

Capacity planning

Capacity planning is dependent on the chosen deployment topology. Because WebSphere InterChange Server and WebSphere Process Server run times differ, it is generally not a straightforward exercise to determine the capacity of the migrated solution. IBM can provide estimates based on Techline tools and benchmark studies. However, be sure to run performance tests on-site as early as possible to get realistic data.

Product coexistence

WebSphere Process Server and WebSphere InterChange Server environments must coexist under the following circumstances:

- Both environments are up and running during the piloting phase.
- Applications are migrated in several stages.
- A period of coexistence must be achieved in order to finish LLBP collaborations or to resubmit unresolved flow on WebSphere InterChange Server while starting and working on migrated processes in the new WebSphere Process Server environment.

Product coexistence has a potential effect on costs, because it implies parallel infrastructure, which affects hardware consumption and license consumption. However, parallel infrastructure reduces risks. You can reduce costs in the following ways:

- ▶ WebSphere Process Server and WebSphere InterChange Server can call each other through Web services or JMS. This interoperability between WebSphere InterChange Server and WebSphere Process Server provides flexibility in how existing WebSphere InterChange Server solutions are reused and when they are migrated to WebSphere Process Server. A side effect is cost reduction when the WebSphere Process Server calls an existing service hosted in WebSphere InterChange Server, sparing the effort and cost of new developments.
- ▶ WebSphere Process Server can use the relationship tables that are created and used by the WebSphere InterChange Server environment, saving on investment by sharing common data that does not need migration.
- ▶ The extended entitlement can reduce costs that are related to licenses.

Therefore, investigate these possibilities carefully to optimize the transition phase.

Run time, quality of service, and security

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

Installed run time

The installed run time of WebSphere InterChange Server does not migrate directly to the run time of WebSphere Process Server:

- ▶ There is no standard migration path for instance data, such as unresolved flow events and LLBPs. The only supported approach is to let them terminate in the WebSphere InterChange Server environment.
- ▶ There is no standard migration path for system configuration. The only supported way is to redo the configuration steps for WebSphere Process Server. Redo, for example, server tuning parameters, server security settings, and system monitoring.
- ▶ You must use the usual project life cycle, from development to production, with installation, setting, and tuning of the new product to set up your new production system with migrated applications.

The relationship tables are an exception to this rule, because WebSphere Process Server can reuse them.

Supported platforms for run time

Investigate precisely the supported platforms for the WebSphere Process Server run time in terms of hardware and the operating system. You can find that information in the product's support pages and in the WebSphere Process Server information center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

Note: The same question arises for the third-party products, such as databases and LDAP for example. The answer is also in the support pages. During the transition phase, WebSphere InterChange Server and WebSphere Process Server run times might have to be used in parallel. If so, verify the feasibility of their coexistence on a common platform (compatibility to run on a single machine, required memory, and so on).

Software upgrade process for run time

Fix packs are provided regularly to fix bugs and provide enhancements through minor product upgrades. Because WebSphere Process Server is different from WebSphere InterChange Server, investigate the supported software upgrade process for WebSphere Process Server. You can find this information on the WebSphere Process Server support pages.

Quality of service

Because WebSphere InterChange Server and WebSphere Process Server have different run times, the QoS that they provide is not always identical. It can effect runtime behavior. Because we cannot systematically document every difference, we mention the typical ones:

- ▶ Store and forward

The WebSphere InterChange Server can detect whether a WebSphere Business Integration Adapter agent is down and then store the event until the agent comes up. There is no equivalent capability in the J2EE Connector architecture (J2C) layer.

- ▶ Flow control

As a reminder, in the WebSphere InterChange Server, you can use flow control, which is a configurable service, to manage the flow of connector and collaboration object queues. You can either configure the parameters for flow control system-wide or on individual components. There is no equivalent capability in WebSphere Process Server.

- ▶ Event sequencing

WebSphere InterChange Server and WebSphere Process Server have similar capabilities in regard to event sequencing. However, because WebSphere Process Server brings new support for native clustering, the

support of event sequencing can be affected, depending on the cluster topology that is deployed.

- ▶ Error handling

There is no native capability in WebSphere InterChange Server to retry a service call in case of failure. It is now a native capability in WebSphere Process Server V6.2 through Mediation Flow Component support (retry settings in the mediation flow).

Security

To secure critical business data, WebSphere InterChange Server includes the following security features:

- ▶ End-to-end privacy, also labeled *end-to-end security*, secures data as it flows from a source adapter process, through the WebSphere InterChange Server, to the destination adapter process.

End-to-end privacy uses asymmetric and dynamic security in addressing authentication, integrity, and privacy.

- ▶ Role-based access control restricts access to parts of the system based on who the authenticated user is and the permissions that the user has been granted.

Role-based access control addresses access control and authentication.

WebSphere InterChange Server supports the following levels of security:

- ▶ None

No security level is set. Therefore, messages are sent from an adapter or WebSphere InterChange Server as normal which is the default level of security.

- ▶ Integrity

The sender, either an adapter or WebSphere InterChange Server, adds a digital signature to the message. The receiver verifies the signature by using the public key of the sender.

- ▶ Privacy

The sender completely encrypts the message. The receiver decrypts the message and passes it on for further processing. The secret key is encrypted with the public key of the receiver. The encrypted secret key can be decrypted only with the private key of the receiver, so the message is secure because only the receiver has access to its own private key.

- ▶ Integrity plus privacy

The sender adds a digital signature to each message, as described for integrity, and then completely encrypts the message, as described for privacy.

► User registry

The user registry is the system that stores user names and passwords. By default, users and passwords are stored in the local InterChange Server repository. The user registry configuration parameters can be used to configure WebSphere InterChange Server to use an LDAP directory as a user registry. The provider is set to LDAP to use an LDAP directory for the user repository, while it is set to REPOS to use the local WebSphere InterChange Server repository as the user registry.

WebSphere Process Server security is based on WebSphere Application Server security. Security tasks can be divided into the security of the WebSphere Process Server environment and the security of applications running in WebSphere Process Server:

► Securing the WebSphere Process Server environment

This type of security involves enabling global administrative security, creating profiles with security, and restricting access to critical functions to selected users.

► Securing an application

This type of security involves the following major aspects:

- Authentication. A user or a process that invokes an application must be authenticated.
- Access control. Whether the authenticated user has permission to perform an operation.
- Integrity and privacy of the data that is accessed by an application.
- Identity propagation with single sign-on (SSO), which permits a user to provide authentication data once and then passes this authentication information to downstream components.

► User registry

Based on WebSphere Application Server capabilities, either federated repositories, the local operating system, LDAP, or a custom user registry can be used.

The implementations for security are quite different between WebSphere InterChange Server and WebSphere Process Server. Investigate the security model for WebSphere Process Server to understand the possible impacts. An example is the different granularity that is offered to administer components through role-based access control when comparing the WebSphere InterChange Server System Manager and the WebSphere Process Server administrative console. Another example is the different ways in which protocols can be secured in WebSphere InterChange Server and WebSphere Process Server.

The following developerWorks article provides an excellent introduction to WebSphere Process Server security:

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

Performance and tuning

From the migrated code perspective, sometimes it is better to modify or redesign migrated components for better performance, if it is in accordance with business requirements.

From the system tuning perspective, WebSphere InterChange Server and WebSphere Process Server offer a broad range of tuning parameters to optimize the system performances of a deployed solution. The tuning parameters are different when comparing WebSphere InterChange Server and WebSphere Process Server.

For example, with the WebSphere InterChange Server, you can define the multi-threading for each collaboration object. In WebSphere Process Server, you can tune thread pools and activation specifications in a more global manner (different granularity). Also, because WebSphere Process Server offers native support for clustering, there are a lot of new possibilities for global system tuning. Therefore, investigate the best practices of WebSphere Process Server tuning, which you can find in developerWorks articles, IBM Redbooks and Redpapers publications, and other IBM publications.

Operations

Because the WebSphere InterChange Server and WebSphere Process Server run times are different, the tools and procedures that are available for operations are also generally quite different and cannot be migrated directly.

Monitoring

In this section, we enumerate the main monitoring facilities and tools that are available for WebSphere InterChange Server and WebSphere Process Server. Monitoring falls generally into two categories, business monitoring and technical monitoring, both of which are discussed in this section.

Monitoring the overall health of the *WebSphere InterChange Server* system includes monitoring all WebSphere InterChange Server components, such as connectors and collaboration objects, as well as the connection to all integrated applications. Several tools exist for monitoring the system. For example, internal tools, such as System Monitor, the InterChange Server Component Management view of System Manager, and Simple Network Management Protocol (SNMP) agent, as well as external tools are available.

You can use the WebSphere InterChange Server Component Management view in System Manager to monitor the WebSphere InterChange Server system and to obtain informational messages for all component status changes in the system. System Monitor is a Web-based tool for monitoring WebSphere InterChange Server. It is configurable to view the current monitoring data and the historical data.

The SNMP agent that is installed with the WebSphere InterChange Server allows an internal SNMP manager to monitor multiple WebSphere InterChange Server instances, collaborations, and connectors, based on a Management Information Base (MIB).

Other technical monitoring products can also be used to monitor the WebSphere InterChange Server, such as IBM Tivoli Monitoring for Business Integration or OMEGAMON® XE.

WebSphere Business Integration Monitor is a stand-alone monitor. By using this monitor, you can monitor a variety of servers, one of which is WebSphere InterChange Server. WebSphere Business Integration Monitor is part of the WebSphere Business Integration Modeler and Monitor product. WebSphere InterChange Server works with WebSphere Business Integration Monitor Version 4.2.4, in conjunction with the WebSphere Business Integration MQ Workflow adapter.

In regard to monitoring for the *WebSphere Process Server*, the WebSphere Process Server operates on top of an installation of WebSphere Application Server. Consequently, it uses much of the functionality of the application server infrastructure for monitoring system performance and troubleshooting metrics. It also includes additional functionality that is specifically designed for monitoring process server service components.

The Performance Monitoring Infrastructure (PMI) section of the administrative console has the options to specify the particular event points and their associated performance measurements. After starting the monitoring service component performance, the generated statistics are published at certain intervals to the Tivoli Performance Viewer. This viewer can be used to observe the results as they occur on the system, and, optionally, log the results to a file that can be later viewed and analyzed within the same viewer.

Common Event Infrastructure (CEI) and Common Business Events can be used for problem determination and business process monitoring. WebSphere Process Server provides native support so that you can perform the following functions:

- ▶ Emit events from SCA, J2C, and BPEL components.

- ▶ Track the events through standard facilities and tools, such as log files, the Common Business Event browser (WebSphere Application Server-based), and the Business Process Choreographer Observer.

With CEI, you can develop specific frameworks and applications to emit events programmatically and collect and process them in a customized way. CEI can be clustered and allows central monitoring of the distributed WebSphere Process Server cluster topology.

Other specialized technical monitoring products can also be used to monitor the WebSphere Process Server, such as IBM Tivoli Composite Application Manager for SOA.

IBM also provides WebSphere Business Monitor Version 6 and native support in WebSphere Integration Developer and WebSphere Process Server to use key performance indicators (KPIs).

Error management

In this section, we focus on error management as it relates to integration logic that is deployed and executed in the server. It has similarities and differences when comparing WebSphere InterChange Server and WebSphere Process Server.

In the context of integration servers, such as WebSphere InterChange Server and WebSphere Process Server, an *event* is a request that is received by the server. It can come from an external source through an asynchronous interaction, such as an inbound application adapter, or through a synchronous interaction.

In the case of synchronous interaction, generally error management is under the responsibility of the caller because the error is sent back to this person. In the case of asynchronous interaction, generally error management is under the responsibility of the integration server. The event then comprises a reference to the business logic that it wants to operate and its data, which is typically stored in a business object.

You can use two tools in WebSphere InterChange Server to locate, view, and process failed events:

- ▶ Failed Event Manager. A browser-based tool with role-based security with which you can work with failed events from the Web.
- ▶ Flow Manager. A tool installed with the WebSphere InterChange Server product.

With WebSphere Process Server, you can use a richer set of components for implementing integration patterns. It relies on separate layers, such as

WebSphere Application Server, the Service Integration Bus, the SCA, and the business process engine. Therefore, error management is more complex than with the WebSphere InterChange Server. In case of an error during an asynchronous interaction, the error can fall into the following categories:

- ▶ **Messaging-based dead letter queues**

This situation can happen if a JMS or MQ binding, for example, is used. The data is then stored in the messaging layer (Service Integration Bus). Such tools as the WebSphere Process Server administrative console, the Service Integration Bus explorer, and certain Java Management Extensions (JMX™) APIs can be used to manage errors at this level.

- ▶ **Business Process instances in failed state**

This situation can occur when an error is raised during a long running process execution. The data is stored in the business process engine database (BPEDB). The Business Process Choreographer Explorer and the Business Flow Manager APIs can be used to manage errors at this level.

- ▶ **Failed Event**

This situation can happen when an error occurs during an SCA asynchronous interaction, for example. The failed event manager (FEM), which is available through the WebSphere Process Server administrative console, and several JMX APIs can be used to manage errors at this level. The FEM for WebSphere Process Server is quite comparable to the FEM for WebSphere InterChange Server. However, FEM provides additional functionalities, such as:

- Changing the payload of the event before resubmission
- Providing a unique interface to access the three main types of errors described previously (Messaging-based dead letter queues, Business Process instances in failed state, and Failed Events)

Administrative tasks

Consider investigating the following points regarding the impact of the migration on administrative tasks:

- ▶ **Administrative consoles**

Because of differences in the deployment model, the component functionalities, and administration granularity, the administrative consoles are quite different:

- In WebSphere InterChange Server:
 - For the administrative tasks, you can use the Component Management view in System Manager to start, stop, pause, and shut down WebSphere InterChange Server components and change component

properties. By using the System Monitor Web interface, you can perform similar actions.

- Several other dedicated tools are also available, such as the Relationship Manager.
- In WebSphere Process Server:
 - By using the WebSphere Application Server-based administrative console, you can execute the major administration tasks, such as configuring clusters and deploying, starting, or stopping applications.
 - Several other dedicated tools are also available through the administrative console, such as the Relationship Manager and the Business Rules Manager.
 - A new Web 2.0 interface called *Business Space* allows you to access those functionalities in a more user-friendly way (for example, to assemble widgets).
- Scripting for administration

WebSphere InterChange Server and WebSphere Process Server provide a rich set of scripts that can be extended and customized for the specific needs of the operation teams. However, the scripts rely on different technologies and cannot be migrated directly. Therefore, the migration has a non-negligible impact on scripting.
- Scheduling

Scheduling is covered by the standard migration tools, as described in 12.2.3, “Scheduler” on page 257.

New operations concepts and skills building

WebSphere Process Server offers numerous new concepts in regard to the operations tasks. Therefore, the operations team must build a new set of skills, procedures, and tools to be operational on the WebSphere Process Server platform.

Further information

You can find further information about administrative tasks and monitoring tools for both WebSphere InterChange Server and WebSphere Process Server in the product documentation.

4.2.4 Deliverables

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

Requirements list

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

Main options to deliver the requirements

At this stage, define the main options to fulfill the requirements. Also, certain decisions cannot be made yet because of a lack of information or experience.

The preparation phase, which is described in 4.3, “Preparation phase” on page 64, helps to resolve the missing points, which include the collection of additional information, decisions, and precise estimates.

Return on investment study

The assessment phase can include an ROI study.

Risk assessment and mitigation plan

The risk assessment must also provide a mitigation plan for each identified risk. The preparation phase can be used to mitigate or even remove certain risks.

Table 4-1 illustrates the following topics:

- ▶ A question or a concern in relation to the migration project
- ▶ The corresponding risk area
- ▶ A proposed mitigation plan

Table 4-1 Basic migration project considerations

Question or concern	Risk area	Mitigation plan
What is the capacity of the development team to deliver migrated components optimized for the WebSphere Process Server run time?	Development skills	<ul style="list-style-type: none">▶ Build WebSphere Process Server and WebSphere Integration Developer skills with training.▶ Get help from the consulting group for best practices.
Is the operations team ready to administer the new WebSphere Process Server solution?	Operations team skills and tools	<ul style="list-style-type: none">▶ Build WebSphere Process Server skills with training.▶ Run a pilot project before the migration project to create and test new administrative tools and procedures.▶ Get help from the CoE or consulting for best practices.

Question or concern	Risk area	Mitigation plan
Who is in charge to decide which services can be reused and the funding model to use them?	SOA governance enablement and maturity	<ul style="list-style-type: none"> ▶ Get help from the SOA CoE or the IT governance structure already in place.
From the technical perspective, it looks like we are ready, but how do we get interest and funding from the line of business?	Business and IT alignment	<ul style="list-style-type: none"> ▶ Get help from the SOA CoE for sponsorship. ▶ Add business value to the project by using new product capabilities around BPM.
It looks like the project is too costly, because we must run a parallel infrastructure during an extended period of time.	Project cost	<ul style="list-style-type: none"> ▶ Investigate the possibilities around WebSphere Process Server and WebSphere InterChange Server interoperability to reduce development costs. ▶ Investigate the Extended Entitlement possibilities. ▶ Add business value to get more funding.

High-level planning

At this stage, you can provide high-level planning with rough estimates for effort and costs.

4.3 Preparation phase

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

4.3.1 Building skills

Ensure that the skill set is properly updated to fulfill the migration implementation needs. The purpose is to have sufficient knowledge of WebSphere Process Server. Education and training are a prerequisite, but it is generally insufficient. Take time to have a real practice.

4.3.2 Using the WebSphere Process Server project experience

Ideally, run a WebSphere Process Server project before doing the actual migration implementation. If the project team has already gone through a full life cycle with WebSphere Process Server and has a working WebSphere Process

Server infrastructure, all the related risks are already eliminated, and you can concentrate on the pure migration tasks.

A real experience can be obtained by performing a pilot project. A pilot project has two advantages:

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

A pilot project must have the following objectives:

- ▶ To build overall skills around WebSphere Process Server

Consider the following examples:

- Development skills
Set up best practices, a methodology, and so on.
- Operations skills
Set up new procedures and tools to administer the solution properly.
- Project management skills
Update or set up new project activities and refine estimates to get them delivered (workload assessment).

- ▶ To enable the new infrastructure for the WebSphere Process Server solution

A typical example is the effort around clustering and security. WebSphere Process Server provides native clustering for HA management, load balancing, and integration with security managers to define roles. These concepts are generally new to WebSphere InterChange Server users and require practical experience to be implemented efficiently as in the following examples:

- Establish best practices to set up the cluster topology and the security roles through an LDAP directory or custom registry.
- Establish best practices to use and administer the cluster. For example, determine how you want to monitor a distributed platform or perform log collection and analysis. Another example is to determine the required infrastructure for shared resources that are highly available.
- ▶ To deliver new functionalities that add business and IT value, compared to the previous WebSphere InterChange Server solution, as in the following examples:
 - Use human tasks to create an application that is dedicated to functional error and is used by business users, which in consequence spares IT resources that are usually dedicated to support users.

- Enable business rules to leave certain control and dynamic decisions to the business users.
 - Set up WebSphere Process Server clustering to raise high availability and load balancing capabilities.
 - Enable technical event tracking or business monitoring through KPIs, based on CEI capabilities.
 - ▶ To gain an independent point of view of the capabilities of the standard migration tools regarding integration patterns
- By implementing integration patterns from scratch in WebSphere Process Server, the project team will have a more accurate idea about the capabilities of the standard migration tools regarding integration patterns.

4.3.3 Refining the assessment

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

Detailed study

A detailed study might be required to specify the technical migration path to migrate the WebSphere InterChange Server artifacts, composing the EAI solution, into the WebSphere Process Server artifacts, composing the SOA solution.

For example, a gap analysis might be required for an accurate status of the functional parity between the WebSphere InterChange Server and WebSphere Process Server products. Such an analysis covers the following topics:

- ▶ It identifies new features in WebSphere Process Server.
- ▶ It identifies features that have disappeared or are not equivalent in WebSphere Process Server.
- ▶ It provides solutions to fill in the gaps.

Chapter 9, “Migration implementation approach” on page 151, is dedicated to product implementation details, and it can be used as a source for input in the detailed study and the gap analysis.

Effort estimation

Based on the decided approach for migration, a precise migration effort estimate needs to be performed. The estimate will become a guide to the scale of effort required to migrate the interfaces. IBM uses a methodology called the Business Integration Value Assessment (BIVA) method. It is used to estimate the standard effort for new development and testing. This estimation methodology takes a

census of the solution artifacts that have been developed and is a basis for the migration assessment. The detailed discussion of the estimation methodology is beyond the scope of this book. IBM Software Services for WebSphere can help in this exercise.

Reducing migration effort

There are several ways to reduce the migration effort. Following one or more of these techniques can result in reduced migration effort. It is certainly not an exhaustive list:

- ▶ Use more experienced resources.

The first and foremost method is to use development resources with extensive WebSphere Process Server implementation experience, at the very least as an advisor or technical lead to mentor the other developers. A very experienced WebSphere Process Server Architect and migration expert acting as an advisor or technical lead to mentor the other developers helps reduce costs as well.

- ▶ Exploit improved building and testing features.

Another method is to exploit the improved testing features in WebSphere Process Server V6.2 to automate the building, deployment, and testing (including regression) of the deployed code.

- ▶ Use Patterns-Based Engineering.

Pattern identification can be performed, and patterns can be created using JET2 technology for repeating design patterns used.

IBM Software Services for WebSphere can help in all of these activities.

Decisions and precise estimates for effort and budget

At this stage, you can choose between the remaining options that come from the assessment phase and assess precisely the corresponding resources of effort and budget.

4.4 Implementation phase

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

- ▶ Complete the assessment and preparation phases to ensure that the correct resources and skills are ready and set up to deliver the migration project.

- ▶ Define precisely the overall strategy for migration, such as decisions about migrating WebSphere InterChange Server flows in stages, by using functional decomposition or technical decomposition.

In the following sections, we describe the steps to execute the implementation phase. Provided that the assessment and the preparation phases have been conducted successfully, the implementation phase closes the usual project life cycle.

4.4.1 Development

The development team must define the technical migration path. For example, it must decide which artifacts can be migrated with the standard migration tools and what needs to be customized and newly developed. Completely new modules can be also considered.

See “Development” on page 47 for further considerations about this phase.

4.4.2 Test

The test phase includes tasks, such as the following examples:

- ▶ Functional and integration testing (for new developments)
- ▶ Regression tests (for migrated developments)
- ▶ Performance tests
- ▶ Robustness tests

See “Testing” on page 49 for further considerations about this phase.

4.4.3 Deployment

The WebSphere Process Server environment must be set up (installation and configuration subphases). Then the migrated components can be deployed into it. As a reminder, it is possible that, during the transition period, a parallel infrastructure is necessary.

For further considerations about this phase, see the following sections:

- ▶ “Deployment” on page 50
- ▶ “Run time, quality of service, and security” on page 54

4.4.4 Go live

As a reminder, the operations team must be able to administer the migrated solution properly. See “Operations” on page 58 for further considerations about this phase.

Archived



Part 2

Migration implementation concepts

In this part, we discuss the relevant concepts to get started with the migration implementation. This part includes the following chapters:

- ▶ In Chapter 5, “Product overview” on page 73, we provide a product overview of WebSphere InterChange Server, WebSphere Business Integration Adapters, and WebSphere Process Server.
- ▶ In Chapter 6, “Product feature comparison” on page 109, we provide a feature comparison between WebSphere InterChange Server and WebSphere Process Server.
- ▶ In Chapter 7, “Server component upgrade” on page 123, we review the WebSphere InterChange Server components upgrade.
- ▶ In Chapter 8, “Adapters and data handlers upgrade” on page 131, we review the WebSphere Business Integration Adapters and WebSphere Business Integration data handlers upgrade.

In Chapter 7, “Server component upgrade” on page 123 and Chapter 8, “Adapters and data handlers upgrade” on page 131, we describe specifically how various artifact types in WebSphere InterChange Server and WebSphere Business Integration Adapter correspond to WebSphere Process Server components. We also discuss the technical benefits and limitations of upgrading.

- In Chapter 9, “Migration implementation approach” on page 151, we progress to the solution perspective and discuss an approach to determine the best technical path for migration implementation. We begin with an exposition of relevant integration principles. Then we explain the technical challenges to be expected during the migration implementation. Finally, we provide recommendations to select the appropriate migration path.

Product overview

As an introduction to the implementation migration concepts, in this chapter, we explain the main features of the IBM WebSphere InterChange Server and the WebSphere Process Server products.

We include the following sections:

- ▶ 5.1, “WebSphere InterChange Server” on page 74
- ▶ 5.2, “WebSphere Business Integration Adapters” on page 82
- ▶ 5.3, “WebSphere Business Integration data handlers” on page 87
- ▶ 5.4, “WebSphere Process Server” on page 90
- ▶ 5.5, “WebSphere Adapters” on page 99
- ▶ 5.6, “WebSphere Process Server bindings” on page 101
- ▶ 5.7, “WebSphere Process Server data bindings” on page 104

The first three sections are relevant to WebSphere InterChange Server, and the last four sections are relevant to WebSphere Process Server.

5.1 WebSphere InterChange Server

In this section, we focus on concepts that are related to WebSphere InterChange Server. For more information, see 5.2, “WebSphere Business Integration Adapters” on page 82, and 5.3, “WebSphere Business Integration data handlers” on page 87.

5.1.1 Introduction to WebSphere InterChange Server

WebSphere InterChange Server is an integration broker with a set of adapters that allows heterogeneous business applications to exchange data, in the form of business objects. A distributed, hub-and-spoke infrastructure enables the execution of business processes across the Internet and across applications on local networks.

5.1.2 Architectural overview of WebSphere InterChange Server

WebSphere InterChange Server uses adapters that are part of the WebSphere Business Integration Adapters product, together with modular and customizable components, such as collaborations, business objects, and maps, to perform these integration tasks. Figure 5-1 on page 75 illustrates the main components of the WebSphere InterChange Server architecture. We explain these components in the following sections.

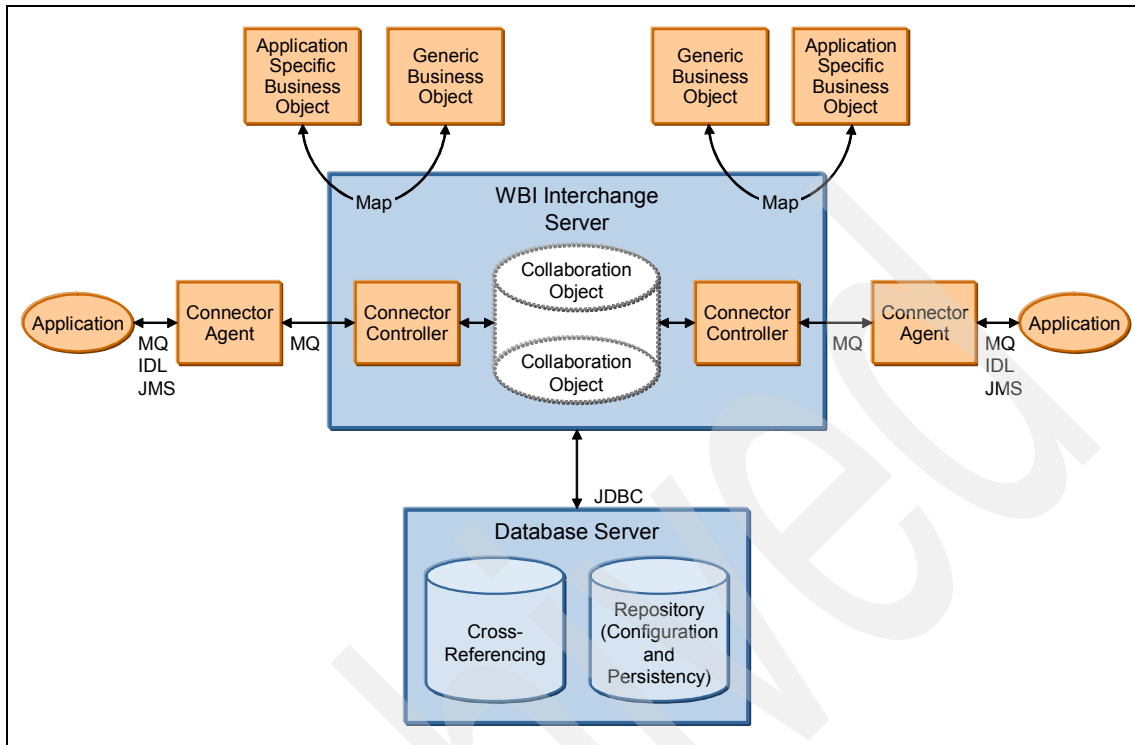


Figure 5-1 WebSphere InterChange Server components

Collaborations

WebSphere InterChange Server *collaborations* are software modules, which contain logic that describes a business process. A collaboration can be a simple business process that consists of a few steps, or it can be a complex model that involves a group of collaborations. Collaborations act as the hub through which data is exchanged in the form of business objects.

Collaborations can be distributed across any number of applications, handle synchronous and asynchronous service calls, and support long-lived business processes (LLBP). The toolset that is available with WebSphere InterChange Server includes a collaboration design tool called *Process Designer*, which can be used to customize and create collaboration templates.

Business objects

Business objects represent the data and processing instructions between an integration broker and connected applications in a business integration system. Business objects can represent one of the following types:

- ▶ A request to an adapter from an integration broker
- ▶ An event from an adapter to an integration broker
- ▶ A call from an external site using server access interface

The communication between two disparate applications is ensured by using adapters, WebSphere InterChange Server, application-specific business objects, and generic business objects.

Application-specific business objects

Information is exchanged between the WebSphere Business Integration Adapter and the WebSphere InterChange Server in the form of application-specific business objects. These application-specific business objects model the appropriate application entries and the adapters communicate with their applications by using the information that is available in their application-specific entities.

Generic business objects

Generic business objects contain the superset of information for the application entities that need to communicate. Maps are used to transform data from one application-specific business object into a generic business object and then from the generic business object to another application-specific business object. The communication between two disparate applications is thereby ensured by using adapters, WebSphere InterChange Server, application-specific business objects, and generic business objects.

To summarize, adapters use application-specific business objects to transfer data between the WebSphere InterChange Server and the particular application. These application-specific business objects are unique to the adapter and reflect the application's data model. Data that is processed within the logic of a collaboration is in the form of a generic business object. Generic business objects are application-independent and, therefore, can be reused in the future.

Maps

Maps transform data to and from generic business objects and application-specific business objects. This way, adapters can communicate with their applications by using application-specific entities, while collaborations can apply integration logic in an application-independent way.

Typically, application-specific business objects are mapped to generic business objects when they are received by the WebSphere InterChange Server. When a

business object is sent to an adapter, the generic business object is mapped to the appropriate application-specific business object. The response back from the application is mapped back to the generic business object.

Adapters

From the WebSphere InterChange Server perspective, adapters are decomposed into the following parts:

- ▶ A controller part that runs inside the WebSphere InterChange Server
- ▶ An agent part that runs in a separate Java virtual machine (JVM) outside the WebSphere InterChange Server JVM

The controller communicates with the agent by exchanging serialized application-specific business objects (ASBOs) through a configurable transport protocol, such as MQ, Internet Inter-ORB Protocol (IIOP), or Java Message Service (JMS). The controller is also in charge of executing the maps. For historical reasons, sometimes the WebSphere Business Integration Adapter is also called a *connector*.

For information about WebSphere Business Integration Adapter agents, see 5.2, “WebSphere Business Integration Adapters” on page 82.

WebSphere Business Integration Toolset

The WebSphere Business Integration Toolset provides administrative and development tools for use with WebSphere InterChange Server. The toolset includes the following administrative tools:

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

The toolset includes the following development tools:

- ▶ Map Designer
- ▶ Relationship Manager
- ▶ Process Designer
- ▶ Business Object Designer
- ▶ Connector Configurator

These tools enable you to perform the following actions:

- ▶ Develop and customize components.
- ▶ Organize your components into projects for deployment in successive stages of development and production.
- ▶ Monitor a deployed WebSphere InterChange Server integration system.

5.1.3 Features of WebSphere InterChange Server

In this section, we describe the services and features of WebSphere InterChange Server.

Event management service

WebSphere InterChange Server persistently stores every business object that it receives during the execution of a collaboration. By storing every business object, the server can recover from an unexpected termination or from the failure of a collaboration without losing event notifications or calls.

Connector controllers

A *connector controller* is an interface between the client-side of a connector and WebSphere InterChange Server. A connector controller routes business objects as they traverse the WebSphere InterChange Server system, linking the client-side of connectors to collaborations and managing the mapping process.

Through the connector controller, an administrator can perform the following functions:

- ▶ Trace interactions between WebSphere InterChange Server and the connector agent.
- ▶ Enable and disable interactions between WebSphere InterChange Server and the connector agent.
- ▶ Specify the type of mapping to perform for each business object arriving from or departing to the connector agent.

Repository

WebSphere InterChange Server maintains configuration information and definitions of all objects in a persistent store called the *InterChange Server repository*, which consists of a set of tables in a relational database. The tables store object definitions and configuration information in the form of Extensible Markup Language (XML) documents.

Database connectivity service

The database connectivity service manages interactions between WebSphere InterChange Server and the repository. The database connectivity service interacts with the repository by means of the Java Database Connectivity (JDBC) API.

Database connection pools

The System Manager tool of the WebSphere InterChange Server system can be used to define database connection pools. User-defined database connection pools make it possible for developers to directly access relational databases from within a collaboration or map.

The database connection pools provide support for the following features:

- ▶ Automated database connection life-cycle management
- ▶ Simplified APIs for SQL statements and stored procedure execution
- ▶ Container-managed transaction bracketing

High availability

The WebSphere InterChange Server system can be configured in conjunction with an external clustering system to provide high availability (HA). When a hardware failure is detected, the HA configuration provides shutdown, automatic migration, and restart of the WebSphere InterChange Server on the cluster backup system and the third-party software and HA-supported connectors. The cluster backup server becomes the active server, assuming the cluster name and IP address of the primary server. This backup server automatically takes over system processing until the time at which the failure is corrected on the primary server and a failback is initiated.

Transaction service

Any application integration solution encounters risks when it moves data from one application to another. When data leaves the protected realm of one application and its database and travels across a network to another application, any number of problems can occur, such as the following examples:

- ▶ The network might go down.
- ▶ The destination application might crash before receiving the data.
- ▶ An error might occur in the destination application before it has a chance to process the new data.

Traditionally, painstaking cross-checking was required, or the data remained faulty until someone noticed the problem. However, the WebSphere InterChange Server system offers a higher level of service with strict controls to which you can

trust enterprise data, with services that can run a collaboration as though it were a type of transaction.

Transactional collaborations

Transactional qualities are desirable and achievable for collaborations where data consistency is important across applications. Like other transactions, a transactional collaboration involves a set of steps. If an error occurs, WebSphere InterChange Server can undo each completed step, performing a transaction-like rollback.

However, collaborations differ from traditional transactions in the following important ways:

- ▶ Collaboration actions are distributed, and there is no centralized control over the participating databases.
- ▶ Collaborations that respond to events (as in the publish-and-subscribe model) are long-lived. The collaborations execute asynchronously, because isolation of the application data while the collaborations execute adversely affects application users.
- ▶ Applications save data changes that are caused by collaborations, thereby providing a distributed, cross-application form of durability. However, if a collaboration must roll back, it might need to undo previously saved operations.

Therefore, the techniques that WebSphere InterChange Server uses to support transactional collaborations differ from those that support traditional transactions. The transaction levels that are associated with collaborations define the rigor with which WebSphere InterChange Server enforces transactional semantics.

Recovery features

The WebSphere InterChange Server implementation provides the following features to improve the time that it takes the server to reboot after a failure, to make WebSphere InterChange Server available for other work before all flows have been recovered, and to control the resubmission of failed events:

- ▶ Asynchronous recovery

WebSphere InterChange Server does not wait for collaborations and connectors to recover before it completes startup. Collaborations and connectors are allowed to recover asynchronously after WebSphere InterChange Server has started. This makes it possible to use System Manager troubleshooting tools, such as System Monitor and the Administer Unresolved Flows window, while the connectors and collaborations are still recovering.

► Deferred recovery

Use of the deferred recovery feature is optional and is configured through the use of collaboration object properties. If this feature is enabled for a collaboration when an WebSphere InterChange Server failure occurs, the recovery of the collaboration's work-in-progress flows is deferred until after the server has rebooted, thereby saving the memory usage that is associated with those flows. After the server has rebooted, use the Administer Unresolved Flows window in System Manager to resubmit the events.

► Persist service call in-transit state

You might want to prevent a recovery from automatically resubmitting all service calls that were in transit when a failure occurred. You can do this to avoid the possibility of a non-transactional collaboration sending duplicate events to a destination application.

To accomplish the persist service call in-transit state, configure the collaboration prior to the server failure so that it persists any service call event in the in-transit state when a failure and a recovery occur. When WebSphere InterChange Server recovers, the flows that were processing service calls remain in the in-transit state. You can use the Administer Unresolved Flows window to examine individual unresolved flows and control when (or if) they are resubmitted.

► Guaranteed event delivery features

For JMS-enabled connectors, the following optional features can be useful for guaranteed event delivery in recovery situations. These features are used only with connectors that use JMS as their transport mechanism:

– Container-managed events

The container-managed events feature is valid for JMS-enabled connectors that use a JMS event store. It ensures that, if a system crash and a recovery occur, an event that was in process between the event store and the connector framework is received exactly once by the connector framework and is not delivered twice.

– Duplicate event elimination

The duplicate event elimination feature is valid for JMS-enabled connectors. This feature uses unique event identifiers in the application-specific code of the connector to ensure that events are not delivered in duplicate to the delivery queue.

5.2 WebSphere Business Integration Adapters

The WebSphere Business Integration Adapters consist of two components. The WebSphere Business Integration Adapter Framework provides a common environment for the various adapters while each adapter implements the Enterprise Information System (EIS)-specific connectivity.

5.2.1 WebSphere Business Integration Adapter Framework

The IBM implementation of the WebSphere Business Integration Adapters has a general component called *WebSphere Business Integration Adapter Framework*. This component is the common runtime environment that is a prerequisite for all WebSphere Business Integration Adapters and for the WebSphere Business Integration Adapter Development Kit. The framework includes a combination of runtime libraries and operational administrative tools. The runtime libraries can be deployed on all supported operating systems.

5.2.2 WebSphere Business Integration Adapters

WebSphere Business Integration Adapters (Figure 5-2) represent a collection of software programs, tools, and APIs that an enterprise can use to enable EISs to exchange business data with an integration server.

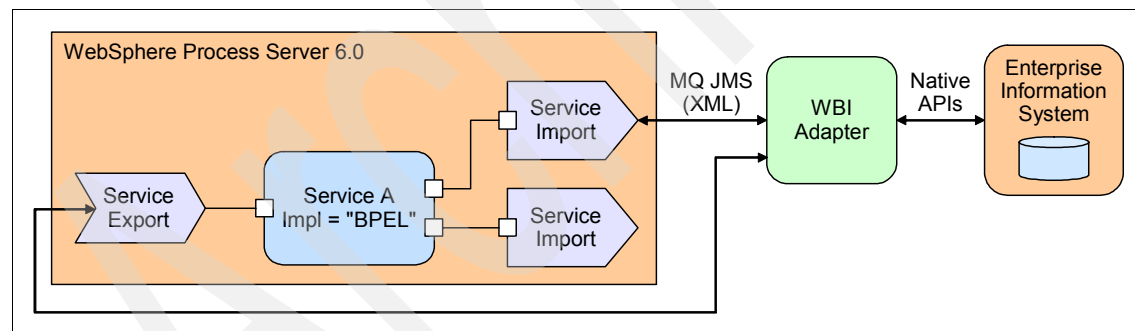


Figure 5-2 WebSphere Business Integration Adapters

This collection can include WebSphere InterChange Server, WebSphere Message Broker, WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Application Server Enterprise. Each EIS requires its own application-specific adapter to participate in the business integration system.

WebSphere Business Integration Adapter includes the following tools:

- ▶ A connector that links the application to the integration server
- ▶ Tools with GUIs to help the user configure the connector and create the business object definitions that are needed for the EIS
- ▶ An Object Discovery Agent (ODA) for several of the adapters, which introspects EIS metadata to generate business objects
- ▶ An Adapter Development Kit (ADK), which provides a framework for developing custom adapters in Java or C++ technology if an adapter for a particular existing or specialized EIS is unavailable

An adapter mediates between an application and one or more WebSphere InterChange Server collaborations. Adapters can be specific to an application, such as SAP, or to a technology, such as a data format or protocol (JMS, Web services, JDBC, JText, XML, electronic data interchange (EDI), and so on).

An adapter communicates with the WebSphere InterChange Server in two ways:

- ▶ Application event notifications
- ▶ Request processing

Adapters can be configured for applications that reside on the local network and for remote applications that reside across an Internet firewall. Most adapters share certain common behaviors, differing only in the manner in which they interact with applications and application-specific business objects.

In the following sections, we describe the adapters that we use or discuss in this book.

WebSphere Business Integration Adapter for SAP

In this book, we use the Business Application Programming Interface (BAPI®) Module and the Application Link Enabling (ALE) Module. Those modules are Java-written subcomponents of the adapter that enable the integration broker to communicate with an SAP R/3® application server for outbound and inbound operations. The modules use the standard SAP Java Connector (SAP JCo) to establish a connection to the SAP system.

The BAPI Module can access any remote-enabled function module in the SAP system. As an SAP JCo client application, this module invokes the SAP Remote Function Call (RFC) library to connect to the SAP Gateway of the respective application server.

The ALE Module is best used for objects, such as batch objects, that are asynchronous in nature. It uses *push* technology that requires a server listening for events. Processes called *registering* and *installing* notify the server about

what to listen to and from whom to expect information. Registering involves using a program identifier to give the SAP Gateway a communication point with listener threads (servers). The ALE Module uses the RFC Server Module for event handling.

Both modules represent a noninvasive integration approach with SAP system. They do not require heavy modification or adjustments in the accessed SAP back-end system.

More modules are available in the SAP adapter to fulfill different missions. For information about the Hierarchical Dynamic Retrieve Module, RFC Server Module, Application Link Enabling (ALE) Module, ABAP™ Extension Module, BAPI Module, see the Redbooks publication *WebSphere Business Integration for SAP*, SG24-6354.

WebSphere Business Integration Adapter for PeopleSoft

The connector, which is written in Java, complies with IBM business integration system standards for adapters. It establishes a PeopleSoft session object and uses the standard connector methods to process components. The connector requires a PeopleSoft Business Component and Component Interface for each hierarchical business object that it processes.

At startup, the connector creates a session object through which it connects to the PeopleSoft Application Server. Connecting to the Application Server gives the connector access to the APIs for all the Component Interfaces that correspond to its supported business objects. The server also provides access to the PeopleCode and the Application Designer objects included with the adapter for event notification.

WebSphere Business Integration Adapter for JText

The adapter for JText enables an integration broker to communicate with an application by exchanging text or binary files. This connector facilitates the integration of data with applications that lack an API.

Use the JText adapter in the following situations:

- ▶ An application does not have a C, C++, or Java standard API through which an integration broker can communicate.
- ▶ It is not feasible to have an event table for a custom-built application.
- ▶ String or binary files are the most appropriate method for exchanging data.

In all of these cases, the simplest method for integrating an application into a larger system might be by exchanging string or binary files through the JText connector.

WebSphere Business Integration Adapter for Java Message Service

The connector for JMS is a runtime component of the WebSphere Business Integration Adapter for JMS. By using the connector, WebSphere integration brokers can exchange business objects with applications that send or receive data in the form of JMS messages. JMS is an open-standard API for accessing enterprise-messaging systems. It allows business applications to send and receive business data and events defined as messages.

The adapter supports both the point-to-point messaging and publish-and-subscribe (pub/sub) messaging interfaces that are defined by the JMS standard. These styles are also commonly referred to as *queue-based messaging* and *topic-based messaging*. A single instance of the adapter supports only one messaging style at a time. That is, topics and queues cannot be mixed in the configuration. However, both messaging styles can be supported by running multiple instances of the adapter in parallel with instances configured for either point-to-point or publish-and-subscribe.

For details about configuring this adapter, see the “User Guide for Adapters” at the following address:

http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbia_adapters.doc/pdfs_adapters.html

WebSphere Business Integration Adapter for Enterprise JavaBeans

The connector for Enterprise JavaBeans (EJB) architecture is a runtime component of the WebSphere Business Integration Adapter for the EJB architecture. The EJB architecture adapter includes a connector, message files, configuration tools, and an ODA. With this connector, the WebSphere integration broker can exchange business objects with enterprise beans that have been designed by using the EJB architecture and deployed on an application server. The connector enables business processes in the broker to pass data to one or more enterprise bean business methods and receive returned data. The adapter is unidirectional; it provides request processing functionality only. It does not perform event notification.

The request processing scenario described here assumes the following points:

- ▶ The enterprise beans are deployed on a Java Platform 2, Enterprise Edition (J2EE)-compliant enterprise application server, such as WebSphere Application Server.
- ▶ The application server is installed and running.

- The connector has been initialized, which loads the ProviderURL and InitialContextFactory connector-specific properties. These properties help to obtain the Java Naming and Directory Interface (JNDI) initial context, which is required to initiate the connection to the EJB server and locate the enterprise bean's home interface.

WebSphere Business Integration Adapter for Web services

The connector is a runtime component of the WebSphere Business Integration Adapter for Web services. With this connector, businesses can aggregate, publish, and consume Web services. The connector and other components provide the functionality that is needed to exchange business object information in the body of a SOAP message. WebSphere Business Integration Adapter for Web services supports SOAP 1.1 and 1.2 in conformance with the Web Services-Interoperability (WS-I) Organization.

The WebSphere Business Integration Adapter for Web services exposes WebSphere integration broker modules as Web services. This adapter can also communicate to externally hosted Web services with enhanced configurability. A Web Services Description Language (WSDL) ODA eases the generation and deployment of business objects.

WebSphere Business Integration Adapter for WebSphere MQ

The connector for WebSphere MQ is a runtime component of the WebSphere Business Integration Adapter for WebSphere MQ. With this connector, the WebSphere integration broker can exchange business objects with applications that send or receive data in the form of WebSphere MQ messages.

The connector uses the WebSphere MQ implementation of the JMS. The JMS is an open-standard API for accessing enterprise messaging systems. By using this API, business applications can asynchronously send and receive business data and events. Accordingly, you must have a compatible version of the WebSphere MQ software installed. In addition to that, you must have the WebSphere MQ Java Client Libraries.

WebSphere Business Integration Adapter for JDBC

The connector for JDBC enables the integration broker to exchange business objects with an application that is built on any database that is supported by a driver that follows the JDBC 2.0 or later specification.

The connector executes SQL statements or stored procedures to retrieve or change data in the database or application. To build dynamic SQL statements or stored procedures, the connector uses application-specific metadata. These SQL statements and stored procedures perform the required retrieval from or changes to the database or application.

Custom WebSphere Business Integration Adapter

IBM provides a Java Connector Development Kit so that you can design custom WebSphere Business Integration Adapters. The connector framework contains all the code that is necessary for the connector to interact with an integration broker and provides a basic infrastructure for interaction with the specific application.

5.3 WebSphere Business Integration data handlers

A *data handler* is a Java class instance that converts a data object between a particular serialized data format and its in-memory representation, usually a form of business object. Data handlers are used by the components of a business integration system that transfers information between a WebSphere business integration broker and an external process. Often, the external process uses a common format, such as XML, for its native serialized data.

Rather than having every adapter (or access client) handle the transformation between these common formats and business objects, the WebSphere Business Integration system provides several IBM-delivered data handlers. In addition, you can create custom data handlers to handle conversion between your own native format. The adapter (or access client) can then call the appropriate data handler to perform the data conversion based on the Multipurpose Internet Mail Extension (MIME) type of the serialized data.

In the following sections, we describe the most popular WebSphere Business Integration data handlers.

5.3.1 XML data handler

The XML data handler is a data-conversion module whose primary role is to convert business objects to and from XML documents. An *XML document* is serialized data with the text or XML MIME type.

The XML data handler uses business-object definitions when it converts between business objects and XML documents. It determines how to perform the conversion by using the structure of the business object definition and its application-specific information. A properly-constructed business object definition ensures that the data handler can correctly convert a business object to an XML document and an XML document to a business object.

The XML data handler uses following components to convert XML data to a business object:

- Name handler

- ▶ Simple API for XML (SAX) parser
- ▶ (Optional) entity resolver

This component resolves the references if the XML document has a document type definition (DTD) and it includes entity references.

The XML data handler can be used by connectors and access clients.

5.3.2 Delimited data handler

The *delimited data handler* is a data-conversion module whose primary role is to convert business objects to and from delimited-formatted strings or streams. A *delimited-formatted string* or *stream* is serialized data with the text or delimited MIME type. The data handler parses text data based on a specified delimiter that separates the individual fields of a business object's data. This type of data conversion is used primarily where the efficiency of machine reading is most important.

By using the delimited data handler, you can set the following strings:

- ▶ Delimiter

The data handler uses a delimiter to separate the fields in the delimited data. You can set the delimiter meta-object attribute to the desired delimiter for your data.

By default, the data handler uses the tilde character (~), because the delimiter is the size of the attribute property MaxLength to determine how to read and write data. MaxLength is an business object attribute property that defines the maximum number of characters of the attribute value, including padding to allow for right-justified or left-justified text. MaxLength is read from the definition of the business object in the repository. Accordingly, the main requirement for a business object is that the MaxLength for each string attribute is set appropriately.

- ▶ Escape string

The data handler uses an escape string to configure a string to escape the delimiter and escape strings. The escape string allows the attribute value data to contain delimiter-like and escape-like strings. You can set the escape meta-object attribute to configure the escape string. By default, the data handler uses the backslash character (\) as the escape string.

5.3.3 FixedWidth data handler

The *FixedWidth data handler* is a data-conversion module whose primary role is to convert business objects to and from strings or streams that have a format of fixed-width fields. A *fixed-width string* or *stream* is serialized data with the text or

fixed-width MIME type. The data handler parses text data by using fixed-width fields. The field lengths are specified by the `MaxLength` property of each business object attribute. The value of this property is stored in the business object definition.

The `FixedWidth` data handler uses the value of the `MaxLength` property of attributes in a business object definition to determine how to read and write data. You can configure a pad character that represents spaces to add to or remove from the data for alignment. These pad characters are added when converting business objects to strings. They are removed when converting strings to business objects. You can also configure the alignment of a business object attribute value to be left-justified or right-justified, which makes it possible for the attribute value data to retain meaningful characters.

5.3.4 Complex data handler

The *complex data handler* is a data-conversion module whose primary role is to convert business objects to and from specific data formats. The complex data handler uses the ContentMaster product, from Itemfield, Inc., and the WebSphere Business Integration XML data handler to perform these conversions.

5.3.5 EDI data handler

The *EDI data handler* is a data-conversion module whose primary role is to convert business objects to and from EDI documents. An *EDI document* is a standardized format for conveying business information. The EDI data handler supports two message standards: X.12 and EDIFACT.

5.3.6 Custom data handler

As with the other IBM-delivered data handlers, a *custom data handler* converts a business object to a specific serialized data format. It also converts serialized data in a specific format to a business object.

Data handlers are written in Java. The Data Handler API provides the abstract `DataHandler` base class to facilitate the development of a custom data handler. This class contains the methods that populate a business object with values extracted from input data. It also contains methods that serialize a business object into a string or a stream. In addition, the class includes utility methods that a custom data handler can use.

5.4 WebSphere Process Server

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

5.4.1 Introduction to WebSphere Process Server

WebSphere Process Server is an advanced business process integration server that is built over the industry standard service-oriented architecture (SOA). WebSphere Process Server uses the Service Component Architecture (SCA) programming model and the Service Data Object (SDO) data model. Business objects are transformed, routed, and mapped by using the SCA components.

WebSphere Adapters, Native Data Bindings, and WebSphere Business Integration Adapters provide the connectivity to the back-end EIS. WebSphere Process Server uses Web Services Business Process Execution Language (WS-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 WebSphere Process Server and can be used to provide connectivity to Web services and the mediation flows. Within WebSphere Process Server, Common Event Infrastructure (CEI) can be used for the monitoring of business events that can occur in the applications that run 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. WebSphere Business Modeler can be used to model the business processes for WebSphere Process Server, and WebSphere Business Monitor can be used to monitor the business processes that run on WebSphere Process Server.

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

5.4.2 Architectural overview of WebSphere Process Server

WebSphere Process Server is a BPEL-based business process engine that is built on an SOA integration platform. The integration platform relies on a uniform invocation programming model and a uniform data representation model. The uniform invocation model is called Service Component Architecture (SCA). The uniform data representation model is called Service Data Object. WebSphere Process Server provides components that are built over the SCA and grouped into modules to provide decoupling and the separation of concerns. This model

allows you to separate the business logic from the integration logic in a clear fashion.

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 as illustrated in Figure 5-3:

- ▶ The *SOA core layer* consists of the SCA, the business objects, and the CEI component. The SCA and the business objects provide the universal invocation and the data representation programming models. The CEI generates events for monitoring applications that are running on WebSphere Process Server.
- ▶ The *supporting services layer* provides the components that are necessary for transformation of business objects and the interfaces.
- ▶ The *service components layer* provides the functional components that are required for composite applications.

The combination of all the components in WebSphere Process Server and the development environment provided by WebSphere Integration Developer, coupled with the usage of WebSphere Adapters and Native Bindings, facilitates quick development and the deployment of integration solutions.

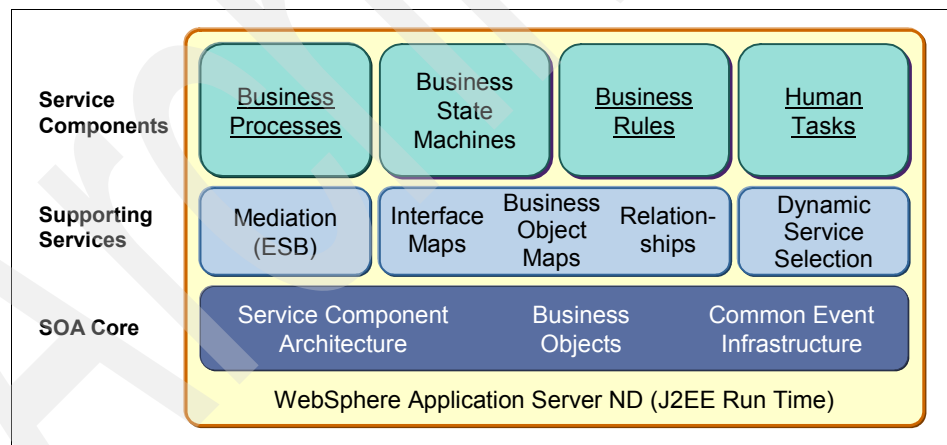


Figure 5-3 Components of WebSphere Process Server

Service-oriented architecture core

The SOA core of WebSphere Process Server provides universal invocation (SCA) and data-representation (business objects) programming models for the applications that run on WebSphere Process Server. It also provides monitoring and management capabilities (CEI) for these applications. See Figure 5-4.

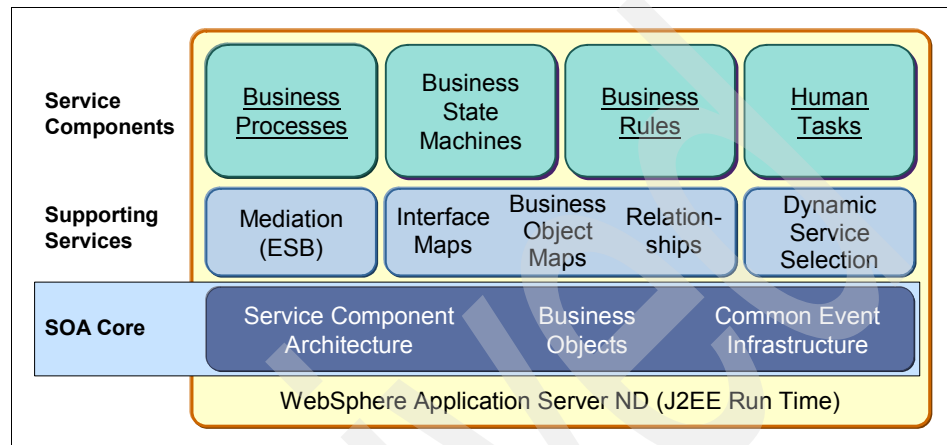


Figure 5-4 Service-oriented architecture core layer of WebSphere Process Server

Service Component Architecture

The SCA (Figure 5-5 on page 93) provides an abstraction layer around the service components. All integration artifacts that run on WebSphere Process Server are represented as components with well defined interfaces. Within the SCA, a service component defines a service implementation. All service components, also called *SCA components*, have an interface and can be wired together to form a module that is deployed to WebSphere Process Server. This capability enables changing any part of the application without changing the other parts by changing the corresponding SCA component. SCA components can also interact with other applications on an EIS by using WebSphere Adapters and Native Bindings. A component within the SCA 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.

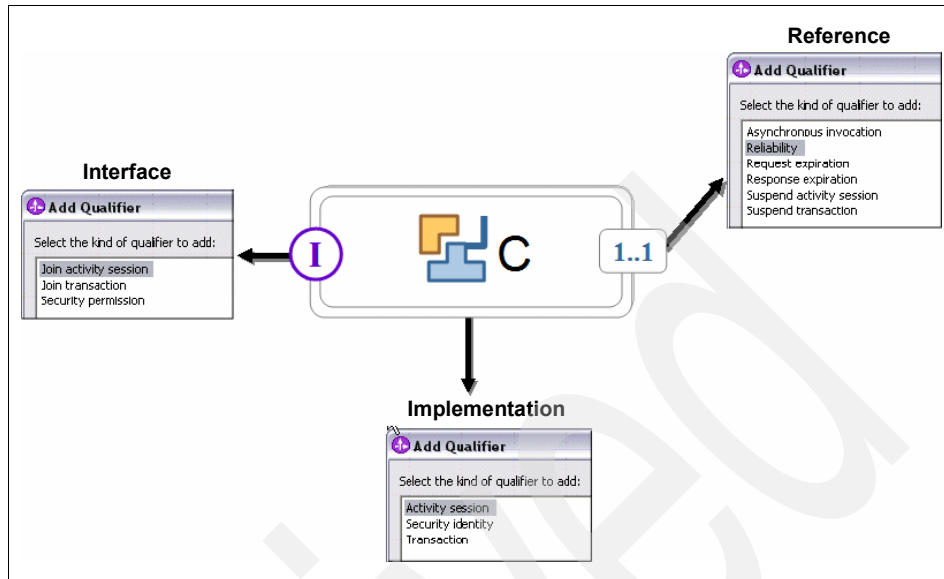


Figure 5-5 Service Component Architecture

Service Data Objects and Business Objects

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

Business objects in WebSphere Process Server are based on Service Data Objects (SDOs). An *SDO* is a framework for the data application development. An SDO abstracts the data in a service-oriented way and provides a technology independent representation of the data. It enables the developers to work with the data without having to worry about the technology-specific API to access and use the data.

Business objects include several extensions to SDOs that are important for integration solutions and are used to further describe the data that is being exchanged between the SCA components.

Common Event Infrastructure

The CEI within WebSphere Process Server provides event management services. The CEI provides functionality for generation, propagation, persistence, and consumption of events that represent service component processes. Events are represented by using the Common Base Event model, which is a standard,

XML-based format that defines the structure of an event. The CEI provides standard interfaces and services for the applications to create, store, send, 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. It routes and transforms messages between the service requestor and 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 (Figure 5-6) in WebSphere Process Server addresses the transformation challenges for connecting components and the external artifacts. Mediation flows, business object maps, relationships, and selectors form the supporting services layer of WebSphere Process Server and address the transformation challenges for integrating applications.

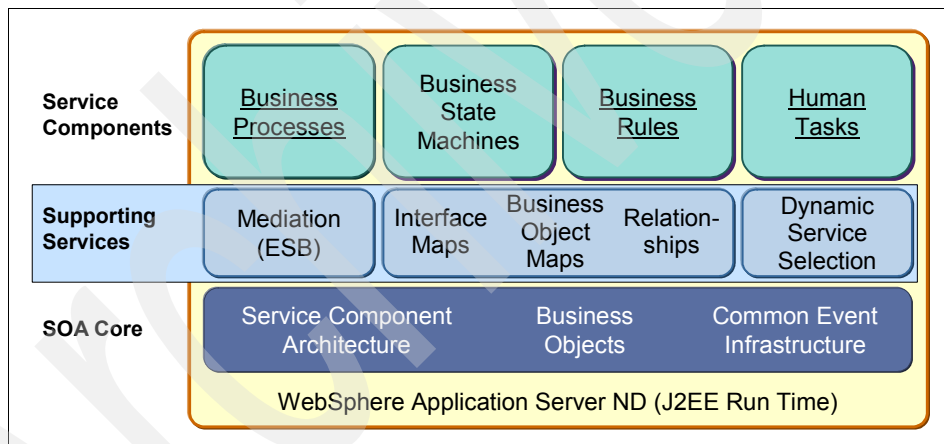


Figure 5-6 Supporting services layer of WebSphere Process Server

Mediation flows

Mediation flows are part of the WebSphere Enterprise Service Bus. They intercept and 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, routing, automatic retry, and dynamic endpoint selection.

Interface maps

An *interface* is a channel through which data is exchanged between components that run 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 support services components within WebSphere Process Server that transform one business object to another. The mapping can be a simple copy of a source attribute value to a destination attribute. Alternatively, it can 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 support services components 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 run time. A selector takes one invocation and allows for different targets to be called based on the selection criteria during run time. Selectors provide the flexibility in business processes to call new partners without having to change the design of the business process.

Service components

Service components, also called *Business Service components*, are the type of SCA components that are optimized for business for example, business processes, business rules, human tasks, and business state machines. See Figure 5-7 on page 96.

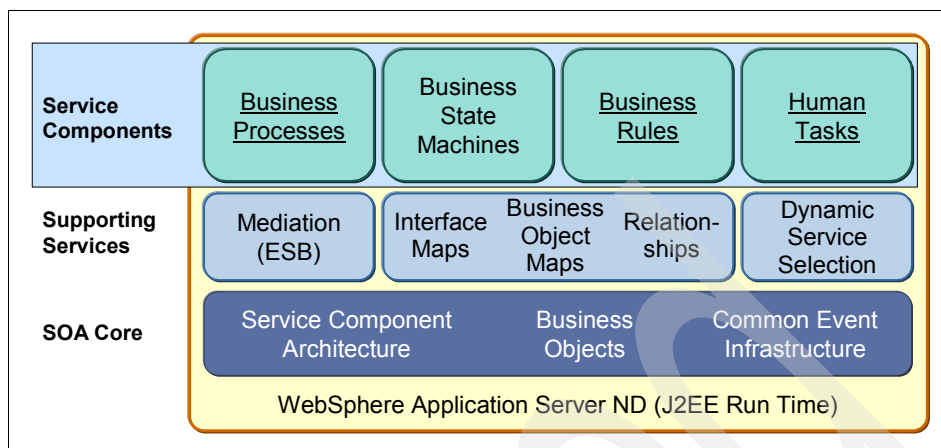


Figure 5-7 Service component layer of WebSphere Process Server

Business processes

Business processes are service components that provide the primary means through which enterprise services are integrated from the business point of view. 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.

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

Human tasks

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

Business state machines

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

Business rules

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

5.4.3 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 SOA and uses WebSphere Application Server as the underlying platform. The benefits of WebSphere Process Server include improved flexibility and scalability, as well as a streamlined development environment and a simplified operational environment.

WebSphere Process Server supports the following features:

- ▶ Business processes that are built on the SOA and that use the Web services BPEL.

The BPEL is an industry standard specification.

- ▶ The SCA, which is the invocation model, describes every integration component through an interface. The interface is completely decoupled from the underlying implementation:
 - With the SCA, a developer can concentrate on the component implementation to provide a new service or functionality, without worrying about the interaction with other components.
 - The new service components can then be easily assembled and grouped into modules, thus enabling a very flexible and encapsulated solution.
- ▶ A deployment environment for applications that is developed in WebSphere Integration Developer

WebSphere Integration Developer is a simplified development tool with visual editors for component development, assembly, integrated testing, and deployment.

- ▶ Support for human tasks, role-based task assignments, and multilevel escalations:
 - Tasks within a process can be assigned to users for manual completion.
 - Task assignments to staff can be based on the role of the staff.
 - Tasks that require multilevel approvals can be escalated.

- ▶ Business rules, business state machines, and selectors to dynamically choose interfaces based on business scenarios:
 - With the embedded business rules engine, the user can define the business rules in the process and design the process based on the outcome of the execution of business rules.
 - By using the business rules engine, the user can dynamically change the behavior of the business process without having to change the business process.
 - With a business state machine, the business process can be modelled based on states and transitions (ad hoc) rather than sequential order.
- ▶ A broad reach in integration with the support for WebSphere Adapters, Native Bindings and WebSphere Business Integration Adapters

A number of WebSphere and WebSphere Business Integration Adapters are available off the shelf to connect to various EISs.
- ▶ The managing and monitoring of events by using the CEI
- ▶ Service governance with dynamic runtime lookup and invocation of services if used in conjunction with WebSphere Service Registry and Repository
- ▶ The ability to change business processes with minimal programming skills, without redeploying the application
- ▶ Dynamic endpoint administration that allows administrators to react to changing business needs by enabling the reconfiguration of service endpoints
- ▶ Integration with Tivoli monitoring functionality, including Performance Monitoring Infrastructure (PMI), Application Response Measurement (ARM), and support for the Tivoli Composite Application Manager suite of products
- ▶ Integrated ESB with prebuilt mediation primitives that increase technical flexibility and responsiveness and enhance usability, saving time and development costs
- ▶ Architecture built over WebSphere Application Server, Network Deployment and uses the transaction, security, clustering, and workload management capabilities of WebSphere Application Server
- ▶ Advanced modeling and monitoring of business processes if used in conjunction with WebSphere Modeler and WebSphere Monitor:
 - With WebSphere Modeler, business analysts can develop the process models along with business measures.
 - WebSphere Monitor can be used to monitor the events in the running processes.

5.5 WebSphere Adapters

WebSphere Adapters are compliant with Java EE Connector Architecture (JCA) Version 1.5. JCA is an open standard and a part of the J2EE specification, which handles EIS connectivity. JCA provides a managed framework. That is, quality of service (QoS) is provided by the application server, which offers life-cycle management and security to transactions.

WebSphere Adapters (Figure 5-8) can be imported in the WebSphere Process Server run time through a resource adapter archive (RAR) file. Resource adapters in WebSphere Integration Developer are used within the context of an import or an export. You develop an import or an export with the external service wizard and, in developing it, include the resource adapter.

An EIS import or EIS export is created with a particular resource adapter. By using an EIS import, your application can invoke a service on an EIS system. By using an EIS export, your application on an EIS system can invoke a service that is developed in WebSphere Integration Developer.

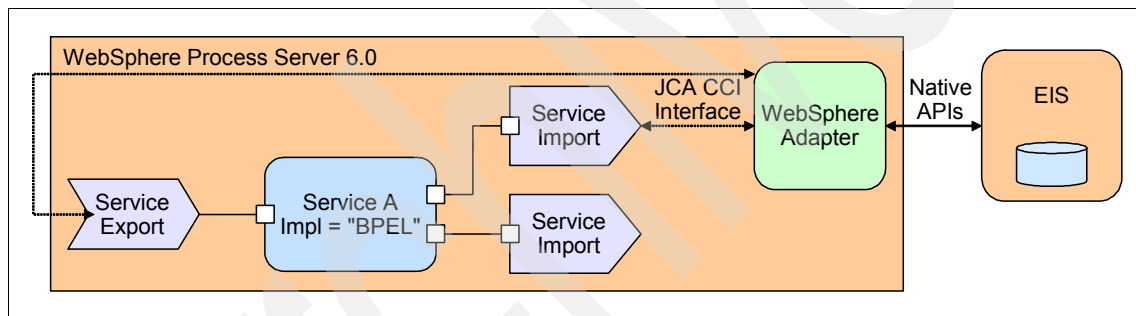


Figure 5-8 WebSphere Adapters

The following IBM WebSphere Adapters are available for WebSphere Process Server:

- ▶ IBM WebSphere Adapter for E-mail
- ▶ IBM WebSphere Adapter for Flat Files
- ▶ IBM WebSphere Adapter for FTP
- ▶ IBM WebSphere Adapter for JDBC
- ▶ IBM WebSphere Adapter for PeopleSoft Enterprise
- ▶ IBM WebSphere Adapter for SAP Software
- ▶ IBM WebSphere Adapter for J.D. Edwards EnterpriseOne
- ▶ IBM WebSphere Adapter for Oracle® E-Business Suite
- ▶ IBM WebSphere Adapter for Siebel® Business Applications

To illustrate the typical capabilities of the application adapters and technical adapters, we provide an overview of the WebSphere Adapter for SAP and the WebSphere Adapter for Flat Files in the following sections.

5.5.1 WebSphere Adapter for SAP

The WebSphere Adapter for SAP software is an application adapter that connects to SAP systems, by using the SAP Java interface called *SAP Java Connector (JCo)*. It does so by making calls, modeled as business objects, to the SAP native interfaces and passing data to and from the SAP system. The adapter supports SAP integration interfaces, such as BAPI and ALE, as well as SAP RFC function modules.

The adapter supports outbound processing (from the adapter to the SAP system) and inbound processing (from the SAP system to the adapter) of events.

Outbound event processing

With outbound support, a client can make calls to the adapter to perform a specific operation in the SAP system. The client requests a connection, which in turn is passed from the adapter to SAP.

Outbound event processing, which the adapter supports for the BAPI and ALE interfaces, consists of the following actions:

1. An SCA component invokes an interaction with SAP.
2. As a result of the invoked interaction, a business object that represents the SAP function call is passed from the component application to the adapter.
3. The adapter extracts the elements from the business object, and by using the metadata information from the business object, recognizes the SAP interface to use (BAPI or ALE).
4. By using the SAP JCo, the adapter converts the business object data to the appropriate SAP function call.
5. The adapter executes the function on the destination SAP software system, sending the event data to SAP.

Inbound event processing

Inbound event processing, which the adapter supports for the ALE interface, consists of the following actions:

1. The adapter spawns listener threads, acting as an RFC Server.
2. Whenever an event occurs in SAP, the event is pushed to the adapter through the event listeners.

3. The adapter can track and recover events in case of an abrupt termination by using the data source to persist the event state in an event recovery table. The adapter supports container-managed sign-on and basic authentication.

5.5.2 WebSphere Adapter for Flat Files

WebSphere Adapter for Flat Files enables integration with applications that provide a basic flat file interface. This adapter can be used with applications that do not provide programmable or service interfaces, and for which integration at the data tier is impractical.

This JCA resource adapter facilitates the bidirectional exchange of any type of text or binary file and provides the following capabilities:

- ▶ Parses and serializes native data formats to and from SDO
- ▶ Writes to new files
- ▶ Appends to or overwrites existing files
- ▶ Polls a directory and its subdirectories for new files to deliver to the server
- ▶ Lists, retrieves, and deletes target files in a directory

5.6 WebSphere Process Server bindings

An SCA module can have one or more import or export components as shown in Figure 5-9 on page 102. An *import* enables a component reference to invoke a remote service, while an *export* exposes a component service outside the current module.

Each import and export contains a binding definition. A binding describes a protocol for calling a piece of code. The binding then determines how the import or export interacts with clients outside the module where the import and export reside.

Bindings for imports and exports have different purposes. An *export binding* describes how that export (or service) is published or made available to clients outside the module. The kind of binding determines the kind of client that is supported. For example, a Web service binding makes the service available to any Web Service-based client, while an SCA binding makes it available to other SCA modules.

Import bindings tell the SCA runtime processes how to access an external service. For example, if a service is published with an SCA export binding, an import with a JMS binding is unable to successfully call it.

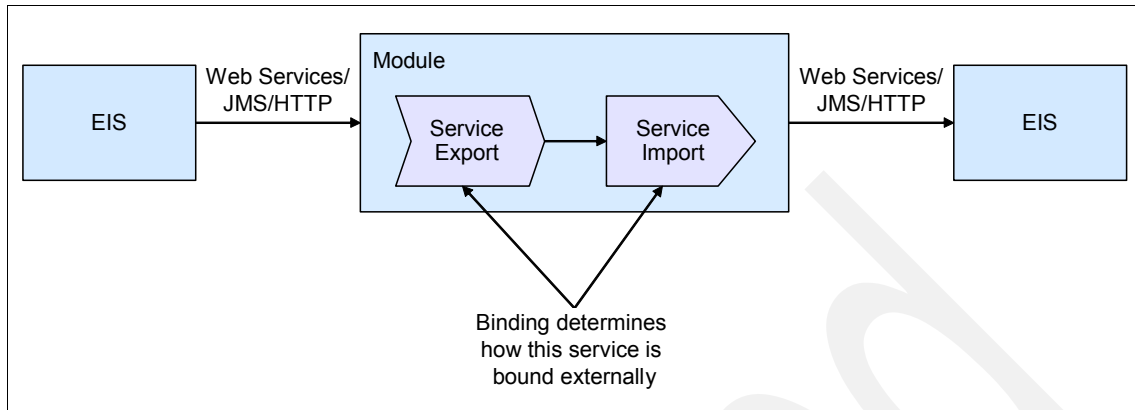


Figure 5-9 WebSphere Process Server Binding examples

In the following sections, we provide an overview of the types of bindings. See the WebSphere Integration Developer information center for more information at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.basics.doc/topics/cbindings.html>

5.6.1 EIS binding

SCA EIS bindings provide connectivity between SCA components and external systems. This communication is mediated by EIS exports and EIS imports.

Your SCA components might require that data be transferred to or from an external EIS. When you create SCA modules that require such connectivity, you must include (in addition to your SCA component) EIS imports, EIS exports, or both for communication with specific external EISs.

EIS bindings are used mostly to interact with JCA adapters (WebSphere Adapters).

5.6.2 EJB binding

An *EJB binding* is also known as a *stateless session bean binding*. WebSphere Process Server supports only outbound invocation of EJB services. An EJB import binding facilitates the communication between an SCA service and a remote stateless session bean.

5.6.3 Web service binding

A *Web service binding* facilitates the communication between an SCA service and an external Web service. A Web service export exposes an SCA service as a publicly accessible Web service. A Web service import enables a component reference to invoke a remote Web service. In Web service invocations, because the native message data format is always in the SOAP format, data conversion is performed in a standard way without any pluggable data bindings.

5.6.4 HTTP binding

The HTTP binding provides connectivity to HTTP. This connectivity allows existing or newly developed HTTP applications to participate in SOA environments. The format handling is taken care of by pluggable data bindings.

5.6.5 JMS binding

A JMS binding allows JMS interactions by using the WebSphere Application Server default messaging provider, which is based on the Service integration bus. The format handling is taken care of by pluggable data bindings.

5.6.6 Generic JMS binding

The generic JMS import and export bindings provide connectivity to third-party JMS 1.1-compliant providers. The operation is similar to that of JMS bindings. The service provided through a generic JMS binding allows an SCA module to make calls or receive messages from external JMS systems. The format handling is taken care of by pluggable data bindings.

5.6.7 WebSphere MQ JMS binding

When WebSphere MQ is acting as a JMS provider, WebSphere MQ JMS bindings allow interoperation with J2EE applications that are deployed to WebSphere MQ. WebSphere MQ JMS bindings provide a framework for J2EE and SOA applications communicating through WebSphere MQ. WebSphere MQ JMS bindings expose the same model as other JMS bindings. The format handling is taken care of by pluggable data bindings.

5.6.8 WebSphere MQ binding

WebSphere MQ bindings expose a model that is more familiar to a WebSphere MQ audience. WebSphere MQ bindings allow interoperability with native WebSphere MQ or WebSphere Message Broker applications. In doing so, WebSphere MQ bindings bring these applications into the SOA framework by providing mediation between, and communication with, them. The specific MQ headers, such as RFH, are taken care of by the MQ binding and a specialized mediation primitive. The format handling is taken care of by pluggable data bindings. When installing an SCA module that contains WebSphere MQ bindings, the run time is automatically configured to allow connectivity with WebSphere MQ. However, the MQ systems administrator must configure the WebSphere MQ queue managers, because these WebSphere MQ queue managers are not part of the WebSphere Process Server solution and therefore are not automatically configured.

5.7 WebSphere Process Server data bindings

Data bindings handle the transformation of data that is passed as an SDO in an SCA-based application and the native format for an EIS J2C adapter, a HTTP application, or a messaging JMS system. The data binding function handles the request and response arguments for both inbound and outbound communication.

When an EIS, HTTP, or messaging import is invoked, a data binding is used to transform the content of an SDO into the native format of the EIS, HTTP, or messaging system. When the reply (if any) is received from the EIS, HTTP, or messaging import, the data binding is used to transform the native data format into an SDO.

Similarly, when an EIS, HTTP, or messaging export is invoked, a data binding is used to transform the native data format into an SDO. When a reply (if any) is sent back to the caller, the data binding is used to transform the content of the SDO into the native format of the EIS, HTTP, or messaging system.

Figure 5-10 shows a schematic representation of the a WebSphere Process Server data binding when communicating with an EIS, which is also perfectly valid for HTTP and Messaging Imports/Exports.

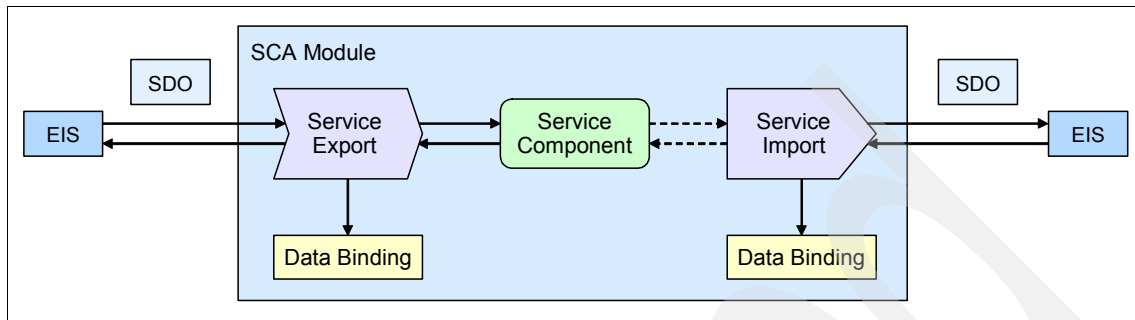


Figure 5-10 Diagram of WebSphere Process Server data binding

In the following sections, we look at the various types of data bindings.

Note: The information in the following sections is relevant to WebSphere Process Server Version 6.2. It does not apply completely to former versions of WebSphere Process Server.

5.7.1 EIS data bindings

Data bindings allow EIS import or EIS export implementation to convert argument data to the native format that is expected by the J2C. You can define the data binding implementation class or classes in the import or export files. *DataBinding* is the root data binding interface.

For example, the following prepackaged databindings and data format transformations are available for the E-mail and Flat File adapters:

- ▶ For EMail adapter:
 - Simple databinding
 - Fixed Structure databinding
 - Wrapper databinding
- ▶ For Flat File adapter:
 - Delimited
 - Fixed Width
 - WTX
 - XML
 - JSON
 - Java Architecture for XML Binding (JAXB)

More information can be found in each adapter's documentation.

It is also possible to write a custom data binding (based on a `RecordDataBinding`, for example, with J2C) or to configure a data binding to take care of specific formats through a data handler. We describe these custom options in 5.7.5, "Custom data bindings and data handlers" on page 108.

5.7.2 JMS data bindings

A JMS data binding converts data between the JMS native message format and the SDO format that is used by SCA. There are built-in JMS data binding implementations for each of the JMS message formats. JMS data bindings are used by the JMS, generic JMS, and the WebSphere MQ JMS import/export bindings.

For example, the following data bindings and data format transformations are predefined for JMS bindings:

- ▶ Serialized Java Object
- ▶ Wrapped bytes
- ▶ Wrapped stream message
- ▶ Wrapped map entry
- ▶ Wrapped object
- ▶ Wrapped text
- ▶ Delimited
- ▶ FixedWidth
- ▶ XML
- ▶ SOAP
- ▶ JSON
- ▶ JAXB
- ▶ Atom
- ▶ WTX

Alternatively, users can implement a custom data binding that is based on the `JMSDataBinding` interface or custom datahandlers.

See the following link for comprehensive information about the JMS data bindings:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/concepts/cwesb_jmscustombindings.html

5.7.3 WebSphere MQ data bindings

A WebSphere MQ data binding converts data between the MQ native message format and the SDO format that is used by SCA. WebSphere MQ data bindings are used by the WebSphere import/export bindings. There are built-in WebSphere MQ data bindings and data format transformations for different MQ message body types:

- ▶ Delimited
- ▶ Fixed Width
- ▶ MQ serialized business object XML
- ▶ MQ serialized Java object
- ▶ MQ unstructured binary message
- ▶ MQ unstructured text message
- ▶ XML
- ▶ JSON data binding for MQ
- ▶ JAXB
- ▶ Atom
- ▶ WTX data binding

Alternatively, users can implement custom data bindings that are based on the `MQHeaderDataBinding` and `MQBodyDataBinding` interfaces or custom datahandlers.

See the following link for comprehensive information about the MQ data bindings:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.messaging.doc/topics/cmqdb.html>

5.7.4 HTTP data bindings

An HTTP data binding converts data between the HTTP message and SDO format that is used by the SCA. HTTP data bindings are used by the HTTP import and export bindings.

The following types are prepackaged data format transformations for HTTP:

- ▶ Delimited
- ▶ Fixed Width
- ▶ XML
- ▶ SOAP
- ▶ JAXB
- ▶ JSON
- ▶ Atom
- ▶ WTX
- ▶ Wrapped text
- ▶ Wrapped bytes

See the following link for comprehensive information about the HTTP data binding and associated built-in data format transformations:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.http.doc/topics/chtptconfig.html>

Alternatively, users can implement specific data format transformation logic using custom datahandlers.

5.7.5 Custom data bindings and data handlers

Users can implement custom data bindings. A data handler can then be used to work with a data binding. The data handler is not the WebSphere Business Integration Adapter data handler that comes from the WebSphere InterChange Server. It is a similar concept, but it is implemented with WebSphere Process Server technology and APIs.

The data binding can delegate the transformation logic to the data handler. That is, the data binding works as a mediator between the transport-dependent protocol and the transport-independent protocol of the data handler. The following data handlers are predefined, but you can also develop custom data handlers:

- ▶ Delimited data handler
- ▶ FixedWidth data handler
- ▶ XML data handler
- ▶ SOAP data handler
- ▶ JSON data handler
- ▶ Atom data handler
- ▶ WTX data handler

See the following link for comprehensive information about the WebSphere Process Server data handlers:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.config.doc/topics/createdh.html>

Product feature comparison

Conceptually, WebSphere InterChange Server features supported by components, such as adapters, business objects, maps, and relationships, are available in WebSphere Process Server as well. In this way, WebSphere Process Server extends the functionality of WebSphere InterChange Server.

Although, other features of these two products, including the runtime architecture, component granularity, and quality of service (QoS), differ greatly. WebSphere Process Server provides a service-oriented architecture (SOA) run time and addresses and concepts, such as Service Component Architecture (SCA) and Business Process Execution Language (BPEL).

In this section, we compare several of the features provided by the WebSphere InterChange Server and WebSphere Process Server products.

We include the following sections:

- ▶ 6.1, “Architecture and design aspects” on page 110
- ▶ 6.2, “Development and testing” on page 111
- ▶ 6.3, “Deployment” on page 112
- ▶ 6.4, “Run time” on page 113
- ▶ 6.5, “Feature comparison summary” on page 119

6.1 Architecture and design aspects

WebSphere Process Server architecture is an extension of the WebSphere InterChange Server base. But, it has several key differences and add-ons, making WebSphere Process Server an overall better architecture-based product.

6.1.1 Server architecture

The WebSphere InterChange Server is based on the standards and programming model of the Java Platform Standard Edition (J2SE) technologies. Integration solutions are organized into components, such as collaborations, adapters, business objects, maps, and relationships, which are organized together into collaboration objects, each of which manages the execution required for one or a specific part of an integration usage scenario.

WebSphere Process Server has support for an extremely powerful Java 2 Platform Enterprise Edition (J2EE) environment. WebSphere Process Server is implemented using the J2EE programming model and is supplied as a collection of features that execute on the class-leading WebSphere Application Server software platform. Integration solutions are composed of service components, such as processes, adapters, interfaces, business objects, interface maps, data maps, and relationships, which are organized into modules, each of which is built for one or a specific part of an integration usage scenario.

6.1.2 Programming model

The significant change of platform from J2SE to J2EE brings a significant change to the programming model used to implement integration solutions. The interaction between components in WebSphere InterChange Server takes place through the exchange of events or requests performed either over MQ messaging or over a synchronous distributed request protocol, such as Internet Inter-Object Request Broker (ORB) Protocol (IIOP). Each component's *interface definition*, that is, the set of message or request types that are acceptable at that component, is built into the component's definition in a proprietary manner.

In WebSphere Process Server, all components are defined using the Service Component Architecture meta-model, which is a formal technique that is currently being standardized by the Apache Tuscany project. Each component exposes an abstract interface, which can be defined either as a Java Interface or as a Web Services Description Language (WSDL) port type. Components are accessed via the operations defined on these interfaces. The number of types of interaction, the data types exchanged, and the interaction style involved (one-way, request-response, and so forth) are wholly defined in the set of

operations and their signatures. The physical data formats and transport and access protocols are determined through binding specifications.

WebSphere Process Server has better logical separation of concerns, with dedicated and specialized components, which enables the user to better separate business and integration logic.

6.2 Development and testing

The WebSphere Process Server development environment relies on the same principles as the WebSphere InterChange Server environment:

- ▶ Build-time environment tooling that is based on Eclipse
- ▶ Embedded test server

WebSphere Process Server brings valuable enhancements, especially regarding source control and testing.

6.2.1 Source control

In WebSphere InterChange Server, the source control system is not tightly connected to the WebSphere InterChange Server tooling during development. It requires separate source control of the WebSphere InterChange Server artifacts.

WebSphere Process Server tooling has support for several source control systems. WebSphere Integration Developer is based on the Eclipse platform and the WebSphere Integration Developer artifacts can be used with a source control (Concurrent Versions System (CVS) or Rational ClearCase) at the component level.

6.2.2 Testing

WebSphere InterChange Server provides tools, such as the Test Connector, to speed up the testing process of an interface. But, it lacks other more elaborate testing features around data pooling and regression testing. Almost always, testing within the WebSphere InterChange Server environment remains a manual process.

WebSphere Process Server has better facilities for build and test automation. It has much better management and sharing of test data, has the ability to test any scope of the components (individual or multiple modules) and the ability to perform automated regression testing. The data can be saved in a data pool within WebSphere Integration Developer (WID). It also provides the ability to

share the data pool for input and output by saving the data in a file in a directory for the component test. This approach accelerates testing by being able to share test data in multiple tests and automatically compare results. The Test Client within WID has the ability to import test data from a file, XML, or an HTTP link. For Web Services testing, a SOAP message can be imported or stripped out of the http message. You can create the test values using the XML Editor so that you can edit the SOAP header and body and import data in any format. A full XML Editor is included in the test client, which allows you to Format, Find/Replace, Copy/Paste, and perform Syntax validation.

6.3 Deployment

WebSphere Process Server and WebSphere InterChange Server differ greatly regarding the deployment model due to the underlying infrastructure on which they rely: J2SE for WebSphere InterChange Server and J2EE for WebSphere Process Server. We outline the major differences in the following sections.

6.3.1 Packaging and deployment model

In WebSphere InterChange Server, you create integration components in a library in your local file system and deploy them to a WebSphere InterChange Server instance to make them executable. When components are deployed to the server, the server run time is updated so that they can be used immediately. You can use either the System Manager graphical interface or the **repos_copy** command-line interface to deploy a package of integration components. To use the **repos_copy** command-line interface for this task, you must first create a packaged JAR file.

In WebSphere Process Server, there are flexible ways of packaging and deploying the modules. You can design monolithic modules or decide to have a finer-grained design by separating business logic from integration logic, for example. After your modules are designed, they become Enterprise Application Archives (EARs) to be deployed into deployment targets. A deployment target can be a server or a cluster.

6.3.2 Inter-environment deployment

WebSphere InterChange Server provides a feature called *Deployment Descriptor*. In WebSphere InterChange Server, deployment time properties are those properties of an artifact that are configured at the time of deployment, rather than at design time. System Manager allows you to specify deployment time properties specific to a server. When an artifact is deployed to a server,

System Manager applies the deployment time properties for that artifact before deploying it. You can also copy the properties of an artifact to more than one server, which keeps you from having to keep track of all the properties of the various components for the separate servers.

There is no direct equivalent feature in WebSphere Process Server. However, using scripting, you can use environment variables and promotable properties to reach an equivalent level of service for inter-environment deployment automation needs.

6.3.3 Versioning

Versioning support does not exist in WebSphere InterChange Server, whereas WebSphere Process Server supports versioning at several levels, including Modules and Service Component Architecture (SCA), Business Process Execution Language (BPEL), and Human Tasks.

In WebSphere Process Server, you can specify the Module versioning using the version scheme set in the dependencies editor for the SCA modules and libraries so that, at run time, multiple versions of SCA modules can exist. You can use the version number to differentiate the components so that you can use a specified version number for SCA imports and Endpoint Lookup Primitive Library references. You can develop and deploy one or more versions of a module and its artifacts into the runtime environment for use by specific clients.

Just as a module can have a version number, so can the SCA import and export bindings in a module. SCA bindings inherit their version information from the module with which they are associated. Although, SCA bindings are the only binding type that can be versioned. Libraries can also be versioned. Modules that use a library have a dependency on a specific version of that library, and libraries can also have dependencies on specific versions of other libraries.

WebSphere Process Server also supports versioning of Human Tasks and Business Processes based on the valid-from date.

6.4 Run time

WebSphere Process Server and WebSphere InterChange Server differ greatly regarding the runtime model mainly due to the underlying infrastructure on which they rely: J2SE for WebSphere InterChange Server and J2EE for WebSphere Process Server. We outline the major differences in the following sections.

6.4.1 High availability and clustering

To satisfy a service level agreement (SLA) for a service, availability needs to be built in while designing a solution. High availability (HA) is used to describe a system that can detect a failure of any component of the system and to react to it automatically within a matter of a few minutes. To the user, a system with high availability will continue to operate, even after a brief delay.

A *cluster* is a grouping of two or more interconnected systems that are viewed and used as a single system resource. Clustering solutions provide the prevalent mechanism to achieve high levels of availability.

The availability of the WebSphere InterChange Server system is built on an external hardware and software clustering solution. For WebSphere InterChange Server, HA is available on the following operating systems:

- ▶ For UNIX®-based systems: Sample scripts with a readme file are provided as part of an IBM SupportPac, for example, IBM High Availability Cluster Multi-Processing for AIX® (HACMP™).
- ▶ Windows 2000: Refer to *The System Installation Guide for Windows*, which is available at this Web site:

ftp://ftp.software.ibm.com/software/websphere/integration/wicserver/library/doc/wics43/pdf/installation_windows_30sept04.pdf

This guide provides a set of instructions to set up HA using Microsoft Cluster Server (MSCS) software. An IBM Category 2 SupportPac provides dynamic link library (.dll) files for use with MSCS.

All clustering approaches provide various mechanisms to define resources that are going to be managed by the HA clustering software. For the remote environment, you can create an exact replica in the hardware and software of the original production environment. All releases of software to the production environment can also be performed on the remote environment. In the event of failover to the remote server, the status of the production WebSphere InterChange Server environment's transactional data must be synchronized with the remote system. Also, a backup strategy needs to be in place for backing up the WebSphere InterChange Server repository and queues (including adapter queues).

The WebSphere InterChange Server HA solution is a somewhat complex solution and requires additional steps outside of the WebSphere InterChange Server stack to achieve true HA service.

WebSphere Process Server provides many ways to use clustering techniques to achieve availability. WebSphere Process Server includes innate HA based on the capabilities of the underlying WebSphere Application Server server. It also

allows several clustering topologies to support HA configuration, such as Golden Topology.

A *WebSphere Process Server cluster* is a logical collection of WebSphere Process Servers configured to perform the same task. The member servers can be distributed across one or more nodes in any configuration. In a WebSphere Process Server Network Deployment (ND) environment, you can cluster your applications so that the applications are protected from the failure of a single server for HA or distribute the workload of the applications across a number of equivalent servers for work load balancing. It supports both vertical and horizontal clustering, with a deployment manager for the management and configuration. In a vertical cluster, multiple application servers are placed onto the same node in order to better utilize the available resources. In a horizontal cluster, multiple application servers are distributed across nodes in order to utilize more physical resource.

Overall, WebSphere Process Server offers a solid base for clients to build a scalable and highly available application environment.

6.4.2 Transactionality

In WebSphere InterChange Server, collaboration can be made transactional so that the data modifications can be rolled back. Like database transactions, transactional collaborations are all-or-nothing operations: Either the entire collaboration succeeds or the entire collaboration fails. In addition, a transactional collaboration can detect and react to data isolation violations that might compromise the logic of its data operations. Transactional collaborations are based on the principles established in database transactions and two-phase commit protocols. A transactional collaboration executes under the control of the WebSphere InterChange Server's transaction service, which controls execution, rollback, and isolation checking.

Collaboration's transaction level determines the mechanisms by which the system executes the scenarios in the collaboration. A collaboration template's Minimum Transaction Level property determines whether the collaboration is transactional, and applies to all of its scenarios. All collaboration objects created from a template inherit its minimum transaction level.

Collaboration becomes transactional at the development phase, when its developer determines whether the collaboration requires transactional execution. If it does, the developer adds compensation steps to the scenarios in the template and specifies the collaboration's transaction level. Every collaboration has one of the following transaction levels: None, Minimal Effort, Best Effort, or Stringent.

As specified in the migration notes, there is no direct mapping of the levels of transaction between WebSphere InterChange Server collaborations and WebSphere Process Server BPEL. Therefore, during the migration, the transaction level specified in the WebSphere InterChange Server collaboration is ignored and the default BPEL transaction level will be used in the migrated application. You need to understand BPEL transactions and adjust your migrated applications accordingly to get the desired functionality.

6.4.3 Compensation

A WebSphere InterChange Server's transactional collaboration rolls back if it encounters an error that stops its execution. For rollback to be successful, each scenario must have compensation specified for each sub-transaction step. The transactional collaboration uses the compensation defined for its sub-transactions or Service calls to perform the actual rollback. A *Service call* is an operation in which the collaboration sends a request that causes a transactional data change in an application data store.

Compensation is a logical undo action. That is, if execution of a collaboration object fails; it causes rollback of the previously executed operations. When rollback occurs, the collaboration runtime environment steps backward through the path of execution, executing a compensation step for every normal step that has already executed and that has compensation defined. Rollback thereby logically returns data to the state that it was in before the transactional collaboration started to execute.

In WebSphere InterChange Server, compensation is supported for both synchronous service calls and asynchronous outbound service calls. If the collaboration is transactional, and if the verb for this service call requests a data modification, you can specify the compensation operation that rolls back the regular service call. You can define the compensation by clicking the Compensation property check box.

Compensation in WebSphere Process Server differs from the compensation techniques that are used in WebSphere InterChange Server. In WebSphere Process Server, there are two ways in which you can compensate a business process. The two options are compensation pairs and compensation handler:

- Compensation pairs

Compensation pairs are the original properties of each of the individual parts of a business process. These properties are saved so that they can be restored if the process cannot be committed and must be rolled back. The original status of the activity is stored in an operation and its value in a variable.

- Compensation handler

A *compensation handler* is a series of isolated activities that are associated with an activity within a business process. The activities within a compensation handler will only run when a fault is thrown, and after the parent activity has already been committed. The goal of a compensation handler is to return a failed process to a balanced state.

After migration, evaluate the new types of compensation offered by WebSphere Process Server and choose the type that best suits your application.

6.4.4 Event Sequencing

In WebSphere InterChange Server, *Event Sequencing* is an assurance that the server provides that collaboration processes multiple events that relate to the same business object one at a time and in the same order in which the events arrive. If multiple events have the same business object type and key values, the server queues them and delivers them in order of arrival. Only events with the same key values are sequenced. Events with different keys continue to be processed in parallel.

Due to the inherent support for Event Sequencing in WebSphere InterChange Server, extremely little administrative work is required on the part of the collaboration template developer to take advantage of the Event Sequencing feature. The minimal work that is required involves toggling the `ControllerEventSequencing` property of a given connector to true or false.

WebSphere Process Server implements Event Sequencing in a finer grained way. Instead of delegating the function of Event Sequencing to the server globally, the components within a module can be set up individually to enforce Event Sequencing. Event Sequencing is supported only for components that are invoked using the asynchronous invocation style. In a Network Deployment (ND) environment, Event Sequencing requires that there is only one active messaging engine per cluster. For an ND environment with servers that are not in a cluster, Event Sequencing requires that each server has an active messaging engine.

6.4.5 Flow control

WebSphere Process Server currently lacks direct equivalence for several of the finer grained features controlling critical aspects of non-functional execution behavior. One of the major examples is *flow control*, which allows WebSphere InterChange Server to work in a store and forward mode. Flow control is only supported in WebSphere Process Server as an additional design feature of the interface.

6.4.6 Security

The security of messages and business data on a system is critical. In WebSphere InterChange Server, building a secure system is based on the system infrastructure security configurations. Securing data in any system includes three security services: authentication, message integrity, and privacy. *Authentication* involves verifying that someone is who they claim to be. Authentication governs access control, which restricts access to parts of the system based on who the authenticated user is and what permissions the user has been granted. *Message integrity* ensures that the data has not been modified. *Privacy* ensures that only authorized users can view data. Integrity, privacy and authentication can be accomplished with encryption, while Role Based Access Control (RBAC) can be accomplished by setting up user IDs and passwords.

For data security, messages can be encrypted in the message layer. In WebSphere InterChange Server, data is secured from the moment it leaves a source adapter, through its journey into the WebSphere InterChange Server, right up until it reaches a destination adapter. Critical to any secure system is endpoint verification. In order to use end-to-end privacy to protect your messages, you must activate it in the appropriate configuration file. End-to-end privacy can be turned on or off for each adapter.

Administrative security is provided by the supported RBAC authentication capability. This feature provides the ability to authorize permissions for users accessing the system using roles. Roles can easily be defined by the Administrator and assigned to a group of users, restricting access to key components only to verified users. Roles can be assigned along functional associations, which greatly reduces the administrative burden. Assigning a role to a user or users allows them to access only the components of the system that are included in the role definition.

WebSphere Process Server Security is based on the underlying powerful WebSphere Application Server Security architecture. The J2EE layer security provides role-based access control for applications. WebSphere Application Server provides a security infrastructure and mechanisms that protect sensitive J2EE resources and administrative resources. WebSphere Application Server security is broken down into three components:

- ▶ Application security
- ▶ Administrative security
- ▶ Java 2 security

Application security provides application isolation and requirements for authenticating users and controlling their access to the applications in the environment. It provides authentication and access control with a single sign-on

(SSO) solution. Authentication includes HTTP Authentication, Web services Security (WS-Security), Java Authentication and Authorization Service (JAAS), and Secure Sockets Layer (SSL) client authentication.

Administrative security represents the security configuration that affects the entire security domain. The security domain consists of all the servers that are configured with the same user registry realm name. The basic requirement for a security domain is that the access ID returned by the registry from one server is the same access ID as that returned from the registry on any other servers within the same security domain.

Java 2 security provides a policy-based, fine-grained access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources. Java 2 security guards access to system resources, such as file I/O, sockets, and properties. J2EE security guards access to Web resources, such as servlets, JavaServer™ Pages (JSP™) files, and Enterprise JavaBeans (EJB) methods.

6.5 Feature comparison summary

We provide a list of comparative elements regarding product features in three cases:

- ▶ WebSphere Process Server provides equivalent capability as WebSphere InterChange Server
- ▶ WebSphere Process Server provides better capability than WebSphere InterChange Server
- ▶ WebSphere Process Server does not provide equivalent capability as WebSphere InterChange Server

6.5.1 Functional gaps closed with WebSphere Process Server 6.2

These gaps have been closed:

- ▶ Direct migration to JCA/Native binding counterparts is available for some WebSphere Business Integrator Adapters.
- ▶ All technical WebSphere Business Integrator Adapters are supported through standard migration tooling in WID.
- ▶ WebSphere Business Integrator Data Handler migration is supported through the Heritage API.
- ▶ Basic WebSphere Business Integrator Data Handlers have WebSphere Process Server counterparts (Fixed Width and Delimited).

- ▶ WebSphere Process Server Native support for COBOL copy books (eliminated the need for third-party mapping, that is, Complex Data Handler)
- ▶ Event Sequencing at the edge (Java Message Service (JMS) export)
- ▶ Integration Logic at the Edge (WESB Mediation support for Gateway pattern, Service Invoke, Retry, and Fan-out/Fan-in)
- ▶ Auto-mapping and Reverse Mapping in BO Maps

6.5.2 WebSphere Process Server features superior to WebSphere InterChange Server features

These WebSphere Process Server features are superior:

- ▶ WebSphere Process Server provides superior architecture features:
 - Innate HA
 - Better separation of concerns
 - Very powerful J2EE environment
- ▶ Better Web Service support
- ▶ Better messaging (JMS/MQ) integration
- ▶ Adapters much more efficient (less memory) and more manageable
- ▶ More automatable administration with scripting
- ▶ More automatable procedures for build/test/deploy
- ▶ Programming model (and tools) are finer grained
- ▶ Strategic product, which covers both Integration and BPM spaces

6.5.3 WebSphere InterChange Server features still superior to WebSphere Process Server features

These WebSphere Interchange™ Server features are still superior:

- ▶ There is no equivalent to WBIA remote connector agent capability with JCA adapters.
- ▶ WebSphere Process Server lacks flow control:
 - Controller Store and Forward (for J2C Adapters)
 - Poll period and quantity for bindings
 - Overall flow control (thresholds)
- ▶ WebSphere Process Server lacks certain administration capabilities.
 - No RBAC at the component level.

- ▶ Administration capabilities in WebSphere Process Server are coarse-grained
The WebSphere Process Server administrative console does not provide a solution view today (modules aggregated as end-to-end flows).

Archived

Server component upgrade

In this chapter, we focus on the upgrade path for the server artifacts, from WebSphere InterChange Server to WebSphere Process Server. First, we review the major technical characteristics of the server migration. Then, for each WebSphere InterChange Server artifact type, we explain the equivalent concept in WebSphere Process Server. We conclude with the major benefits and limitations of the upgrade.

This chapter is complemented with Chapter 8, “Adapters and data handlers upgrade” on page 131, which focuses on the adapters and data handlers. Chapter 9, “Migration implementation approach” on page 151 progresses to the solution perspective and analyzes the possible migration paths.

We include the following sections:

- ▶ 7.1, “Overview of the upgrade” on page 124
- ▶ 7.2, “Component mapping” on page 125
- ▶ 7.3, “Benefits” on page 128
- ▶ 7.4, “Limitations” on page 129

7.1 Overview of the upgrade

WebSphere InterChange Server and WebSphere Process Server are separate products that induce certain constraints for the migration path. In the following sections, we describe the major factors to consider.

7.1.1 Planning

Because WebSphere Process Server brings many new concepts and covers a larger scope than Enterprise Application Integration (EAI), implement the migration through a dedicated project. Through such a project, you can assess the right scope for migration and prepare it properly from more than a technical point of view. See Chapter 4, “Migration planning” on page 33 for recommendations about project planning.

7.1.2 Server migration constraints

Because the runtime architectures for the two products differ, there is no “in place” migration of a given WebSphere InterChange Server running environment. Therefore, the following constraints exist:

- ▶ No standard migration support exists for WebSphere InterChange Server runtime data, such as unresolved flows and long-lived business processes (LLBPs). The only supported way to handle runtime data is to let the corresponding flows terminate in the WebSphere InterChange Server environment.
- ▶ No standard migration support exists for the WebSphere InterChange Server system configuration. The only supported way to handle system configuration migration is to redo the configuration steps in the new WebSphere Process Server environment, for example, server tuning parameters, server security settings, and so on.
- ▶ No standard migration support exists for WebSphere InterChange Server monitoring, scripting, and deployment procedures.
- ▶ For the server artifacts, the supported migration path is to upgrade them in the new build time and then deploy them into the new run time. The WebSphere InterChange Server artifacts cannot run directly in the new run time.
- ▶ Sometimes the migrated components or their equivalent in the new run time do not provide exactly the same level of service, runtime behavior, or performance.

For those reasons, you must go through the usual project life cycle, from development to test and production. The migration life cycle requires you to migrate the artifacts, set up the new production platform and deployment procedures, and ensure that the new production environment can host the migrated applications with a satisfactory quality of service (QoS).

7.1.3 Server artifact migration

In 7.2, “Component mapping” on page 125, we describe how you can convert WebSphere InterChange Server components into equivalent WebSphere Process Server components. Be aware that an artifact perspective is not sufficient to assess the overall upgrade path.

Ensure that you read Chapter 8, “Adapters and data handlers upgrade” on page 131 and Chapter 9, “Migration implementation approach” on page 151, because this information is mandatory in order to understand the migration implementation path.

7.2 Component mapping

There is an architectural difference between WebSphere InterChange Server and WebSphere Process Server. WebSphere Process Server is built by keeping the best features of WebSphere InterChange Server and by including industry standards, such as service-oriented architecture (SOA) and Business Process Execution Language (BPEL).

Certain concepts of the WebSphere InterChange Server have straightforward equivalents in the WebSphere Process Server, but others do not. In Table 7-1 on page 126 and Table 7-2 on page 127, we list the artifacts in WebSphere InterChange Server, the possible equivalents in WebSphere Process Server, and what is supported by the standard migration tools.

For more information: For information about the standard migration tools, see Chapter 10, “Overview of migration tools” on page 179.

Table 7-1 Mapping the most usual WebSphere InterChange Server components

WebSphere InterChange Server artifact	WebSphere Process Server artifact or concept	Support from the migration tools for WebSphere Process Server V6.2
Business objects	Business objects or Service Data Objects (SDOs)	The migration tool generates two SDOs (BusinessGraph and BusinessObject) per WebSphere InterChange Server business object.
Maps	Business object maps	The migration tool generates two business object maps per WebSphere InterChange Server map.
Relationships	Relationships	The migration tool generates relationship definitions and a Jython script to reuse existing WebSphere InterChange Server relationships.
Collaboration templates	BPEL processes Mediation flow components	The migration tool generates BPEL.
Collaboration objects	No directly equivalent feature	The migration tool creates a process module that contains the BPEL and other Service Component Architecture (SCA) components, such as interface maps, and SCA wiring.
Collaboration object group	No directly equivalent feature	Partial. The migration tool generates one module per collaboration. Modules are to be wired together manually.
WebSphere Business Integration Adapter definition	J2EE Connector Architecture (J2C) adapter WebSphere Process Server native binding (Web service, Java Message Service (JMS), HTTP, and EJB)	Depending on the type of WBI adapter, the migration tool creates a module that contains a JMS binding (to reuse the WBI adapter) or a Native binding (MQ/JMS/HTTP/EJB). The migration tool also generates Mediation Flow Components to provide integration logic around the adapter. If a J2C adapter is available to replace the WBI adapter, the tool allows you to proceed with the upgrade (JDBC, JText, EMail, SAP, and PeopleSoft are currently supported).

Mapping collaboration templates to BPEL processes: It can be tempting to map WebSphere InterChange Server collaboration templates to WebSphere Process Server BPEL processes only. However, Mediation Flow Components can be the better choice here. See Chapter 9, “Migration implementation approach” on page 151 for more information.

Table 7-2 describes less usual artifacts.

Table 7-2 Artifact mapping for other less common WebSphere InterChange Server components

WebSphere InterChange Server artifact	WebSphere Process Server artifact or concept	Support from the migration tools for WebSphere Process Server V6.1
DB connection pools	JDBC data sources	The migration tool generates a Jython script that is able to create the relevant JDBC data sources. See the following link for more information.
Scheduler	Scheduler	The migration tool generates a Jython script that is able to create Scheduler entries. See the following link for more information.
Server access interface	No equivalent feature. Must be replaced with another technology, such as Web services	No support.
Access EJB Clients	Any synchronous invocation mechanism	Yes, based on new Access EJB implementation and custom Selector, with manual rework. See the following link for more information.
DynamicSend API	Selector Dynamic endpoint in Mediation Flow Component (MFC)	Yes, based on custom Selector, with manual rework. See the following link for more information.
FailedEvent API	No equivalent feature	No
Benchmark	No equivalent feature	No
Business object probe	Technical monitoring or business monitoring. Monitoring can be based on logging or Common Event Infrastructure (CEI).	No

More information: See the WebSphere Process Server information center at the following address for details about specific aspects of the standard migration tools:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/rmig_post_cons_wics.html

To complement the previous build-time aspects, Table 7-3 lists other features that are available in WebSphere InterChange Server and their equivalent features in WebSphere Process Server.

Table 7-3 Other WebSphere InterChange Server features compared to WebSphere Process Server

Feature of WebSphere InterChange Server	Equivalent feature in WebSphere Process Server
Long-lived business process	Long running BPEL. Supported by the standard migration tools.
Event Sequencing	Process Server Event Sequencing, which is slightly different from WebSphere InterChange Server Event Sequencing. Supported by the standard migration tools.
Failed event manager	Failed Event Manager is the equivalent feature. It supports regular failed events as well as BPEL failed instances and SCA exception queues. It brings new capabilities, such as changing the message payload before resubmission. There is no support from the standard migration tools to upgrade unresolved flow instances.
Administration using System Manager and scripts	Administrative console and wsadmin command-line capabilities, which are quite different from WebSphere InterChange Server System Manager and scripts. There is no support from the standard migration tools to upgrade existing scripts or administrative procedures.

7.3 Benefits

WebSphere Process Server provides the following benefits, compared to WebSphere InterChange Server:

- ▶ WebSphere Process Server is a robust platform based on WebSphere Application Server, Network Deployment, which allows native clustering for high availability and load balancing.
- ▶ WebSphere Process Server provides more capabilities to automate administration procedures (using **wsadmin** command line).

- ▶ WebSphere Adapters and Native Bindings are embedded in the server and require a smaller memory footprint compared to WebSphere Business Integration Adapters.
- ▶ WebSphere Process Server implements a superior architecture built over standards and SOA.
- ▶ WebSphere Process Server brings enhanced support for Web services.
- ▶ WebSphere Process Server brings enhanced support for messaging with the help of bindings support for WebSphere MQ and JMS.
- ▶ WebSphere Process Server supports human intervention in the business process.
- ▶ WebSphere Process Server supports dynamic runtime change of service endpoints (late binding).

7.4 Limitations

WebSphere InterChange Server has the following native capabilities that do not have counterparts in WebSphere Process Server:

- ▶ Flow control
In WebSphere InterChange Server, flow control is a configurable service that you can use to manage the flow of connector and collaboration object queues. You can set it up at the system-wide level, connector level, and collaboration level. There is no similar capability in WebSphere Process Server.
- ▶ Fine-grained administration capabilities
In the WebSphere InterChange Server, you can change the status of individual components, such as collaborations, maps, and connectors. For example, you can pause a collaboration or a connector. Because the component model is different in WebSphere Process Server, the administration capabilities are more coarse-grained (at the application level generally, except for BPEL components). Also, the pause functionality does not exist in WebSphere Process Server.

Adapters and data handlers upgrade

In this chapter, we focus on the upgrade path for the WebSphere Business Integration Adapters and WebSphere Business Integration data handlers. First we provide an overview of the various adapter types and the equivalent concepts in WebSphere Process Server. Then we provide more details about the upgrade path for each type of adapter and the data handlers. We conclude with the major benefits and limitations of the upgrade.

This chapter complements Chapter 7, “Server component upgrade” on page 123, which focuses on the server artifacts. Chapter 9, “Migration implementation approach” on page 151 returns to the solution perspective and synthesizes the possible migration paths.

We include the following sections:

- ▶ 8.1, “Overview of WebSphere Business Integration Adapters” on page 132
- ▶ 8.2, “Adapter types” on page 134
- ▶ 8.3, “Adapter mapping” on page 134
- ▶ 8.4, “Equivalence of adapter functions” on page 136
- ▶ 8.5, “Upgrade paths” on page 137
- ▶ 8.6, “Benefits” on page 148
- ▶ 8.7, “Limitations” on page 149

8.1 Overview of WebSphere Business Integration Adapters

Adapters provide connectivity between the integration broker and a specific Enterprise Information System (EIS) or technology. WebSphere Business Integration Adapters can be used in conjunction with various IBM integration brokers, such as the WebSphere InterChange Server.

Because WebSphere Process Server can use the WebSphere Business Integration Adapters, there are two options to upgrade a WebSphere Business Integration Adapter:

- Keep the WebSphere Business Integration Adapter, which is illustrated in Figure 8-1.

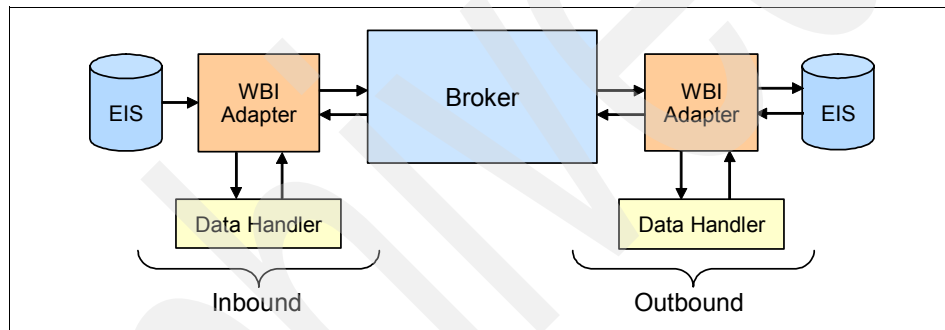


Figure 8-1 WebSphere Business Integration Adapters overview

- Upgrade the WebSphere Business Integration adapter to either a Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (J2C) Adapter or a WebSphere Process Server binding, depending on the specific adapter type. This approach is illustrated in Figure 8-2.

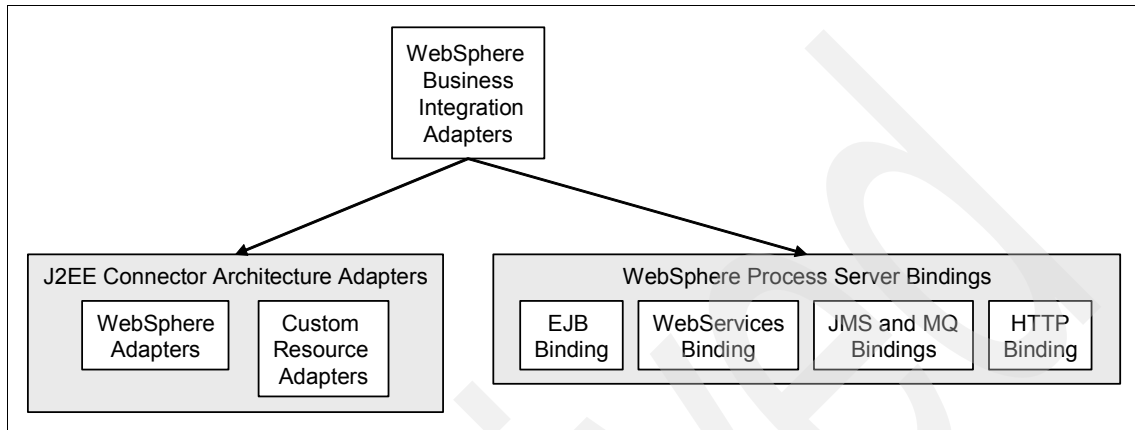


Figure 8-2 Upgrade paths for WebSphere Business Integration Adapters

In this chapter, we explain the adapter upgrade considerations and provide a detailed description of the possible adapter upgrade paths. You must carefully evaluate the upgrade paths that are described in this chapter for each WebSphere Business Integration Adapter in your Business Integration project. By performing this evaluation, you can determine if an upgrade to a more optimal solution is appropriate.

Each WebSphere Business Integration Adapter has a predefined upgrade path. The upgrade path depends on the following key factors:

- The type of WebSphere Business Integration Adapter
- The availability of upgrade solutions
- Support of the features that you require

In the remaining sections in this chapter, we describe the adapter types, adapter mappings, and functionally equivalent adapters. We follow this discussion by high-level descriptions of the upgrade paths, a summary of upgrade considerations, and best practices.

8.2 Adapter types

WebSphere Business Integration Adapters can be classified into one of two categories:

- ▶ *Application adapters* are designed to interface with a specific API for a specific version of an EIS.
- ▶ *Technology adapters* are designed to support a standard technology interface to any EIS that supports the same interface.

Application adapters are typically upgraded to a corresponding J2C-based adapter. These adapters are provided by IBM, in the form of WebSphere Adapters, or written by the user or Independent Software Vendor (ISV) by using the WebSphere Adapter Toolkit. Technology adapters are typically upgraded to corresponding WebSphere Process Server native bindings that are provided by WebSphere Process Server or WebSphere Enterprise Service Bus.

In all cases, where an appropriate J2C adapter or WebSphere Process Server binding does not exist, you have the option to continue using WebSphere Business Integration Adapters (or custom adapters) by using the migration runtime support that is provided in WebSphere Process Server.

8.3 Adapter mapping

WebSphere Business Integration Adapters can be upgraded to WebSphere Adapters or WebSphere Process Server bindings. The general rule is that WebSphere Business Integration application adapters can be upgraded to WebSphere Adapters, while WebSphere Business Integration technology adapters can be upgraded to WebSphere Process Server bindings. However, there are always exceptions to the rules. The tables in this section specify the correlation between WebSphere Business Integration Adapters and their upgrade counterparts.

Table 8-1 provides a correlation between specific WebSphere Business Integration Adapters and WebSphere Adapters.

Table 8-1 Adapter to adapter correlation

IBM WebSphere Business Integration Adapter for	WebSphere Adapters for
mySAP™.com®	SAP Software
PeopleSoft	PeopleSoft
Siebel	Siebel
Java Database Connectivity (JDBC)	JDBC
Oracle Applications	Oracle E-Business
JD Edwards®	JD Edwards
JText	Flat Files or FTP
E-mail	E-mail

Table 8-2 provides the correlation between specific WebSphere Business Integration Adapters and WebSphere Process Server bindings.

Table 8-2 Adapter to binding correlation

WebSphere Business Integration Adapter for	WebSphere Process Server binding type
MQ	MQ Java Message Service (JMS) or MQ
JMS	Internal JMS, MQ JMS, or generic JMS
Web services	Web services or HTTP
Enterprise JavaBeans (EJB)	EJB (or Web services)
HTTP	HTTP

Note: If you use the WebSphere Business Integration Adapter for WebSphere MQ, you can replace it with MQ JMS or MQ binding in WebSphere Process Server, depending on your specific needs.

If you use the WebSphere Business Integration Adapter for Java Message Service (JMS) with WebSphere MQ, the preferred method is to use the MQ JMS binding in WebSphere Process Server. Starting with WebSphere Process Server Version 6.2, support is available in the Migration Wizard for third-party JMS providers through the generic JMS binding and also the internal JMS binding.

Table 8-3 summarizes the possible options to cover other cases.

Table 8-3 Other cases

Situation	Possible options
No WebSphere Adapter counterpart exists for the current WebSphere Business Integration Adapter.	<ul style="list-style-type: none">▶ Keep the current WebSphere Business Integration Adapter.▶ Get a WebSphere Adapter from an ISV.▶ Develop a custom WebSphere Adapter.
You have developed a custom WebSphere Business Integration Adapter.	<ul style="list-style-type: none">▶ Keep the current WebSphere Business Integration Adapter.▶ Develop a custom WebSphere Adapter.

8.4 Equivalence of adapter functions

In addition to the benefits of J2C adapters over WebSphere Business Integration Adapters, generally WebSphere Adapters provide a broader set of features than their WebSphere Business Integration Adapter counterparts. However, it is important to familiarize yourself with the feature set of the WebSphere Adapters that you intend to use so that you can identify any potential issues before you begin the upgrade process.

WebSphere Adapters are constantly being updated to provide the broadest possible range of features and functionality. You must ensure that the features and functions on which you rely for your business integration scenarios are supported by the appropriate WebSphere Adapters. You can find this information in the relevant adapter user guide at the following Web address:

<ftp://ftp.software.ibm.com/software/websphere/integration/wsa/library/pdf620>

Note: The full functional equivalence between WebSphere Business Integration Adapters and WebSphere Adapters is reached with Version 6.2 of WebSphere Adapters. However, investigate the exact capabilities for each target WebSphere Adapter, because there are always slight differences. Also, if you plan to use an older version than 6.2 for a WebSphere Adapter, make sure that it supports the functionality that you require.

8.5 Upgrade paths

There are three primary upgrade paths for WebSphere Business Integration Adapters. Based on the adapter type, the adapter mapping, and the functional equivalence provided in the previous sections, you can use one of the following upgrade paths:

- ▶ Implementing a WebSphere Business Integration Adapter with WebSphere Process Server
- ▶ Upgrading a WebSphere Business Integration Adapter to a WebSphere Adapter
- ▶ Replacing a WebSphere Business Integration Adapter with a WebSphere Process Server binding

For completeness, you might have to investigate the following two additional upgrade options:

- ▶ Replacing a WebSphere Business Integration Data Handler with WebSphere Process Server data binding

If you upgrade a WebSphere Business Integration technology adapter to either a WebSphere Adapter or WebSphere Process Server binding, you generally have to handle the task of transforming data from the native format to (and from) business objects. In WebSphere Business Integration, this task is performed by using a Data Handler. In WebSphere Process Server, this task is performed by using data bindings.

- ▶ Writing a J2C adapter by using the WebSphere Adapter Toolkit

The WebSphere Adapter Toolkit allows you to develop J2C adapters to meet your unique business requirements. The toolkit helps you to create either a basic Java EE Connector Architecture (JCA) 1.5 adapter or an adapter that uses the additional capabilities of the Adapter Foundation Classes that are used by WebSphere Adapters.

We describe each of these paths in this section. In addition, read Chapter 9, “Migration implementation approach” on page 151 to gain a broader picture of

the overall upgrade path. Detailed examples are provided in Part 4, “End-to-end technical solutions” on page 277.

Table 8-4 provides a quick reference to specific examples that are related to each upgrade path.

Table 8-4 Upgrade path examples

Upgrade path	Examples
Implementing a WebSphere Business Integration Adapter with WebSphere Process Server	<ul style="list-style-type: none"> ▶ Appendix A, “WBI Adapters in a secured WebSphere Process Server environment” on page 571
Upgrading a WebSphere Business Integration Adapter to a WebSphere Adapter	<ul style="list-style-type: none"> ▶ Chapter 15, “Data access scenario with technology adapters” on page 293 ▶ Chapter 16, “Data synchronization scenario with technology adapters” on page 343 ▶ Chapter 17, “Data synchronization scenario with application adapters” on page 431
Replacing a WebSphere Business Integration Adapter with a WebSphere Process Server binding	<ul style="list-style-type: none"> ▶ Chapter 15, “Data access scenario with technology adapters” on page 293 ▶ Chapter 16, “Data synchronization scenario with technology adapters” on page 343
Replacing a WebSphere Business Integration Data Handler with WebSphere Process Server data binding	<ul style="list-style-type: none"> ▶ Chapter 15, “Data access scenario with technology adapters” on page 293 ▶ Chapter 16, “Data synchronization scenario with technology adapters” on page 343
Writing a J2C adapter by using the WebSphere Adapter Toolkit	See <i>WebSphere Adapter Development</i> , SG24-6387.

8.5.1 Implementing a WebSphere Business Integration Adapter with WebSphere Process Server

WebSphere Business Integration Adapters can be used with WebSphere Process Server. To use the adapter in this manner, you must create an enterprise application (enterprise archive (EAR) or module) that interfaces with the WebSphere Business Integration Adapter. This application must perform the following actions:

- ▶ Mediate between the WebSphere Business Integration business objects and the Service Data Objects (SDOs) that are used by WebSphere Process Server
- ▶ Resolve the differences in the interfaces that are exposed by the WebSphere Business Integration Adapter and the migrated business process

WebSphere Integration Developer offers several tools to help you automatically create this application:

- ▶ If you migrated from WebSphere InterChange Server to WebSphere Process Server by using the **reposMigrate** command or the WebSphere InterChange Server JAR File import wizard, these tools automatically create the required applications and resources as part of the overall migration.
- ▶ If you migrated from another broker, or you chose not to use **reposMigrate** or the WebSphere InterChange Server JAR file import wizard, you can use the WebSphere Business Integration Artifact Importer wizard. This wizard creates the application that is necessary to interface with your WebSphere Business Integration Adapter.

Figure 8-3 illustrates the generated topology in either case.

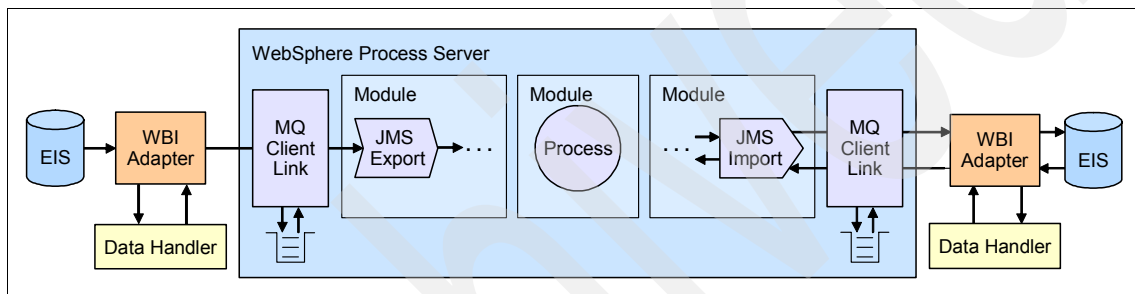


Figure 8-3 WebSphere Business Integration Adapter interface with WebSphere Process Server

The enterprise application (module) contains a JMS binding (export or import) that is used to communicate with the WebSphere Business Integration Adapter through JMS queues.

An intermediary MQ Client Link configuration is used as a bridge between the JMS binding and the WebSphere Business Integration Adapter agent JVM for the following reasons:

- ▶ The adapter agent must be configured to use JMS as a delivery transport, which means MQ underneath.
- ▶ The JMS binding communicates with the Service Integration Bus (SIBus) only (internal messaging layer).
- ▶ The MQ Client Link, which is hosted in the SIBus layer, can simulate a WebSphere MQ queue manager.

8.5.2 Upgrading a WebSphere Business Integration Adapter to a WebSphere Adapter

If an analogous WebSphere Adapter exists for your WebSphere Business Integration Adapter, you can choose to upgrade your WebSphere Business Integration Adapter to a WebSphere Adapter. Figure 8-4 illustrates this configuration.

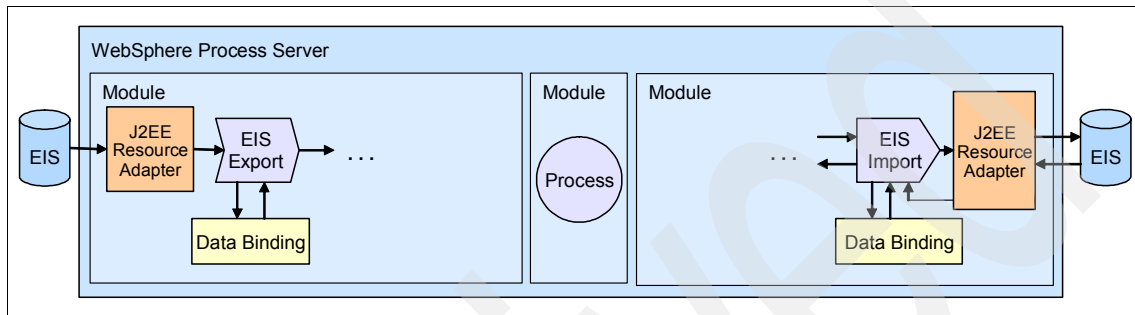


Figure 8-4 WebSphere Adapters with WebSphere Process Server

You can compare the configuration in Figure 8-4 with the one shown in Figure 8-3 on page 139. With a WebSphere Adapter configuration, the JMS binding shown in Figure 8-3 on page 139 is no longer needed, because the WebSphere Business Integration Adapter is not used anymore. Instead, EIS bindings (import and export) are created by the Enterprise Service Discovery wizard to allow usage of the specific WebSphere Adapter.

8.5.3 Upgrading a WebSphere Business Integration Adapter with a binding

If an analogous binding exists in WebSphere Process Server for your WebSphere Business Integration Adapter, you can choose to replace your adapter with the appropriate WebSphere Process Server binding. Figure 8-5 illustrates this configuration.

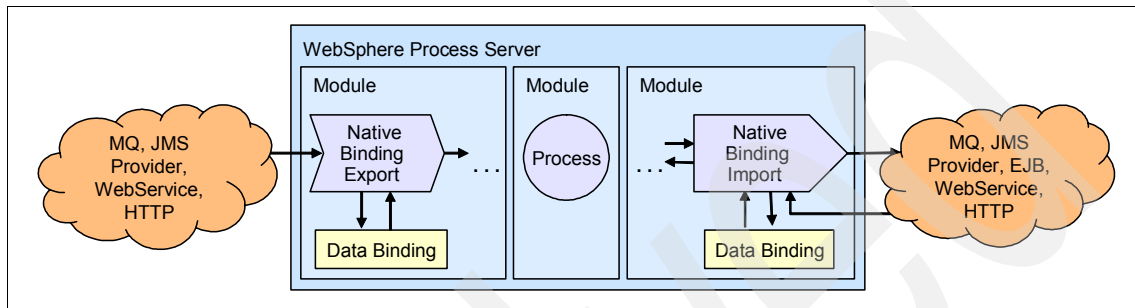


Figure 8-5 WebSphere Process Server bindings

You can compare the configuration in Figure 8-5 with the one in Figure 8-3 on page 139. With a WebSphere Process Server binding configuration, the JMS binding that is shown in Figure 8-3 on page 139 is no longer needed, because the WebSphere Business Integration Adapter is not used anymore. Instead, the appropriate WebSphere Process Server binding type is created, depending on the specific binding to use, either generic JMS, MQ JMS, MQ, Web services, HTTP, or Enterprise Java Bean (EJB). We provide details about each kind of binding in the following sections.

Replacing the WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture

By using the WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture, a WebSphere InterChange Server collaboration can exchange data with and invoke a stateless enterprise bean that resides on an application server. Figure 8-6 shows the configuration if the WebSphere Business Integration Adapter for EJB is used with WebSphere Process Server.

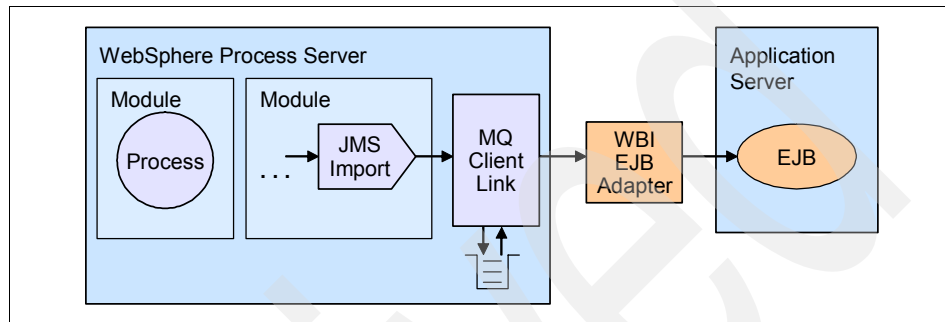


Figure 8-6 WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture used with WebSphere Process Server

Figure 8-7 shows the corresponding configuration by using a WebSphere Process Server EJB binding. Currently, only outbound operations (import) are supported with the EJB binding.

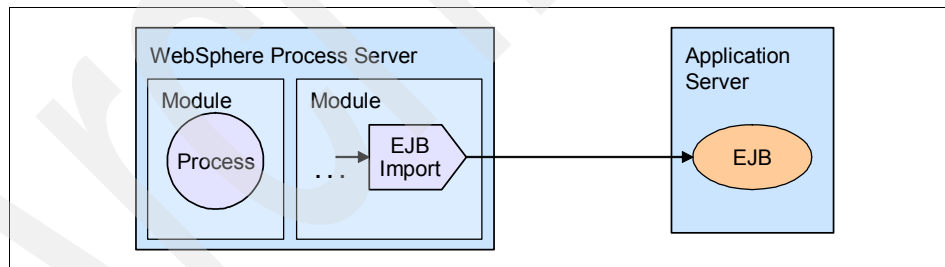


Figure 8-7 EJB Import binding

Replacing the WebSphere Business Integration Adapters for WebSphere MQ and JMS

The WebSphere Business Integration Adapter for WebSphere MQ is used to exchange MQ messages with WebSphere MQ. By using the WebSphere Business Integration Adapter for JMS, WebSphere integration brokers can exchange data with applications or EISs that send or receive data in the form of JMS messages. Figure 8-8 on page 143 shows the configuration that is obtained

if we reuse the WebSphere Business Integration Adapter with WebSphere Process Server.

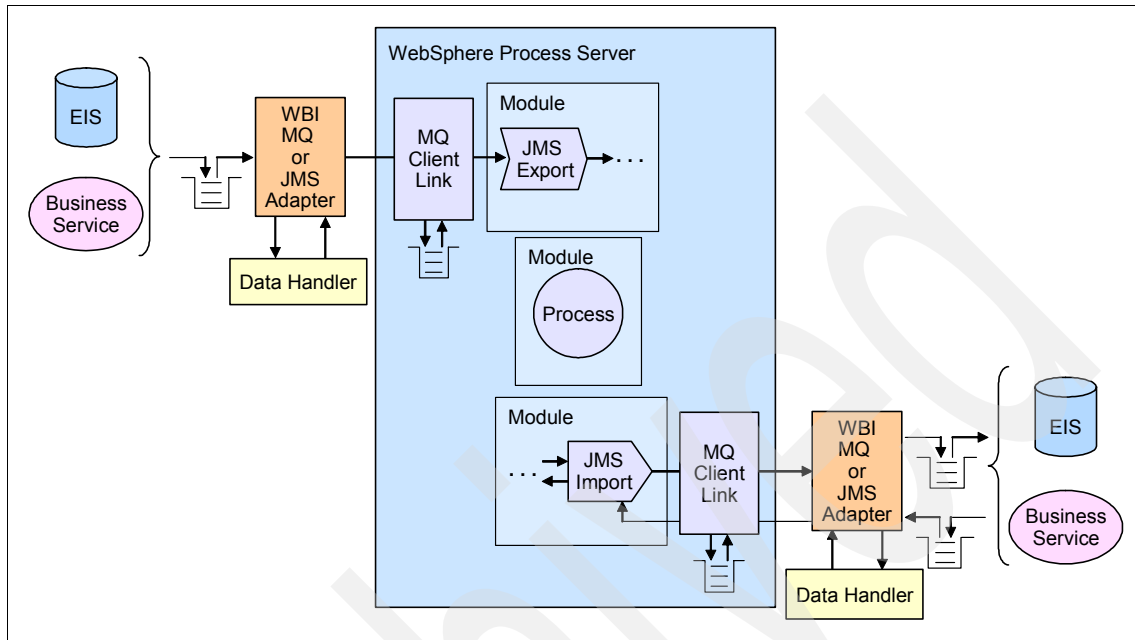


Figure 8-8 WebSphere Business Integration Adapter for WebSphere MQ or for JMS used in conjunction with WebSphere Process Server

Figure 8-9 shows the corresponding configuration by using a WebSphere Process Server binding.

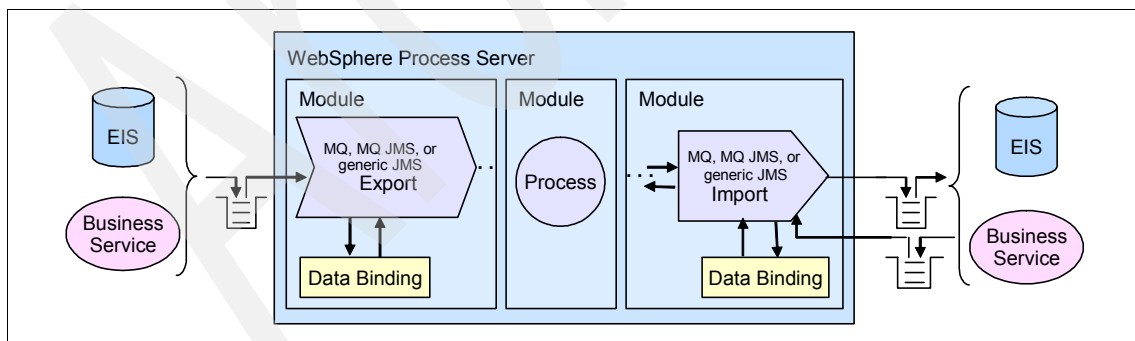


Figure 8-9 MQ or MQ JMS binding after adapter replacement

Replacing the WebSphere Business Integration Adapter for Web Services

The WebSphere Business Integration Adapter for Web Services provides the functionality to exchange business object information in the body of a SOAP message. With this adapter, WebSphere InterChange Server collaborations can invoke external Web services, and other clients can invoke WebSphere InterChange Server collaborations as Web services. Figure 8-10 shows the configuration prior to migrating from WebSphere InterChange Server to WebSphere Process Server.

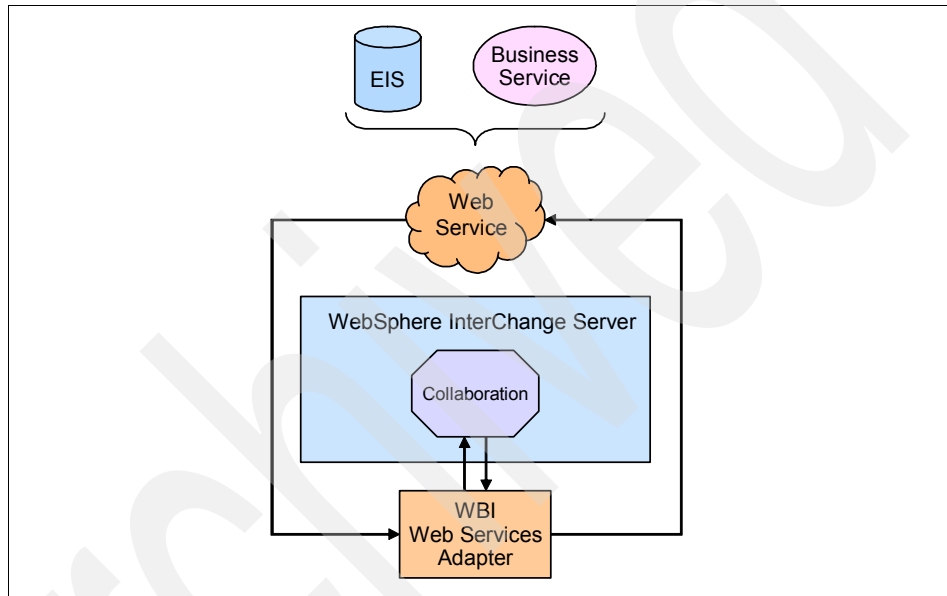


Figure 8-10 WebSphere Business Integration Adapter for Web Services before migrating to WebSphere Process Server

When migrating to WebSphere Process Server, we can replace the WebSphere Business Integration Adapter with a WebSphere Process Server binding as shown in Figure 8-11.

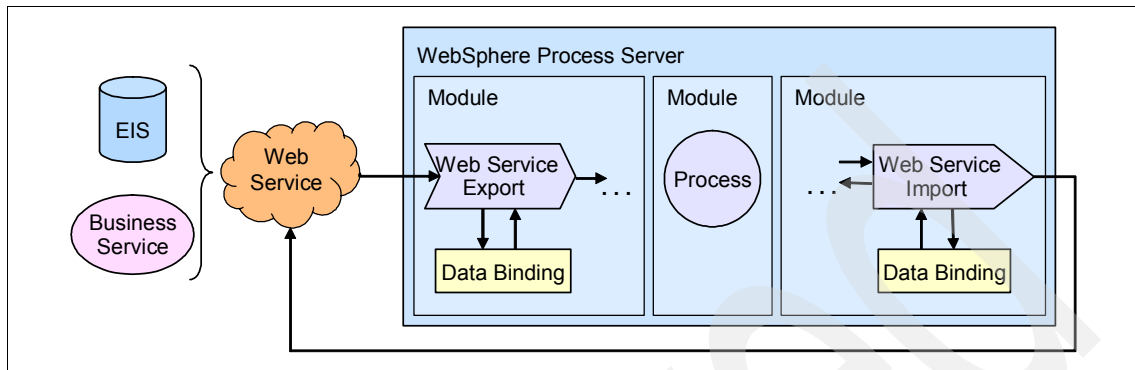


Figure 8-11 WebSphere Business Integration Adapter for Web Services upgraded to Web services binding

Replacing the WebSphere Business Integration Adapter for HTTP

The WebSphere Business Integration Adapter for HTTP is quite similar to the WebSphere Business Integration Adapter for Web services. Therefore, we do not provide additional detail here. The WebSphere Business Integration Adapter for HTTP can be replaced by a WebSphere Process Server HTTP binding.

8.5.4 Replacing a WebSphere Business Integration Data Handler

If you upgrade a WebSphere Business Integration technology adapter to either a WebSphere Adapter or WebSphere Process Server binding, you might have to handle the task of transforming data from the native format to (and from) business objects. In WebSphere Business Integration, this task is performed by using a Data Handler. In WebSphere Process Server, this task is performed by using data bindings.

To provide more background information, we first discuss data transformation. *Data transformation* refers to the process by which data is converted from an application native format to a business object or from a business object to an application native format. Data transformation differs fundamentally from data mapping, which is the process by which data elements are mapped (and possibly manipulated) from one business object to another business object.

WebSphere Business Integration Adapters perform data transformation in one of two ways, internally or externally:

- ▶ If the transformation is specific to the native data format of an EIS, the data transformation is handled internally by the adapter (such as JDBC or SAP).
- ▶ If the transformation is not specific to the native data format of an EIS, or the transformation depends on the contents of the native data, the data transformation is handled externally by a WebSphere Business Integration data handler.

Typically, WebSphere Business Integration application adapters handle data transformation internally. WebSphere Business Integration technology adapters, such as JText, MQ, or EMail, delegate the data transformation task to WebSphere Business Integration data handlers.

In contrast, in a WebSphere Process Server environment, data transformation is performed by a data binding. A *data binding* performs the same basic function as a WebSphere Business Integration data handler. However, rather than converting to and from a WebSphere Business Integration business object, the conversion is to and from an SDO. Another difference is that data bindings are typically invoked from import and export bindings that are specified in an Service Component Architecture (SCA) module.

Although WebSphere Business Integration data handlers and WebSphere Process Server data bindings are functionally equivalent, they are based on fundamentally different technologies. WebSphere Business Integration data handlers are based on the data handler interface within the WebSphere Business Integration Adapter Framework. Data bindings are based on SCA programming model extensions. When upgrading from a WebSphere Business Integration Adapter that uses an external WebSphere Business Integration data handler, to either a WebSphere Adapter or Process Server binding, an equivalent data binding must be provided.

The following options are available to provide an equivalent data binding:

- ▶ Reuse the WebSphere Business Integration data handler through a compatibility API called the *Heritage API* (HAPI). There are a few limitations related to this API, regarding performance and binary streams. Therefore, investigate the precise capabilities before adopting such a solution. However, generally speaking, HAPI can handle any text-based WebSphere Business Integration data handlers and therefore provides an interesting option for a tactical, short-term solution. For more information, see “Support for WebSphere Business Integration data handlers” in the WebSphere Process Server information center at the following address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/cmig_suptwbi_datahand.html

- Use an analogous WebSphere Process Server data binding, either by using an existing data binding or by writing a new custom one. The data bindings are detailed in the WebSphere Process Server documentation and the major ones are described in Chapter 5, “Product overview” on page 73.

Figure 8-12 illustrates the support of WebSphere Business Integration data handlers in WebSphere Process Server through HAPI.

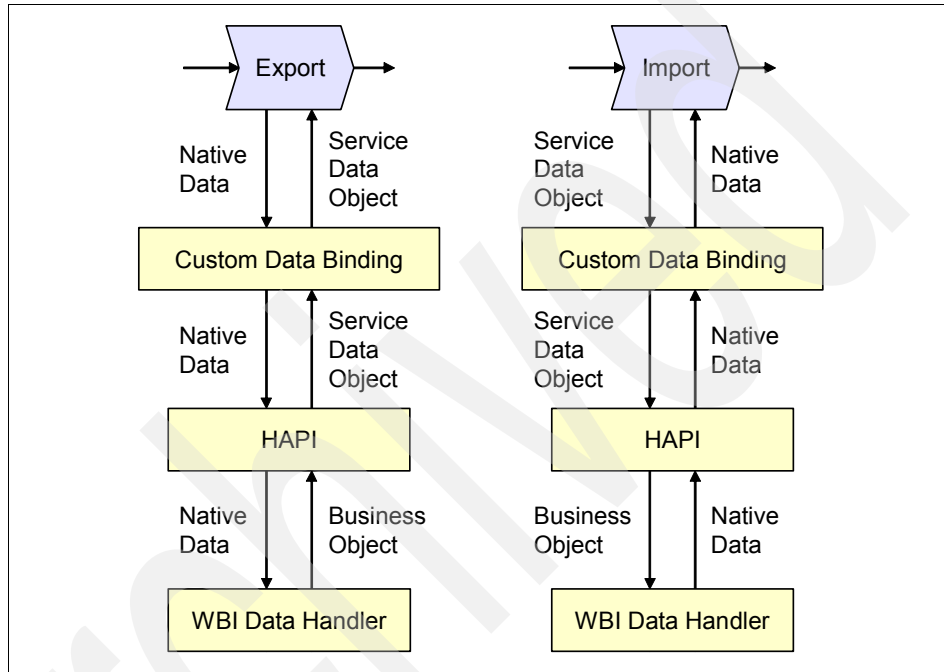


Figure 8-12 WebSphere Process Server support to reuse WebSphere Business Integration data handlers

8.5.5 Writing a custom adapter using the WebSphere Adapter Toolkit

With the WebSphere Adapter Toolkit, you can develop J2C adapters to meet your unique business requirements. The toolkit helps you to create either a basic JCA 1.5 adapter or an adapter that uses the additional capabilities of the Adapter Foundation Classes that are used by WebSphere Adapters.

See the Redbooks publication *WebSphere Adapter Development*, SG24-6387, which describes the adapter development process and how to use the WebSphere Adapter Toolkit.

8.6 Benefits

Upgrading your WebSphere Business Integration Adapter to either an analogous WebSphere Adapter or WebSphere Process Server binding has the following benefits, among others:

- ▶ Runtime integration

WebSphere Adapters and WebSphere Process Server bindings are integrated into WebSphere Process Server. WebSphere Business Integration Adapters are outside the application server. Communication overhead, memory footprint, and points of failure are reduced, while administration and monitoring are improved. Also, WebSphere Adapters can run natively as singletons in a WebSphere Process Server cluster, which is not the case for WebSphere Business Integration Adapters.

- ▶ Tools integration

All tools for WebSphere Adapters and WebSphere Process Server bindings are included in WebSphere Integration Developer. WebSphere Business Integration Adapters have separate tools for business object editing, connector configuration, data mapping, and process creation.

It is also much easier to deploy and debug issues when WebSphere Adapters and WebSphere Process Server bindings are used. WebSphere Adapters use Common Event Infrastructure (CEI) for logging.

- ▶ Open standards-based implementation

WebSphere Adapters and WebSphere Process Server bindings are based on the J2C and SCA specifications respectively. WebSphere Business Integration Adapters are based on a proprietary framework.

- ▶ Managed runtime environment

Both WebSphere Adapter and WebSphere Process Server bindings run in a managed application server environment. In this environment, they provide higher qualities of services, such as life cycle and connection management, security, and scalability. WebSphere Business Integration Adapters run in a non-managed environment and do not provide these benefits.

- ▶ Improved features and functionality

Overall, WebSphere Adapters provide a superior set of features and functionality compared to their WebSphere Business Integration Adapter counterparts.

The following improvements have been made to features and functions that are available in WebSphere Adapters:

- Synchronous outbound request with an XA or local transaction within the adapter
- Inbound request with assured “once and only once” delivery of inbound events
- Event management of the incoming events for the inbound request
Only asynchronous event delivery is supported by the WebSphere Adapters.
- Support for CEI
- Globalization with support for bidirectional data
- High availability support
- Filtering capabilities
- Multiple endpoint support
- First-failure data capture (FFDC) and Performance Monitoring Infrastructure (PMI) support

Although a limited set of WebSphere Adapters is available, use them wherever possible.

8.7 Limitations

When comparing the WebSphere Adapter and WebSphere Process Server bindings to the WebSphere Business Integration Adapters, the following limitations exist:

► Remote agent support

Remote agent support functionality is not provided with WebSphere Process Server, because the adapter or the binding is embedded in the server. Not having this support can be an issue in the following circumstances:

- Having a remote adapter that is local to the application to improve performance for event polling
- Having a remote adapter that is local to the application to allow access through an intermediate firewall

If these situations exist, investigate the possible options for WebSphere Process Server deployment to find an equivalent level of functionality.

► Native store and forward capability

As a reminder, within the WebSphere InterChange Server, you can specify Store and Forward mode and indicate how a connector controller will respond to a collaboration's request in a situation where the connector agent is unavailable:

- If you set the property to *True*, the connector controller does not fail any collaboration requests even if the connector agent is unavailable. A request is blocked until the connector agent is operational. This blockage causes a collaboration to wait until the connector agent is operational before it completes the processing flow for the request.
- If you set the property to *False*, the connector controller fails all collaboration requests if the connector agent is unavailable. This failure causes a collaboration to complete the processing of the request according to its business logic for processing a failed request.

This native capability does not exist with WebSphere Process Server. Therefore, custom design and development might be required to have similar functionality.

Migration implementation approach

In this chapter, we describe an approach to define the best technical path to implement the migration. We begin by describing integration principles that are useful to put the migration implementation into a solution perspective. Then we explain the major technical challenges to be expected during the migration implementation. Finally, we provide recommendations to select the appropriate migration path.

The focus of this chapter is on the technical implementation scope for the migration. Prior to reading this chapter, first read Chapter 4, “Migration planning” on page 33, which provides information about the overall migration scope. You can use the present chapter as an input for the Assessment and Preparation phases that are described in Chapter 4, “Migration planning” on page 33.

We include the following sections:

- ▶ 9.1, “Integration solutions perspective” on page 152
- ▶ 9.2, “Migration implementation challenges” on page 160
- ▶ 9.3, “Selection criteria for migration implementation” on page 169

9.1 Integration solutions perspective

In this section, we provide background information about integration concepts and integration solutions, in relation to the WebSphere InterChange Server to WebSphere Process Server migration implementation. Because we provide various and complementary views on the topic, this section serves as a relevant introduction before going into more detail about the technical challenges of the migration implementation.

9.1.1 Architectural design patterns and migration implementation

Architectural design patterns are a convenient way to describe any integration solution, such as WebSphere InterChange Server or WebSphere Process Server, without entering into implementation details (product specifics). Design patterns are helpful to start with a migration implementation, because they can be used as a common language. They allow WebSphere InterChange Server and WebSphere Process Server specialists to talk to each other with a limited risk of misunderstanding.

The next step is to see how design patterns can be used as a basis to build advanced migration tools (automatic migration). Ideally, we can imagine a migration tool that can perform the following tasks:

- ▶ Analyze the WebSphere InterChange Server artifacts
- ▶ Understand how they work together and their architectural purpose
- ▶ Translate the functionality into a solution view, decomposed into integration patterns
- ▶ Based on the solution view, generate the relevant artifacts in the new WebSphere Process Server programming model

These tasks are challenging, because the tool must be clever, especially for the recognition phase. Remember that the current standard migration tools approach has little to do with that phase. The major characteristics of the current standard migration tools are discussed in 9.3, “Selection criteria for migration implementation” on page 169. For a detailed explanation about the current standard migration tools, see Part 3, “Migration tooling” on page 177.

Even if such a powerful design pattern-based migration tool is not available, the design pattern approach remains a valid option to ensure that the migrated solution in WebSphere Process Server is compliant with design best practices.

For information about the design pattern approach and to find more pattern synergy, see Chapter 2, “Integration background” on page 9 and Chapter 3, “Usage patterns” on page 23.

9.1.2 Component model and integration layers

In this section, we provide a simple way to compare the WebSphere InterChange Server and WebSphere Process Server component models, which is important when determining the best migration path. A good starting point is to take an integration solution perspective.

Integration is the logical sharing of the same data between multiple physical systems. As an example, WebSphere InterChange Server is used mostly for Enterprise Application Integration (EAI), which is data driven. WebSphere InterChange Server is also middleware. Therefore, it is never the system of record for business data, except when storing data that is required for transformation (relationships).

In regard to the fundamental components of an integration solution, we look at a bottom-up approach, which is in the context of migration from WebSphere InterChange Server to WebSphere Process Server. Therefore, an integration solution already exists along with prerequisite systems and their methods of access.

To avoid any confusion with the products' component models, we use the term *layer*. An integration solution can be divided into three layers. The layers are connectivity, data transformation, and integration logic, which are explained in the following sections.

Note: WebSphere InterChange Server examples are discussed with an understanding that you are familiar with WebSphere InterChange Server.

Connectivity

To begin, we either access a system of record or make data (or even services) available to other systems, which requires connectivity. In WebSphere InterChange Server, we use WebSphere Business Integration Adapters or the Server Access Interface for connectivity. Without connectivity, we do not have integration.

Data transformation

The second concern is data transformation (sometimes called *data translation*). The data is represented in a variety of formats as diverse as the systems of record and technologies to which we must connect. Of importance to users is

that you can represent the data in its native format, which can be the following types:

- ▶ Unstructured (binary, such as .pdf, .doc, .xls, and so on)
- ▶ Semi-structured (CCB, fixedwidth, delimited, name-value, IDOCs, and so on)
- ▶ Structured (Java Database Connectivity (JDBC) result set, Business Application Programming Interface (BAPI), XML and XML Schema Definition Language (XSD), prepackaged APIs, and so on)

WebSphere InterChange Server creates the business object to allow for consistency in invoking the WebSphere Business Integration Adapters. In addition, the business objects provide structured transformation (business object maps) and interaction logic (collaborations).

WebSphere Business Integration data handlers are used to handle semi-structured to structured transformation (and the reverse is also true), where the structured data is always a WebSphere InterChange Server business object. The prepackaged applications (SAP, PeopleSoft, Siebel, Oracle Applications, and so on) have well defined APIs and data formats. For them, the data handler is embedded in the WebSphere Business Integration Adapter.

Integration logic

The third concern is integration logic. *Integration logic* is the logic when the access method, granularity, or number of systems does not match the desired representation.

Examples of integration logic include actions by a single system. It can refer to a retrieve that is done before the action when data enrichment is required. It can be that the granularity of the API does not match the unit of work of the business object. For example, the API might be unable to process a list of child objects at one time. It can be that the prescribed operation does not match the back-end data set. For example, an update must be changed to a create and a create must be changed to an update.

Examples of integration logic also include the actions of multiple systems. The data might need to be acted upon in a coordinated manner (because the systems are interdependent). Either all systems are updated, or none are updated. Another example is when we need data from one system for data enrichment to act upon another.

Certain integration logic is performed in WebSphere Business Integration Adapters, but most of it is, in the end, performed in collaborations. The `USE_RETRIEVE`, `CONVERT_CREATE`, `CONVERT_UPDATE`, and `ADDITIONAL_RETRIEVE` properties are standard in all collaborations (based on WebSphere Business Integration Collaboration Foundation). When dealing with

multiple systems, collaborations have the ability to enable compensation on all service calls (also called *logical rollback*).

Connectivity, data transformation, and integration logic are the key functional areas of integration. Without these layers, integration is impossible. They make up the core integration solution. Obviously, the layers depend on the following items:

- ▶ The physical location of the data
- ▶ The access methods for the data
- ▶ The format of the data in the access methods
- ▶ The relationships between the desired data representation and the locations and access calls that are required to perform the actions

These areas build upon each other with connectivity as the base, data transformation dependent on connectivity, and integration logic dependent on data transformation.

Figure 9-1 represents the component models of WebSphere InterChange Server and WebSphere Process Server through the core integration solution view.

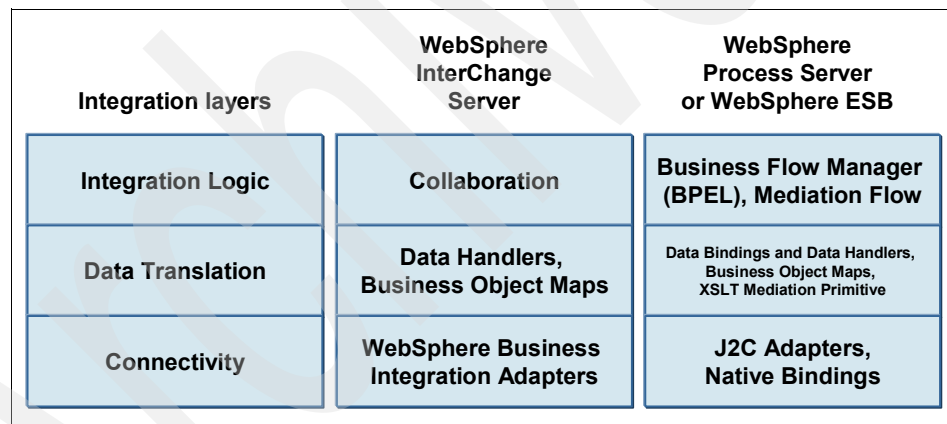


Figure 9-1 Integration layers and the components in WebSphere InterChange Server and WebSphere Process Server

9.1.3 Separation of concerns

To complement the preceding section about integration layers, in this section, we discuss an important concept called *separation of concerns*. Separation of concerns is a design best practice about doing the right thing, in the right place. Take this concept into account when determining the technical migration path.

Separation of concerns in the WebSphere InterChange Server

From this perspective, the WebSphere InterChange Server component model is straightforward:

- ▶ WebSphere Business Integration Adapter to handle connectivity
- ▶ Business object maps and data handlers to handle data translation
- ▶ Collaborations to handle integration logic

However, even with the simple WebSphere InterChange Server component model, in certain situations, the implementation might deviate from the design best practice. This deviation can be due to compelling reasons, such as performance enhancement or product limitations, but sometimes it is the result of a bad practice. The following two examples are typical of where the separation of concerns is not respected:

- ▶ Using JDBC code directly from a collaboration, instead of using a JDBC adapter
- ▶ Using an application-specific business object (ASBO) directly in a collaboration service call, instead of using the generic business object (GBO) and a map

Integration logic and business logic

Before we continue, to avoid confusion, you must understand that business process logic (also called *business logic*) has nothing to do with integration logic. We extend Figure 9-1 on page 155 with a new layer that represents business logic as shown in Figure 9-2.

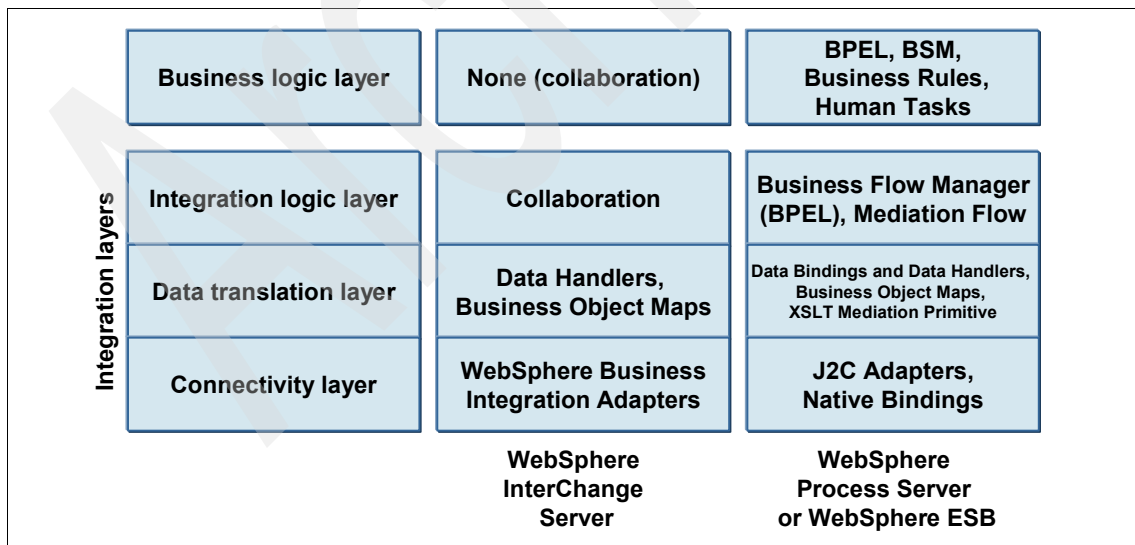


Figure 9-2 Integration layers and business logic layer

WebSphere InterChange Server is not optimized to handle business process logic, but it can handle integration logic. Obviously, collaborations can be used to a certain extent for business logic by calling an external rule engine through an adapter, for example. However, the WebSphere InterChange Server does not provide specific capabilities to handle that, except for long-lived business processes (LLBPs) in certain cases. Alternatively, WebSphere Process Server provides the following dedicated components to fulfill business logic implementation needs:

- ▶ Business Process Execution Language (BPEL)
- ▶ Business state machines
- ▶ Business rules
- ▶ Human tasks

In regard to possible upgrade paths between components, WebSphere InterChange Server collaborations do not necessarily map into WebSphere Process Server BPEL templates. There is no simple upgrade path rule, because collaborations are quite often a mix of integration logic and business logic, which is not good in terms of separation of concerns. However, here, it is due to the WebSphere InterChange Server component model limitation. Table 9-1 provides a basic categorization.

Table 9-1 Integration logic and business logic implementations

Type of logic	WebSphere InterChange Server	WebSphere Process Server
Simple integration logic	Simple collaboration (for example, pass-through from A to B, and simple data access pattern)	Mediation flow
Complex integration logic	Complex collaboration (for example, data enrichment and verb change handling, complex error handling and compensation, and LLBP for asynchronous or callback interaction)	Short running BPEL (microflow) or long running BPEL (macroflow)
Business logic	Collaboration doing business logic (for example, LLBP to handle business logic through human interactions, or collaboration calling an external rule engine)	Long running BPEL (macroflow) or business state machine, human tasks, or business rules

Separation of concerns and WebSphere Process Server

In regard to WebSphere Process Server, separation of concerns is about using the right granularity for modules. The general rule is to create separate modules for integration logic and business logic, but note that there might be exceptions.

We illustrate the common rule with two examples. In the first example (Figure 9-3), we show four modules that interact to provide an integration solution.

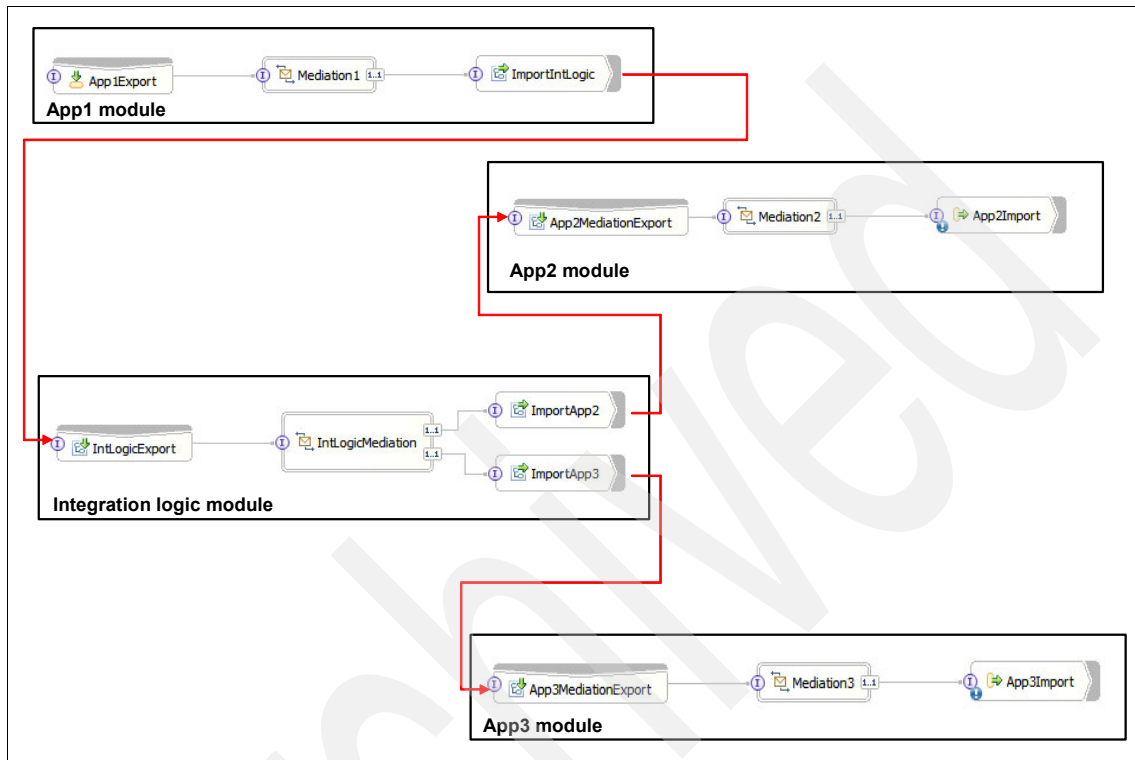


Figure 9-3 Separation of concerns in WebSphere Process Server: Integration logic example

In this example, note the following points:

- ▶ The integration logic is isolated in its own module called the *integration logic module*. The module uses a Mediation Flow Component called *IntLogicMediation*.
- ▶ Each application endpoint has also its own dedicated module:
 - *App1 module* contains a Mediation Flow Component called *Mediation1*. It represents the inbound endpoint of the integration solution.
 - *App2 module* contains a Mediation Flow Component called *Mediation2*. It represents the first outbound endpoint of the integration solution.
 - *App3 module* contains a Mediation Flow Component called *Mediation3*. It represents the second outbound endpoint of the integration solution.

This example shows how the three integration layers can be covered with WebSphere Process Server modules:

- ▶ The integration logic module handles data aggregation from App2 and App3 (integration logic) and sends it back to App1.
- ▶ The App modules handle data translation (business object mapping or XSL transformation, for example) and connectivity (through a WebSphere Adapter, for example) to communicate with each application properly.

Note that this example does not intend to provide a best general practice.

In the second example, shown in Figure 9-4, four modules interact to provide a business solution. This example is a bit artificial, but its purpose is to show how business logic and integration logic can be completely separated in WebSphere Process Server.

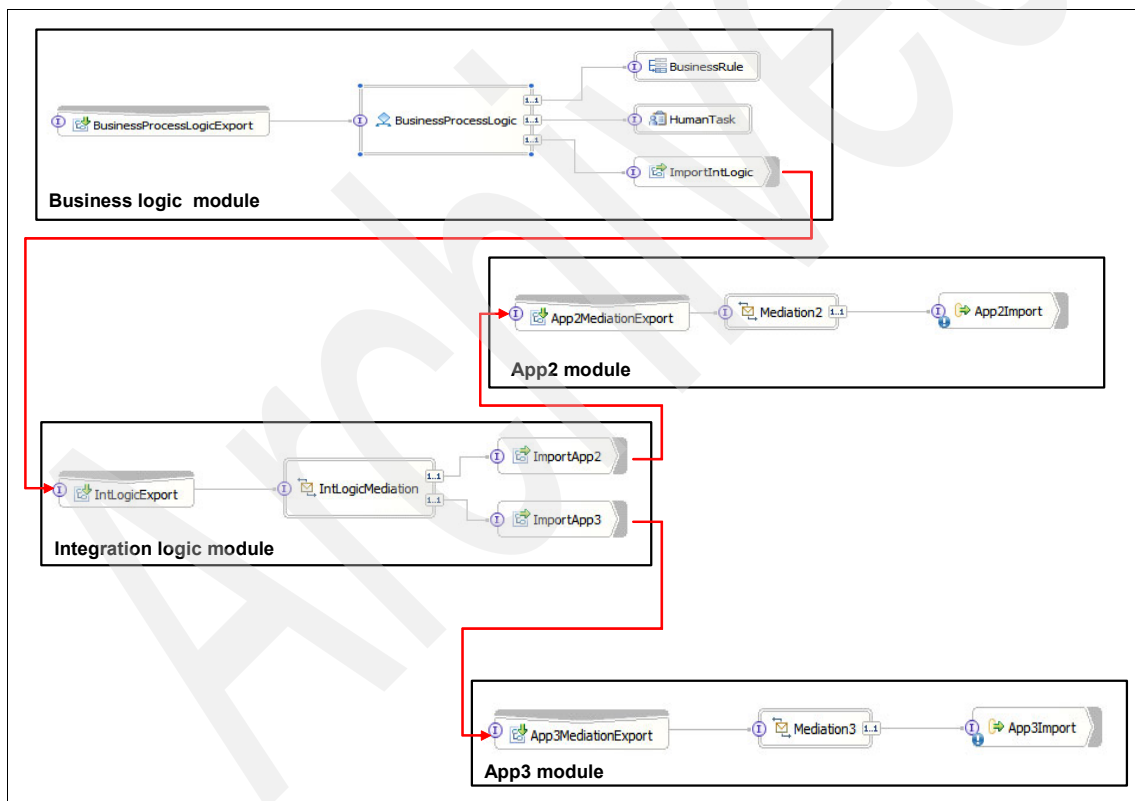


Figure 9-4 Separation of concerns in WebSphere Process Server: Business logic example

In this example, note the following points:

- ▶ The business logic module uses business components, such as human tasks and business rules. Instead of also implementing integration logic, it reuses the one available in the previous integration logic module.
- ▶ The other three modules are the same as in the preceding example (integration logic module, App2 module, and App3 module).

Benefits of separation of concerns

The major benefits of separation of concerns are related to the development and the deployment fields:

- ▶ Reuse (correct granularity of modules facilitates reuse)
- ▶ Team development (easy separation of tasks)
- ▶ Dynamicity and versioning (correct granularity of modules to facilitate advanced concepts)

See “Versioning and dynamicity with WebSphere Process Server” at the following Web address for more information:

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

- ▶ Flexible deployment

The correct granularity of the modules reduces the impact and eases deployment operations.

9.2 Migration implementation challenges

In this section, we describe the major technical challenges that can affect any migration implementation. It is important that you first read 9.1, “Integration solutions perspective” on page 152, because the concepts in this section rely on the concepts that we explained in the previous section.

9.2.1 Upgrade of WebSphere Business Integration Adapters

To introduce the challenge of upgrading WebSphere Business Integration Adapters, we return to the integration layers that we describe in 9.1.2, “Component model and integration layers” on page 153. It is important to understand that the three integration layers are interrelated and depend on each other. It is particularly obvious for the case where we want to know what happens when we change something in the connectivity layer:

- ▶ Generally, the data representation is also changed. This change always leads to changing the data transformation. That sometimes implies a change to the integration logic depending on how much the data representation has changed. For example, the new connectivity means might require only a single call to get all the relevant data, where before you needed two separate calls.
- ▶ A change in the connectivity layer can also impact aspects of the run time, such as the quality of service (QoS). If QoS is impacted, it is sometimes necessary to compensate this change by adding new (or changing) features in the connectivity or in the integration logic layer, for example, to implement or remove an automatic retry.

We now investigate what it means for the migration implementation.

Impact on ASBOs and maps

What happens if we change a WebSphere Business Integration Adapter into a WebSphere Adapter or a WebSphere Process Server binding? There is possibly an impact at the ASBO level, because ASBOs depend upon the connectivity means. That is, ASBOs that are related to WebSphere Business Integration Adapters might be different from the ASBOs that are related to WebSphere Adapters and WebSphere Process Server bindings.

As a consequence of ASBO changes, business object maps might also be impacted. The maps must be modified or even replaced if new ASBOs, with a different structure, are required. Figure 9-5 on page 162 and Figure 9-6 on page 162 show the different ASBOs and maps, in relation to the WebSphere InterChange Server run time and the WebSphere Process Server run time.

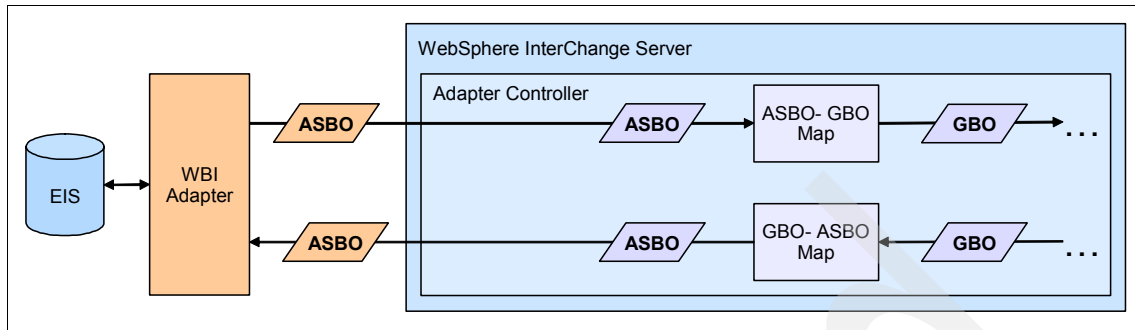


Figure 9-5 WebSphere Business Integration Adapter and WebSphere InterChange Server

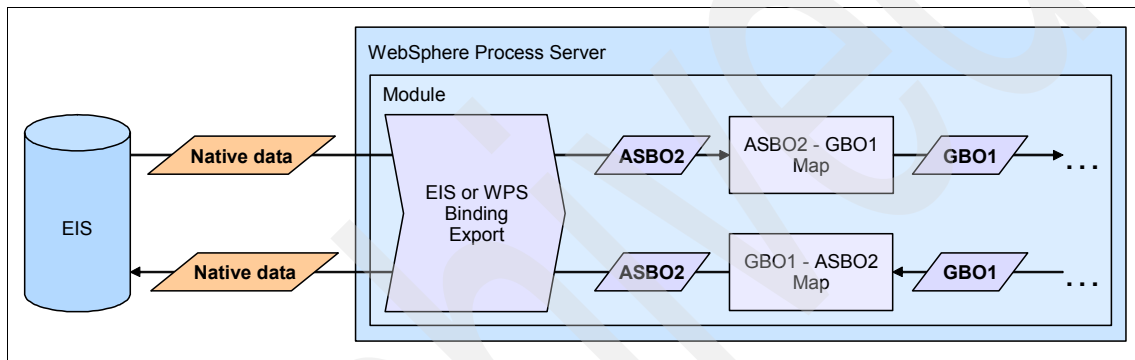


Figure 9-6 WebSphere Adapter or WebSphere Process Server binding

Note: All figures in 9.2.1, “Upgrade of WebSphere Business Integration Adapters” on page 161 show inbound interactions only, but the discussion is also perfectly valid for outbound interactions.

WebSphere Business Integration Adapters are technologically different from WebSphere Adapters and WebSphere Process Server bindings, at first glance. However, there is no reason why the WebSphere InterChange Server ASBO (called ASBO in our illustration) is the same as the WebSphere Process Server ASBO (called ASBO2 in our illustration). As a consequence, if the structure of the ASBO is changed, the maps cannot be the same neither. Here, the ASBO-GBO map in the WebSphere InterChange Server is different from the ASBO2-GBO1 map in WebSphere Process Server.

There is no real point to compare components that live in different run times. These components must obviously differ if you think of the Java implementation on which they rely. Therefore, we must go a step further and see how

WebSphere InterChange Server components, such as business objects, are translated into WebSphere Process Server business objects (Service Data Objects (SDOs)).

Translation of business objects into Service Data Objects

In Figure 9-7 on page 164, we show how a WebSphere Business Integration Adapter that was previously used with the WebSphere InterChange Server can be reused with WebSphere Process Server. Note the following points:

- ▶ The WebSphere InterChange Server uses business objects as its runtime data format.
- ▶ WebSphere Process Server uses SDOs as its runtime data format.
- ▶ Often the term *business object* is also used in the WebSphere Process Server context, but the underlying implementation is *always* an SDO.

You might wonder how the WebSphere Business Integration Adapter is able to communicate with the WebSphere Process Server, because they do not use the same runtime data format.

WebSphere Process Server incorporates a series of artifacts (MQ client link (not shown here) and Java Message Service (JMS) export) that acts as an intermediary layer with the WebSphere Business Integration Adapter. Therefore, the adapter remains entirely unaware of the real server run time and continues to exchange serialized WebSphere InterChange Server business objects with the server through JMS.

The JMS export can then convert the serialized WebSphere InterChange Server ASBO into a live SDO and the reverse, too. This conversion is possible, because WebSphere Integration Developer incorporates an upgrade logic that can import any WebSphere InterChange Server business object definition (XML schema) and convert it into a corresponding SDO definition (XML schema close to the previous one but slightly different).

The resulting SDO is close to the original WebSphere InterChange Server business object, because it relies on the same definition. However, because it lives in the WebSphere Process Server run time, it is different. That is why we used different names: ASBO becomes ASBO1 and GBO becomes GBO1 when you compare Figure 9-5 on page 162 and Figure 9-7 on page 164.

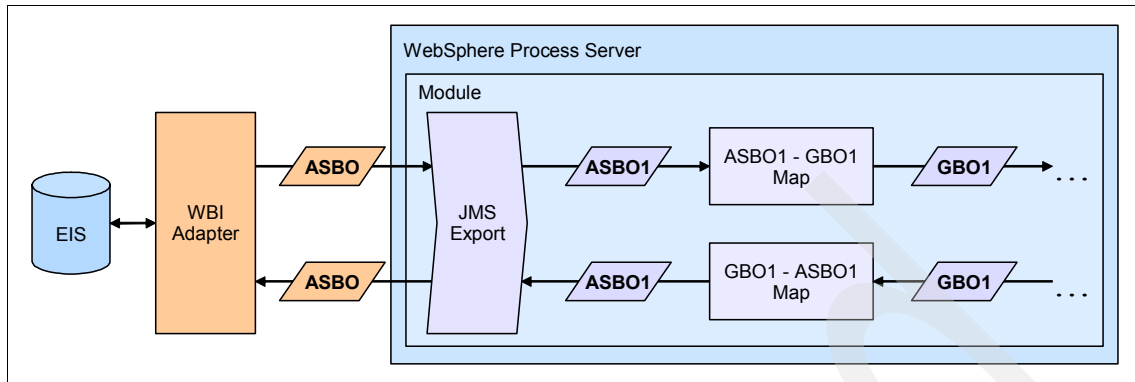


Figure 9-7 WebSphere Business Integration Adapter reused with WebSphere Process Server

Now that we understand the difference between the WebSphere InterChange Server business object and the WebSphere Process Server SDO, we return to understanding the impact on the ASBOs and business object maps when upgrading a WebSphere Business Integration Adapter.

ASBOs and mappings

WebSphere Business Integration Adapters are technologically different from WebSphere Adapters and WebSphere Process Server bindings. We now compare Figure 9-6 on page 162 and Figure 9-7. Note the following points:

- ▶ ASBO1 is an SDO that corresponds to the original WebSphere Business Integration Adapter.
- ▶ ASBO2 is the SDO that corresponds to a new WebSphere Adapter or a WebSphere Process Server binding.
- ▶ ASBO2 might be different from ASBO1, because the connectivity component that uses it is different.
- ▶ If ASBO2 is structurally different from ASBO1, there is an impact on maps, because new maps are needed to transform ASBO2 into a canonical representation (GBO1) to allow proper downstream processing.

In summary, the impacts that are related to the adapter upgrade mainly depend on the differences between ASBO1 and ASBO2. There are two scenarios:

- ▶ In the first scenario, the ASBO1 and ASBO2 definitions are close or even identical. In this case, the WebSphere Business Integration Adapter upgrade does not bring extra effort to the overall migration effort, because its replacement is transparent. The differences between the definitions for ASBO1 and ASBO2 depend on the specific WebSphere Business Integration Adapter used and the complexity of the original ASBO. It must be investigated on a case-by-case basis. For example, the standard migration tooling

available with WebSphere Process Server 6.2 supports migration of JDBC and SAP adapters. The migration tooling is specifically designed to generate new ASBOs that are very close to the original ones. Only the Application Specific Information is different, and therefore, mappings are not impacted.

- In the second scenario, we must deal with a different ASBO2 definition. In this case, the following options are possible:

- Redevelop the mappings from scratch (as shown in Figure 9-6 on page 162).

In this case, you must create new maps to convert ASBO2 into GBO1 and GBO1 into ASBO2. This option produces the most optimized code but is the more costly in terms of new developments. However, it can be a valid option if the new maps are straightforward (small BO definitions, or simple transformations, such as moves, for example).

- Add an extra mapping to reuse the migrated mappings.

The WebSphere Integration Developer tool can provide artifacts that correspond to GBO1, ASBO1 and ASBO1-GBO1, and GBO1-ASBO1 maps. It can then reuse the available migrated components and add new secondary maps (ASBO2-ASBO1 and ASBO1-ASBO2 maps), as shown in Figure 9-8.

The standard migration tools available for V6.0.2 and V6.1 were designed with this approach. They included support to generate the secondary maps (through a tool called a *BOconverter* in WebSphere Process Server V6.0.2 and V6.1). This option spares development time but might not be optimal in terms of performance and maintainability. That is why the standard migration tooling has been enhanced in V6.2 for certain adapters (such as JDBC and SAP) to produce ASBOs that do not impact the migrated maps and do not require additional maps.

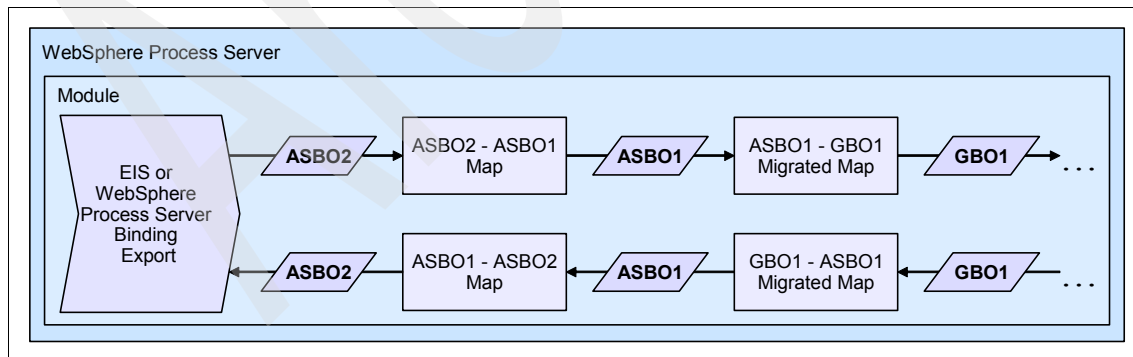


Figure 9-8 Reusing migrated maps

Impact on run time

Upgrading a WebSphere Business Integration Adapter to a WebSphere Adapter or a WebSphere Process Server binding might impact the resulting QoS or performance at run time, because the components are not strictly equivalent:

- ▶ When replacing with a WebSphere Adapter, the following capabilities among others might require investigation:
 - Performance
 - Automatic reconnection
 - Automatic retry in case of a communication error
 - Store and forward
- ▶ When replacing with a WebSphere Process Server binding, the following capabilities, among others, can be impacted:
 - Performance
 - Poll quantity tuning
 - Poll period tuning
 - Automatic reconnection
 - Automatic retry in case of a communication error

Therefore, it is necessary to verify if QoS and performance requirements are met after the upgrade. If they are not met, you might be required to develop the missing capabilities to fill in the gaps. Depending on the situation, the required extra developments can be located in the connectivity layer (module containing the WebSphere Adapter or the WebSphere Process Server binding artifacts). Alternatively, the developments can be in the integration logic layer (module containing the components in charge of the integration logic, such as the main Mediation Flow Component or BPEL component).

9.2.2 Functional gaps between products

The challenge of functional gaps between products is related to the differences between the run times of WebSphere InterChange Server and WebSphere Process Server. It is also related to how run times can affect the resulting QoS and runtime behavior of the migrated solution. This challenge comes with the following concerns:

- ▶ Flow control

In WebSphere InterChange Server, flow control is a configurable service with which you can manage the flow of connector and collaboration object queues. The parameters for configuring flow control can be configured system-wide or on individual components. There is no equivalent capability in WebSphere Process Server.

► Event sequencing

WebSphere InterChange Server and WebSphere Process Server present similar concepts around event sequencing. However, WebSphere Process Server provides a richer set of interaction styles (synchronous and asynchronous) and a richer model for deployment (native clustering). Therefore, event sequencing in WebSphere Process Server can be more powerful but also more complex than in WebSphere InterChange Server.

► Error handling

WebSphere Process Server provides a richer set of components to implement integration patterns, compared to the WebSphere InterChange Server. WebSphere Process Server relies on different layers, such as WebSphere Application Server, Service Integration Bus, Service Component Architecture (SCA), and the business process choreography engine. For these reasons, error handling is generally more powerful, but also more complex than with the WebSphere InterChange Server. As an example, when an error occurs during an asynchronous flow, it can fall into three categories:

– Messaging-based dead letter queues

Messaging-based dead letter queues can happen if a JMS binding is used. The data is then stored in the internal messaging layer (Service Integration Bus).

– Failed event

A failed event can happen in the case of an error during an asynchronous interaction between two SCA components. The data is then stored into specific tables that are quite similar to the WebSphere InterChange Server work-in-progress tables. WebSphere Process Server provides a new and useful functionality with which you can modify the data before resubmission.

– Business process instance in a failed or stopped state

Having a business process instance in a failed or stopped state can happen when an error is raised during a long running process execution. The data is then stored in the Business Process Choreographer layer (BPEDB) and can be resubmitted at the process instance or process activity level.

In summary, you might need more thorough investigation into these topics to ensure that the migration implementation is optimal. Sometimes you can redesign in a specific way to use new technical functionalities or to develop extra components to fill in gaps, depending on the requirements of the target platform.

9.2.3 Specific Java developments

The challenge of specific Java developments is that they are related to the customizations that are possibly done on the WebSphere InterChange Server project. It is common for any WebSphere InterChange Server project to have custom developments done in Java to fulfill specific needs that the product cannot provide natively.

Custom Java libraries or Java code snippets are typically used for the following purposes:

- ▶ Extensive custom Java coding in collaboration templates
- ▶ Extensive custom Java coding in maps
- ▶ Custom WebSphere Business Integration data handlers
- ▶ Custom WebSphere Business Integration Adapters
- ▶ Custom frameworks (for example, error handling, auditing, and monitoring)

The possibility to migrate the specific Java code into the new WebSphere Process Server run time depends on the following three categories:

- ▶ Usage of public and documented WebSphere InterChange Server APIs (including WebSphere Business Integration Adapter APIs)

WebSphere Process Server provides support to directly reuse the WebSphere InterChange Server Java public APIs in its run time, through a compatibility layer called the *Heritage API* (HAPI). The supported APIs are described in “Supported WebSphere InterChange Server or WebSphere Business Integration Server Express APIs” in the WebSphere Process Server information center at the following Web address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/rmig_wics_apis.html

However, keeping old APIs in the new run time can impact the performance and maintainability of the solution. Therefore, consider this approach only for the short term. The problem is that translating an API into another API is also a complex task. There are few chances that this can be done automatically, which is not how the current migration tools work.

- ▶ Usage of non-public or non-documented WebSphere InterChange Server APIs

There is no support for this usage. The impacts related to the upgrade must be examined on a case by case basis. The migration implementation can be seen as an opportunity to remove these APIs calls.

- ▶ Non-WebSphere InterChange Server APIs

There is no support for these external APIs. The impact that is related to the upgrade must be examined on a case-by-case basis. Because the new

platform is J2EE-based, and the old one is Java 2 Platform Standard Edition (J2SE)-based, a good part of the impact depends on the backward compatibility of Java. Also, the J2EE specification brings new concepts and best practices around transactionality and container-managed components that can impact the usage of certain APIs (JDBC, threads, or I/O).

Always verify whether the new run time can provide equivalent (or even better) native capability to avoid custom Java code as much as possible.

9.3 Selection criteria for migration implementation

In this section, we discuss the possible technical approaches to implement the migration and the related selection criteria to consider. The two first criteria to take into account are obviously the capability of the migration to fulfill the technical requirements and the overall cost of the migration.

More information: We focus on the technical scope in this section. As a reminder, the migration project is much broader than a technical upgrade path. See Chapter 4, “Migration planning” on page 33 for general considerations about project planning.

9.3.1 Technical approaches

There are two major and opposite approaches regarding the technical migration path:

- ▶ Rely completely on the standard migration tools (maximum automation).
- ▶ Redesign and develop completely from nothing (maximum customization).

Both approaches present advantages and limitations. Generally, the best choice is somewhere in between and can be executed in two ways:

- ▶ Use the standard migration tools and rework the results afterward to enhance the migrated artifacts. A typical scheme is to fully rely on the standard migration tool to generate the artifacts and spend the necessary time to fix the errors or issues that arise. We call this scheme the *just make it work* approach.
- ▶ Redesign completely for certain parts, and produce new developments in conjunction with automatically migrated artifacts from the standard migration tools to accelerate certain upgrade steps. A typical scheme here is to rely on the standard migration tool to generate the simplest and most numerous artifacts, such as BOs, maps, and relationships, and then to rewrite the

integration logic from the beginning. We call this the *redesign integration logic* approach.

So, we have to choose between four major approaches. Obviously, the overall migration project might decide to use several approaches, depending on the complexity and the requirements of the different interfaces to migrate. Figure 9-9 summarizes the four approaches and their capacity to provide general quality as compared to the quick delivery of the migrated solution.

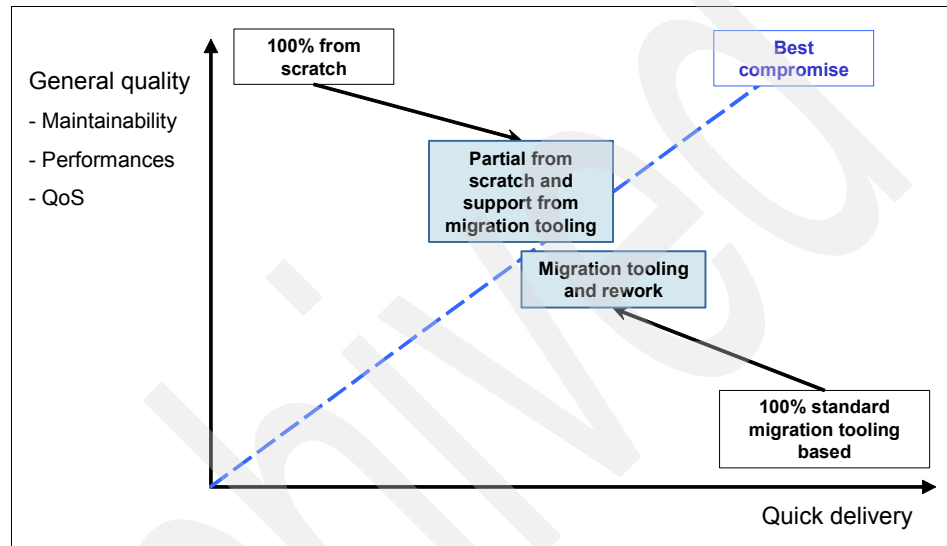


Figure 9-9 Approaches comparison

In addition to the previous considerations, the WebSphere Business Integration Adapters weigh heavily on the scale. Regarding the upgrade of the adapters, there are two approaches:

- ▶ The *two-step approach for adapters* consists of the following actions:
 - a. Migrate the server artifacts only into the new WebSphere Process Server run time, keeping the WebSphere Business Integration Adapters.
 - b. Upgrade the WebSphere Business Integration Adapters to WebSphere Process Server bindings or WebSphere Adapters.
- ▶ The *server and adapters all together approach* consists of migrating the adapters and the server artifacts conjointly at the same time.

Again, both approaches present advantages and limitations. However, we consider that the *server and adapters all together approach* is preferable due to the strong impact adapters have on the overall design, the runtime behavior, and QoS. Consider the *two-step approach for adapters* only on an exception basis

and always as a tactical, short-term solution. For example, custom WebSphere Business Integration Adapters cannot be migrated directly with the standard migration tools. It is possible to reuse them as is with WebSphere Process Server, but an effort needs to be made to rewrite them with JCA technology to get a long-term solution.

In the next sections, we provide more detailed information about the factors to consider to select the approach that best fulfills the specific project needs.

Comparison of the migration approaches

Consider several factors before deciding to use either approach. One of the factors is the effort estimation, which can be a driver for the migration project. To make the decision, we need to compare the approaches for migration.

Table 9-2 compares the migration approaches along with the potential benefits.

Table 9-2 Comparison of the migration approaches

Benefit	WebSphere InterChange Server: No migration	WebSphere Process Server: Small migration effort <i>(just make it work approach)</i>	WebSphere Process Server: Large migration effort <i>(redesign integration logic approach)</i>	WebSphere Process Server: Large migration effort, plus migrating all WBI Adapters to J2C	WebSphere Process Server: Complete rewrite <i>(from scratch approach)</i>
Better error handling	X	X	Yes	Yes	Yes
Better Availability	X	Yes	Yes	Optimal	Optimal
Better Scalability	X	Yes	Yes	Yes	Yes
Lower memory resources	X	Yes	Yes	Optimal	Optimal
Improved response time performance	X	Yes	Yes	Optimal	Optimal
Better testing	X	Yes	Yes	Yes	Yes
Automated build	X	Yes	Yes	Yes	Yes

Benefit	WebSphere InterChange Server: No migration	WebSphere Process Server: Small migration effort (<i>just make it work</i> approach)	WebSphere Process Server: Large migration effort (<i>redesign integration logic</i> approach)	WebSphere Process Server: Large migration effort, plus migrating all WBI Adapters to J2C	WebSphere Process Server: Complete rewrite (<i>from scratch</i> approach)
Automated deployment	X	Yes	Yes	Optimal	Optimal
More powerful tooling	X	Yes	Yes	Optimal	Optimal
Run on WESB Server	X	Some modules	Some modules	Some modules	Yes
Reduction of custom adapters	X	No	No	Yes	Yes
Optimal design	X	No	Almost	Almost	Yes
Optimal performance	X	No	Almost	Almost	Yes
Concern about support of code	Yes	Some	Little	Almost none	No

In the following sections, we provide additional details to complete this discussion about the possible technical approaches.

9.3.2 Technical approaches and the standard migration tools

The standard migration tool is optimized to reuse existing components. It was designed originally to follow the *two-step approach for adapters* described in the previous section. However, starting with Version 6.2, it is able to follow the *server and adapters all together approach*, for certain adapters. Here is a quick overview of the capabilities of the standard tooling with regard to the technical approaches described previously:

- WebSphere Integration Developer migration tool provides support to migrate server components (business objects, business object maps, collaborations, and adapter definitions):

- The custom Java snippet code in the WebSphere InterChange Server artifacts, such as collaborations and business object maps, is copied into the new artifacts nearly unchanged. It exploits the runtime support for the WebSphere InterChange Server APIs in WebSphere Process Server.
- The tool provides support to reuse the existing WebSphere Business Integration Adapters through JMS binding.
- The tool is now able to migrate WebSphere Business Integration Adapters to native bindings and JCA adapters. Currently, all technical adapters are supported, and application adapters, such as SAP and PeopleSoft adapters, are also supported.
- ▶ With the former (before V6.2), external, non-WebSphere Integration Developer standard migration tool that was dedicated to WebSphere Business Integration Adapters, you can still upgrade several WebSphere Business Integration Adapters that are not supported in Version 6.2 and get a solution that reuses existing ASBOs and maps (BOConverter tool to implement the secondary maps).

See Part 3, “Migration tooling” on page 177, for more information about the standard migration tools.

9.3.3 Custom code and standard migration tool constraints

The standard migration tools require that you apply best premigration practices to properly migrate collaborations and maps that contain custom Java code. Often, you must modify the existing WebSphere InterChange Server collaboration templates and map artifacts to allow their upgrade by the standard migration tools.

For more details, see Chapter 13, “Best practices” on page 263 and “Premigration considerations” in the WebSphere Process Server information center at the following address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/rmig_bestwics.html

9.3.4 Quality of automatically migrated code

The following potential issues are related to the quality of the generated code by the standard migration tools:

- ▶ Custom Java code

The custom Java code in the WebSphere InterChange Server artifacts, such as collaborations and business object maps, is copied into the new artifacts

nearly unchanged. It exploits the runtime support for the WebSphere InterChange Server APIs in WebSphere Process Server, which is practical to accelerate the upgrade phase.

The disadvantage is that the generated WebSphere Process Server artifacts still contain old WebSphere InterChange Server API calls, which can have a negative effect on maintainability of the Java code. A WebSphere Process Server developer might find it strange to deal with old WebSphere InterChange Server APIs.

- Component and module generation

The standard migration tools generate WebSphere Process Server artifacts and modules to produce a similar application to the original application running in WebSphere InterChange Server. The new application can be used to replace the old application from a functionality perspective. However, the new application is not a “textbook example” of a WebSphere Process Server application that can be developed by completely redesigning the application with the new WebSphere Process Server programming model in mind.

For example, the current migration tools do not migrate a collaboration into a Mediation Flow Component, which sometimes might be the best choice. The tools can be practical to accelerate the upgrade phase. However, the disadvantage is that it can negatively affect the maintainability and flexibility of the solution.

9.3.5 Performance of automatically migrated code

The following potential issues are related to the performance of the generated code by the standard migration tools:

- Custom Java code

Runtime support for the WebSphere InterChange Server APIs in WebSphere Process Server is provided by the Heritage API. This support allows direct portability of the WebSphere InterChange Server code into the WebSphere Process Server run time. During execution, in the background, “bridge code” is run to transform the WebSphere InterChange Server constructs (business objects) into WebSphere Process Server constructs (SDOs) and the reverse is also true. This code generates overhead during execution and can negatively affect the performance of the application.

- Keeping WebSphere Business Integration Adapters

The *two-step approach for adapters* keeps the WebSphere Business Integration Adapters during the first phase. The issue is that WebSphere Business Integration Adapters are better optimized to work with WebSphere InterChange Server than with WebSphere Process Server.

With WebSphere Process Server, WebSphere Business Integration Adapters generally consume more memory, because JMS transport must be used. JMS transport consumes more memory than the native Internet Inter-ORB Protocol (IIOP) transport that is available with the WebSphere InterChange Server. Also, more communication hops exist (several JVMs, intermediate Service Integration Bus layer). Therefore, you can expect performance impacts, depending on the usage of the adapter.

Archived

Archived



Part 3

Migration tooling

In this part, we cover the standard migration tools that are available for upgrading from WebSphere InterChange Server to WebSphere Process Server:

- ▶ In Chapter 10, “Overview of migration tools” on page 179, we provide an overview of the migration tools with WebSphere Integration Developer, the Migration Wizard, the reposMigrate utility, and the tools that are dedicated to the adapter migration.
- ▶ In Chapter 11, “Artifacts migration” on page 215, we provide more details about the standard migration tool support for server artifacts, provided by the Migration Wizard and reposMigrate tools.
- ▶ In Chapter 12, “Post-migration tasks” on page 253, we explain the post migration tasks to be executed for certain artifacts, such as database connection pools and relationships.
- ▶ In Chapter 13, “Best practices” on page 263, we provide best practices to ease the usage of the migration tools and to optimize the generated artifacts.

Overview of migration tools

In this chapter, we discuss the migration tools that are available for migration from IBM WebSphere InterChange Server to WebSphere Process Server. We begin with an overview of the WebSphere Integration Developer environment. Then we introduce the Migration Wizard, which is the main graphical tool to use when upgrading WebSphere InterChange Server artifacts in IBM WebSphere Integration Developer. Next, we describe the equivalent command-line tool, called **reposMigrate**. Finally, we focus on the adapter Migration Wizard.

We include the following sections:

- ▶ 10.1, “Migration tools” on page 180
- ▶ 10.2, “WebSphere Integration Developer” on page 181
- ▶ 10.3, “Migration Wizard” on page 192
- ▶ 10.4, “The reposMigrate utility” on page 205
- ▶ 10.5, “Migration tools specific to WebSphere Business Integration Adapters” on page 208

10.1 Migration tools

IBM provides migration tools that assist in migrating existing WebSphere InterChange Server and WebSphere Business Integration Adapter source artifacts into WebSphere Process Server artifacts. With WebSphere Process Server Version 6.2, the main part of the migration tools is embedded with the build-time and the runtime environments, that is, WebSphere Integration Developer and WebSphere Process Server. A separate tool is dedicated to help with the WebSphere Business Integration Adapters to WebSphere Adapters migration.

The migration tools can be used to support the migration implementation, but they are generally insufficient to provide a completely optimized migrated solution. See Part 2, “Migration implementation concepts” on page 71 and especially Chapter 9, “Migration implementation approach” on page 151 for more information about the migration implementation approach.

The technical implementation for the migration is only a part of the overall migration scope. See Part 1, “Migration concepts” on page 1, for more information about the overall migration strategy.

In the following sections, we review the migration tools.

10.1.1 Supported versions

The standard migration tools support the WebSphere InterChange Server product versions as summarized in Table 10-1.

Table 10-1 Overview of migration pathways

Migration pathways	Description
WebSphere InterChange Server Version 4.3	Use the migration tools that are provided with WebSphere Process Server.
WebSphere InterChange Server Version 4.2.2 and prior to Version 4.2.2	Migrate to Version 4.3 of WebSphere InterChange Server before migrating to WebSphere Process Server.

10.2 WebSphere Integration Developer

As an introduction to the embedded migration tools, which are the Migration Wizard and the reposMigrate utility, we provide an overview of the WebSphere Process Server build-time environment. 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, including all the tools and development aids that are necessary to build service-oriented architecture (SOA) and process integration solutions.

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. A primary purpose of WebSphere Integration Developer is to provide the appropriate tools to build and test Service Component Architecture (SCA)-based applications easily.

Eclipse: For more information about Eclipse and tutorials, see the following Web address and explore the Getting Started pages:

<http://www.eclipse.org>

In this section, we introduce the basic terms and concepts of WebSphere Integration Developer that are used in migrating the WebSphere InterChange Server V4.3 source artifacts. We include the following concepts in our discussion:

- ▶ Workbench
- ▶ Editors (assembly, process, business object, interface, interface mapping, mediation flow, business object map, relationship, and visual snippet)
- ▶ Module
- ▶ Shared library
- ▶ Integration test client

When a new workspace is started, the Welcome window is displayed as shown in Figure 10-1. From this window, you can access information, such as the product overview, tutorials, samples, migration information, and Web resources.

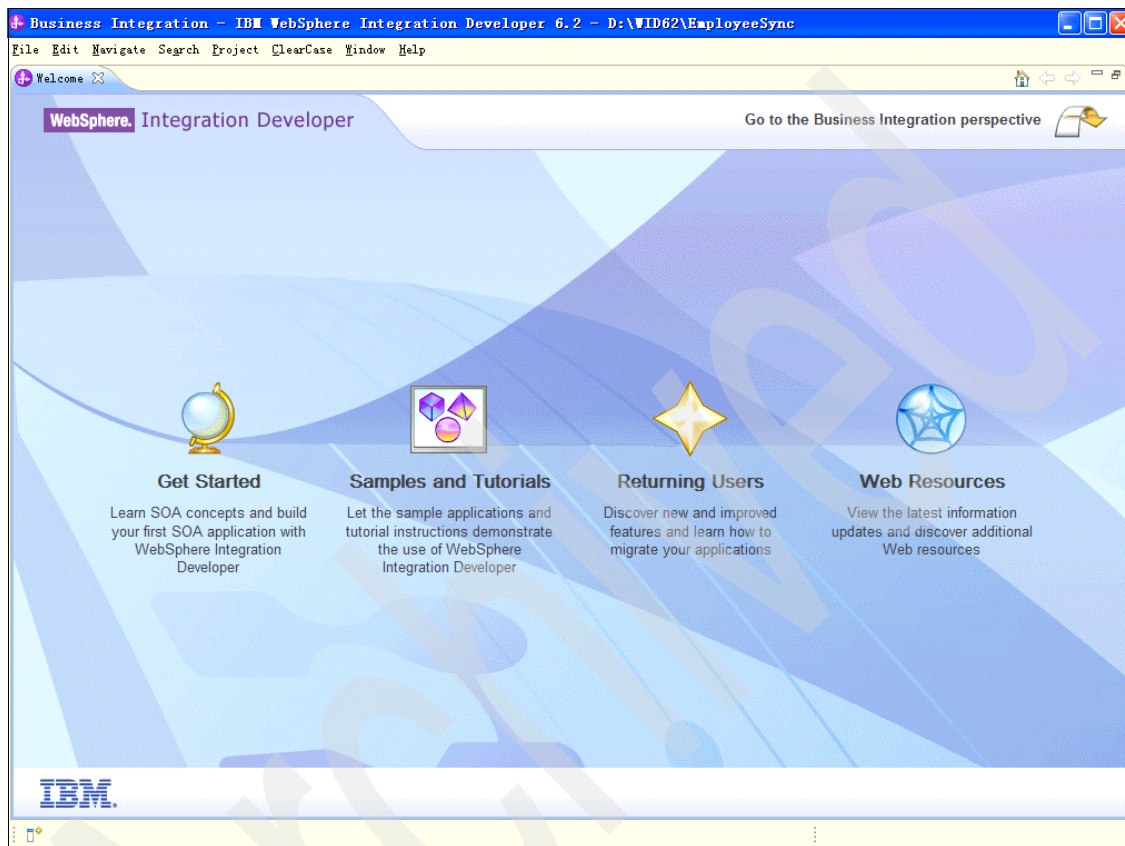


Figure 10-1 Welcome page

Close the Welcome page and click **Go to the Business Integration perspective** in the upper right corner.

The Business Integration perspective and workbench open as shown in Figure 10-2. In this window, you develop and author the migrated artifacts and other integration modules. The window offers a choice of perspectives and an array of toolbars and menu items that are used to accomplish a variety of tasks.

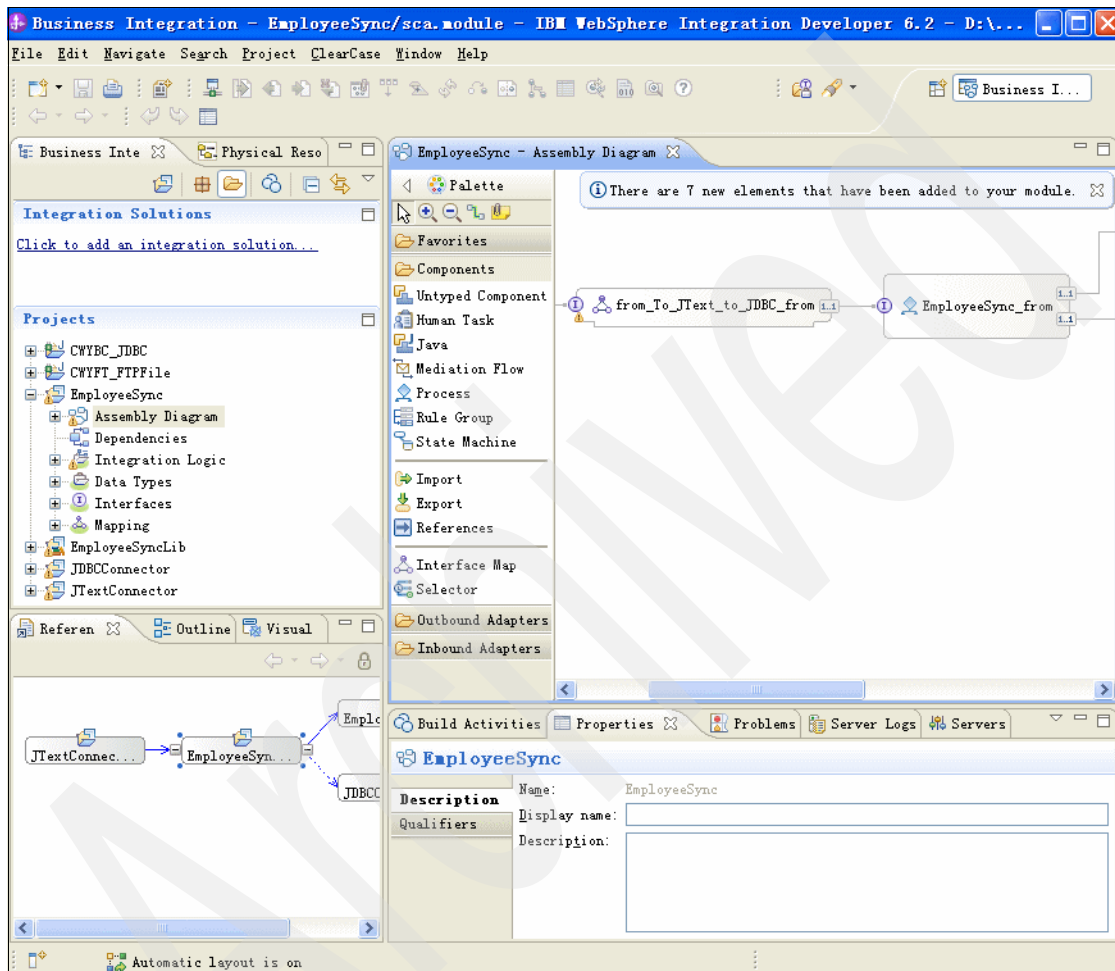


Figure 10-2 Workbench

10.2.1 Editors

An *editor* is a tool to create and to modify files. Depending on the type of file that is being edited, 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

You use the WebSphere Integration Developer *Assembly Editor* to build applications by assembling the SCA components. You drag SCA components, such as Business Process, components, imports, and exports, to the canvas. Then you specify their interfaces and bindings and wire them together by using the Assembly Diagram Editor as shown in Figure 10-3.

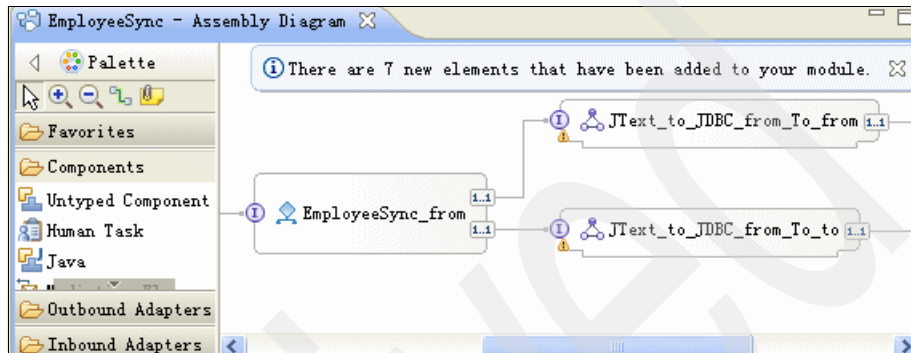


Figure 10-3 Assembly Editor

Business Object Editor

The *Business Object Editor* enables the building and editing of business objects, their attributes, and business graphs through a graphical interface as shown in Figure 10-4. You use this editor to add, delete, and reorder attributes and to change the type of an attribute.

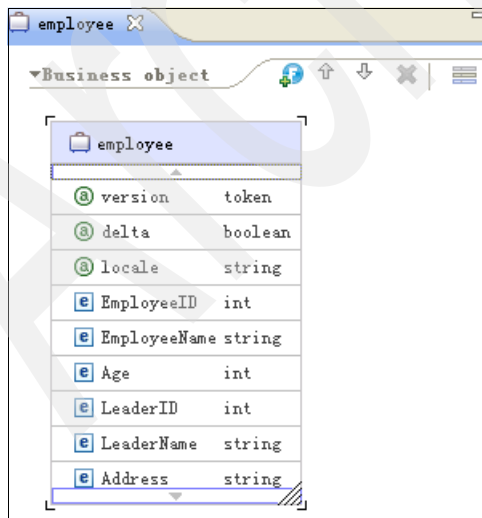


Figure 10-4 Business Object Editor

Process Editor

The *Process Editor* is a graphical programming environment, with which developers can visually create and manipulate business processes as shown in Figure 10-5.

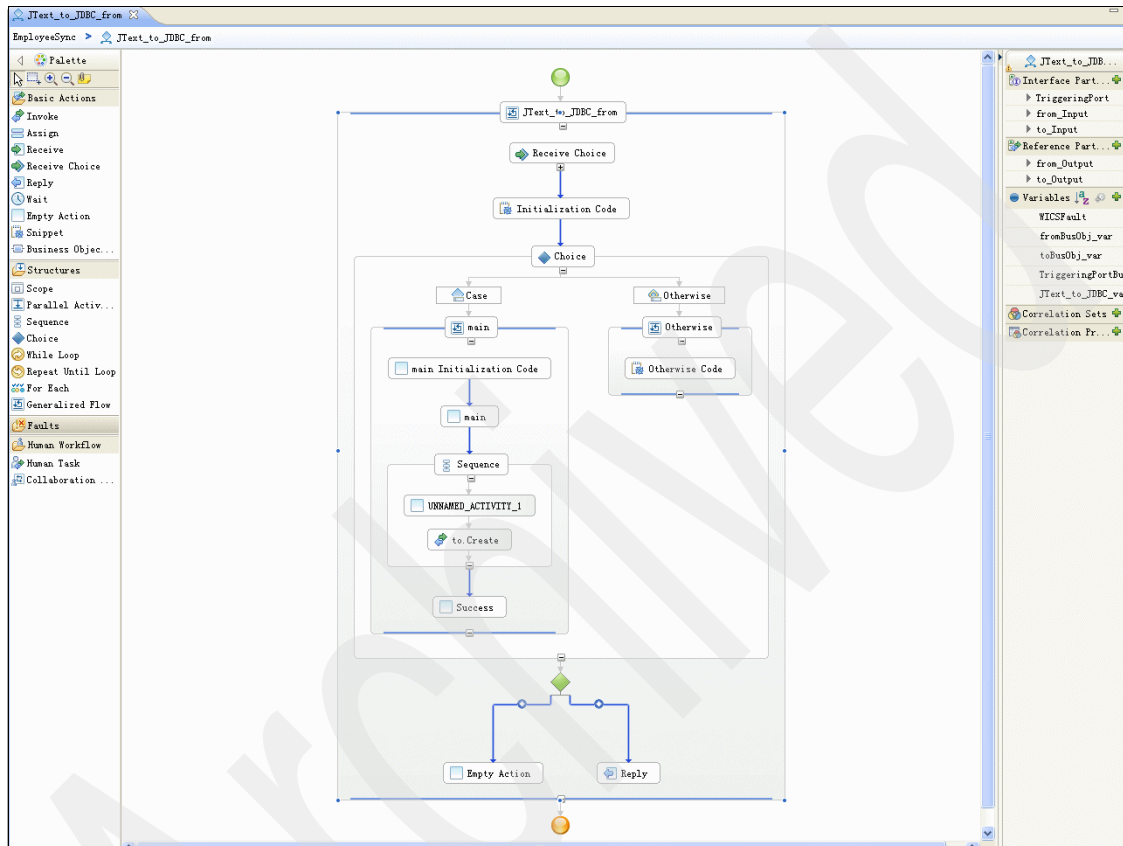


Figure 10-5 Process Editor

Interface Editor

The *Interface Editor* is used to build Web Services Description Language (WSDL) Port Type interfaces that are used to define SCA components as shown in Figure 10-6. You use this editor to add and to remove operations and specify operation inputs and outputs.

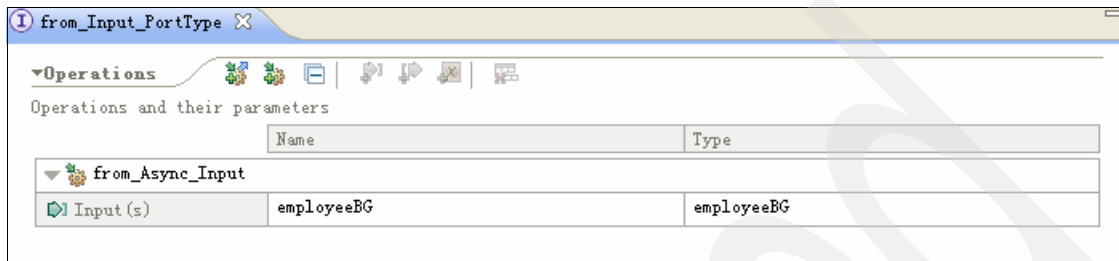


Figure 10-6 Interface Editor

Interface Mapping Editor

The *Interface Mapping Editor* enables you to build and edit interface maps through a graphical interface as shown in Figure 10-7. The Interface Mapping Editor provides the methodology to deal with the differences between interfaces and to reconcile them for your integrated development environment (IDE).

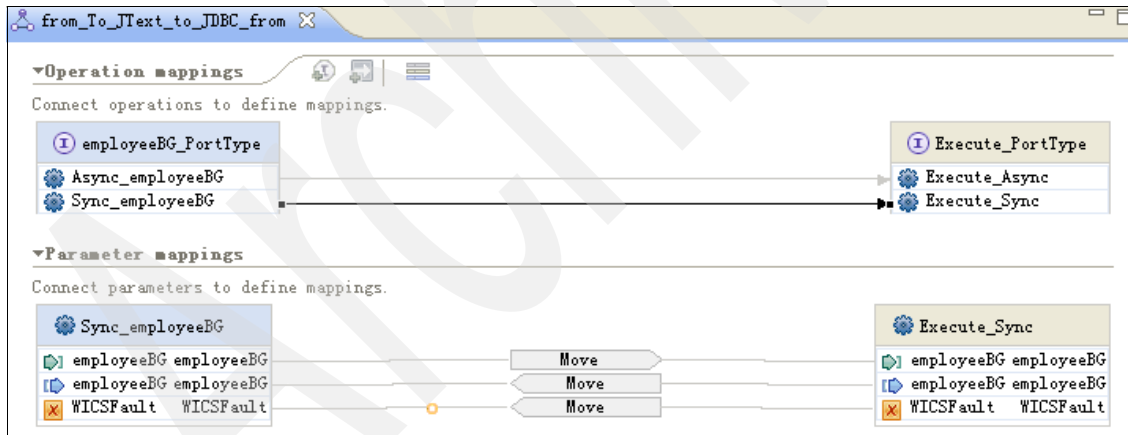


Figure 10-7 interface Mapping Editor

Mediation Flow Editor

The *Mediation Flow Editor* is a graphical programming environment that you use to visually create and manipulate mediation flows as shown in Figure 10-8 on page 187. You can visually compose a mediation flow by defining the source and target operations in the Operation connections section, and then visually

adding and wiring mediation primitives in the flow section. You then add properties for the primitives in the properties view.

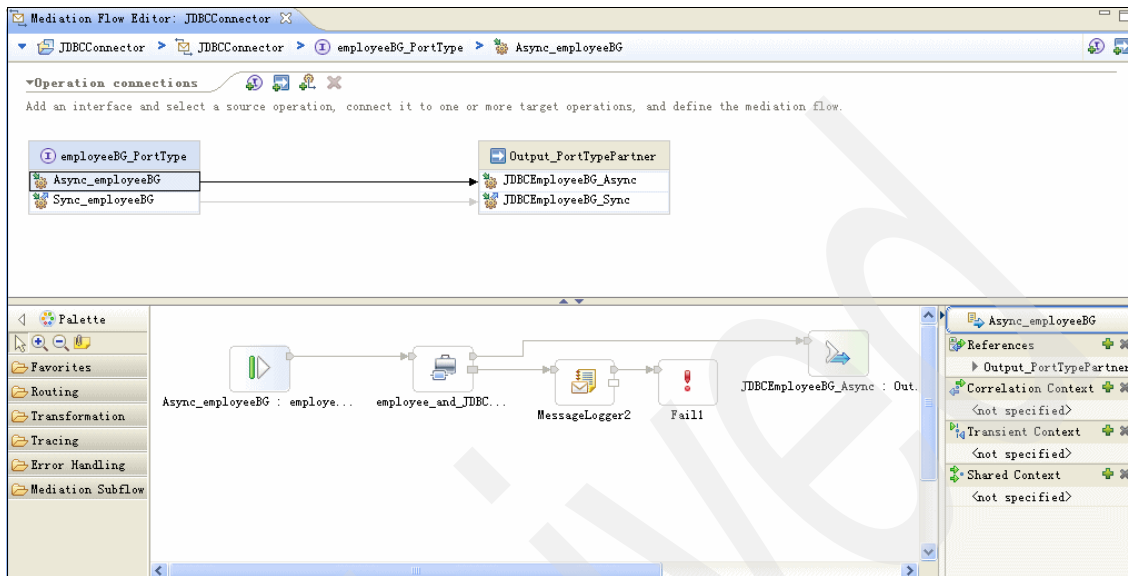


Figure 10-8 Mediation Flow Editor

Business Object Mapping Editor

The *Business Object Mapping Editor* is a graphical interface, with which you can build and edit business object maps and their attributes as shown in Figure 10-9 on page 188. This editor allows developers to define the mapping of data between business objects.

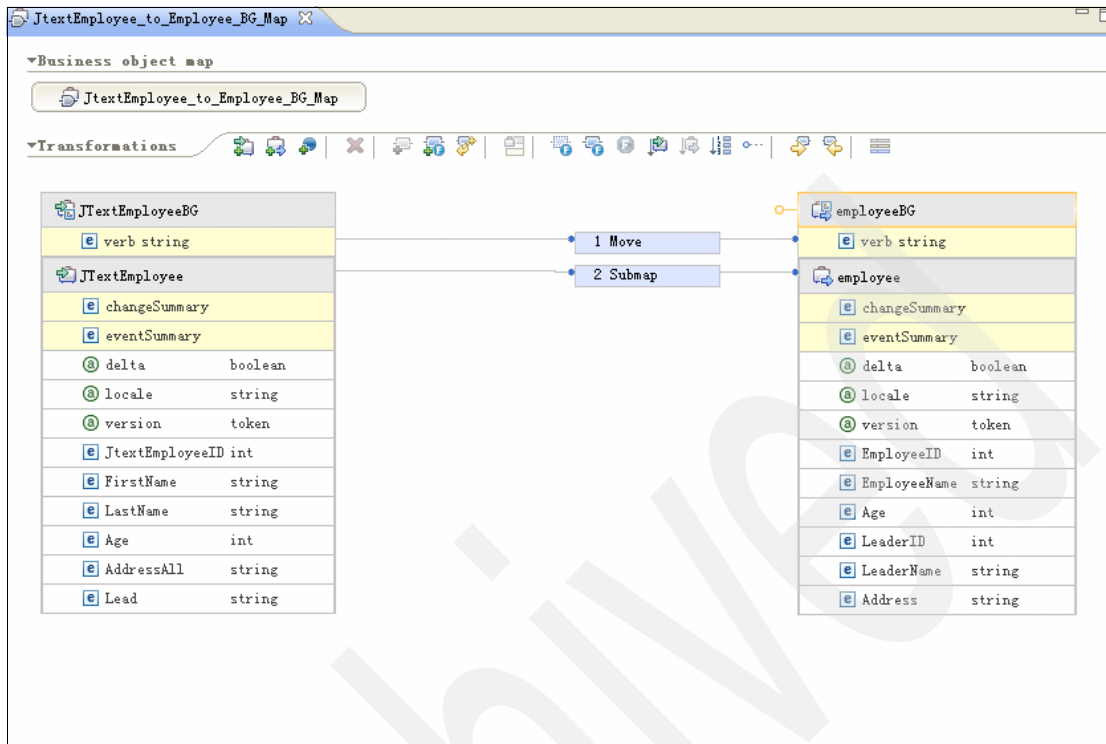


Figure 10-9 Business Object Mapping Editor

Relationship Editor

With the *Relationship Editor* in WebSphere Integration Developer, you can build and edit relationships and their attributes through a graphical interface as shown in Figure 10-10.

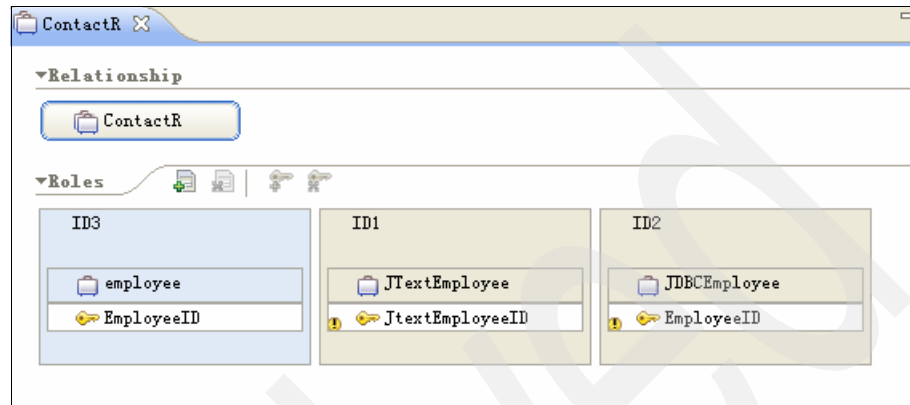


Figure 10-10 Relationship Editor

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 10-11 shows Java code in the Visual Snippet Editor.

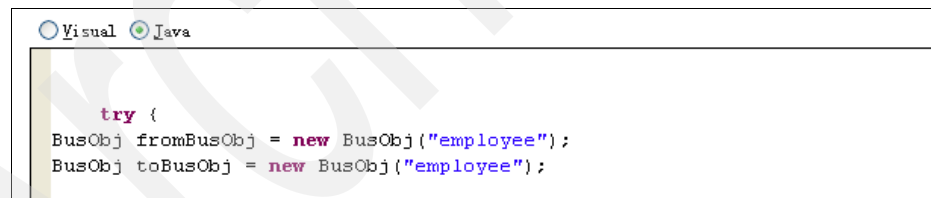


Figure 10-11 Visual Snippet Editor

10.2.2 Module

A *module* is a Business Integration project for developing SCA-based applications. It is the basic unit of deployment to WebSphere Process Server. A module is packaged in an enterprise archive (EAR) file and contains the following artifacts:

- ▶ SCA resources and module assembly
- ▶ Java 2 Platform, Enterprise Edition (J2EE) projects
- ▶ Java projects
- ▶ Dependent libraries

Dependent libraries, Java projects, and J2EE projects can be added to a module and deployed with the module.

10.2.3 Shared library

A *shared library* is a Business Integration project for storing artifacts that are shared between multiple modules. In order for a module to use the resources from a library, it must be added as a dependent to the module by using the Dependencies Editor as shown in Figure 10-12.



Figure 10-12 Dependencies Editor

A library cannot be deployed by itself. However, a library can be added to the module and selected to be deployed with the module. At run time, libraries are not shared, but they are deployed with the module that depends on the library. Also, library dependencies can be added to a library.

10.2.4 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 that enables you to easily manage and precisely control your tests.

Figure 10-13 shows the integration test client.

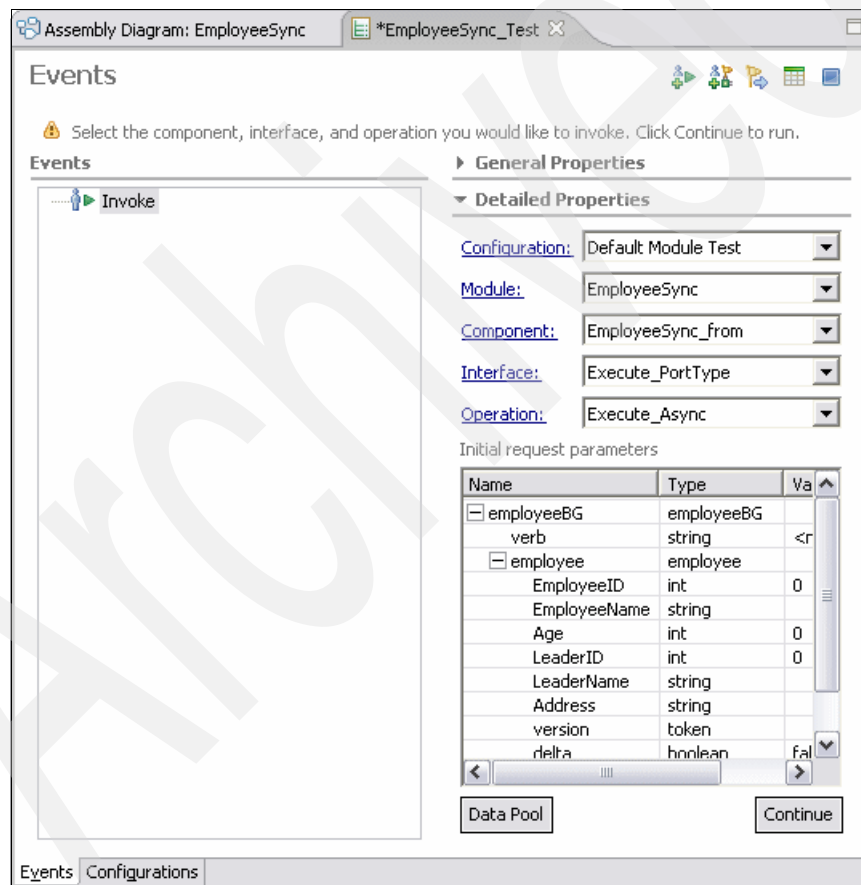


Figure 10-13 Integration test client

10.3 Migration Wizard

Migration from WebSphere InterChange Server to WebSphere Process Server is supported by using the Migration Wizard that is available in WebSphere Integration Developer. The wizard generates the migrated artifacts in WebSphere Integration Developer as modules, libraries, and components so that you can review the artifacts and modify them if necessary.

10.3.1 Using the Migration Wizard

WebSphere Integration Developer provides a Migration Wizard for migrating the WebSphere InterChange Server artifacts. You can start the Migration Wizard in one of two ways:

1. From the File menu option, either select **File** → **Import**, or, in the Import window (Figure 10-14 on page 193), select **WebSphere InterChange Server Repository**, and then click **Next** to invoke the Migration Wizard.

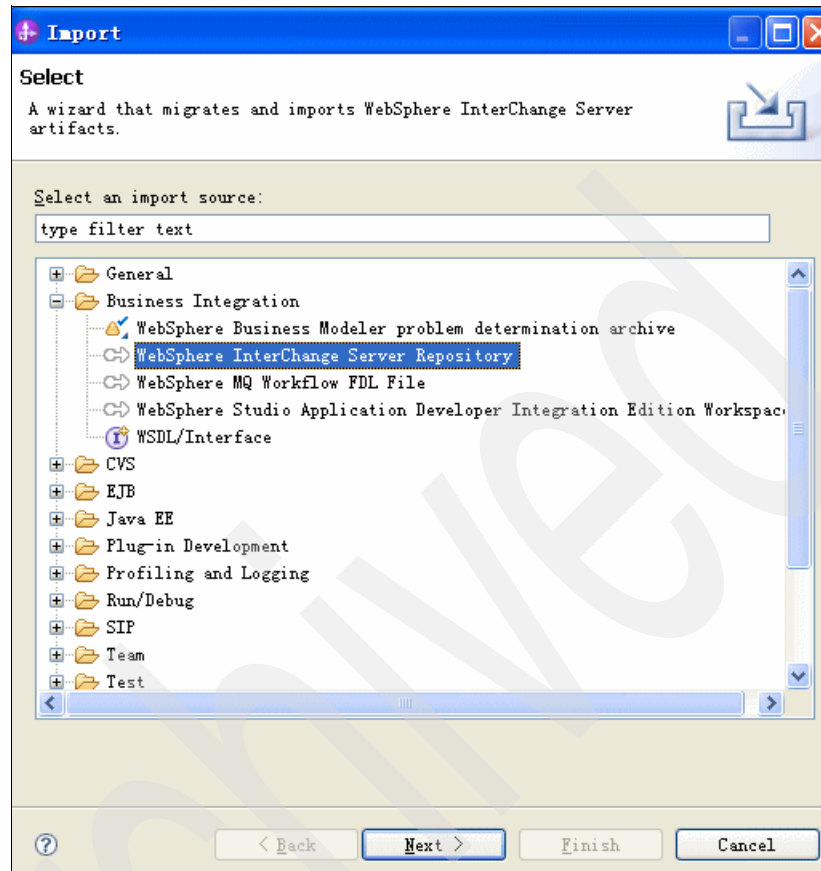


Figure 10-14 Migration Wizard

2. From the Help menu option:
 - a. Select **Help** → **Welcome**.
 - b. In the welcome window in WebSphere Integration Developer, click the **Returning Users** icon (Figure 10-15 on page 194).

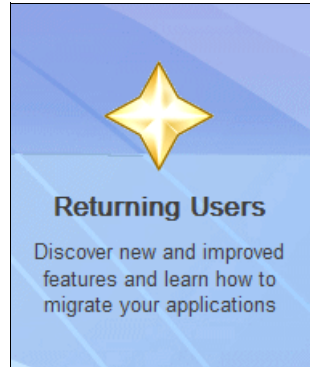


Figure 10-15 Returning users icon

3. In the Returning Users window (Figure 10-16 on page 195), in the left pane, click **Migration**.

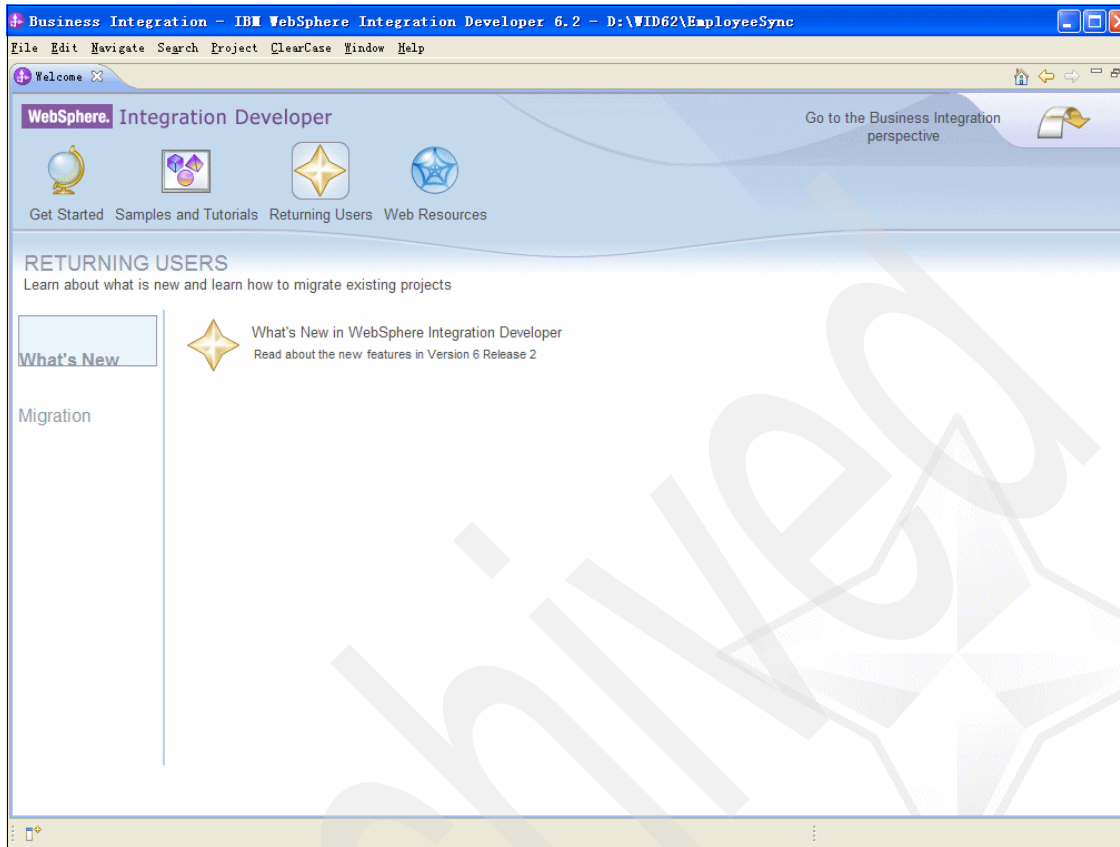


Figure 10-16 Returning users page

4. In the Migration pane (Figure 10-17), click **Migrate a WebSphere ICS repository**.

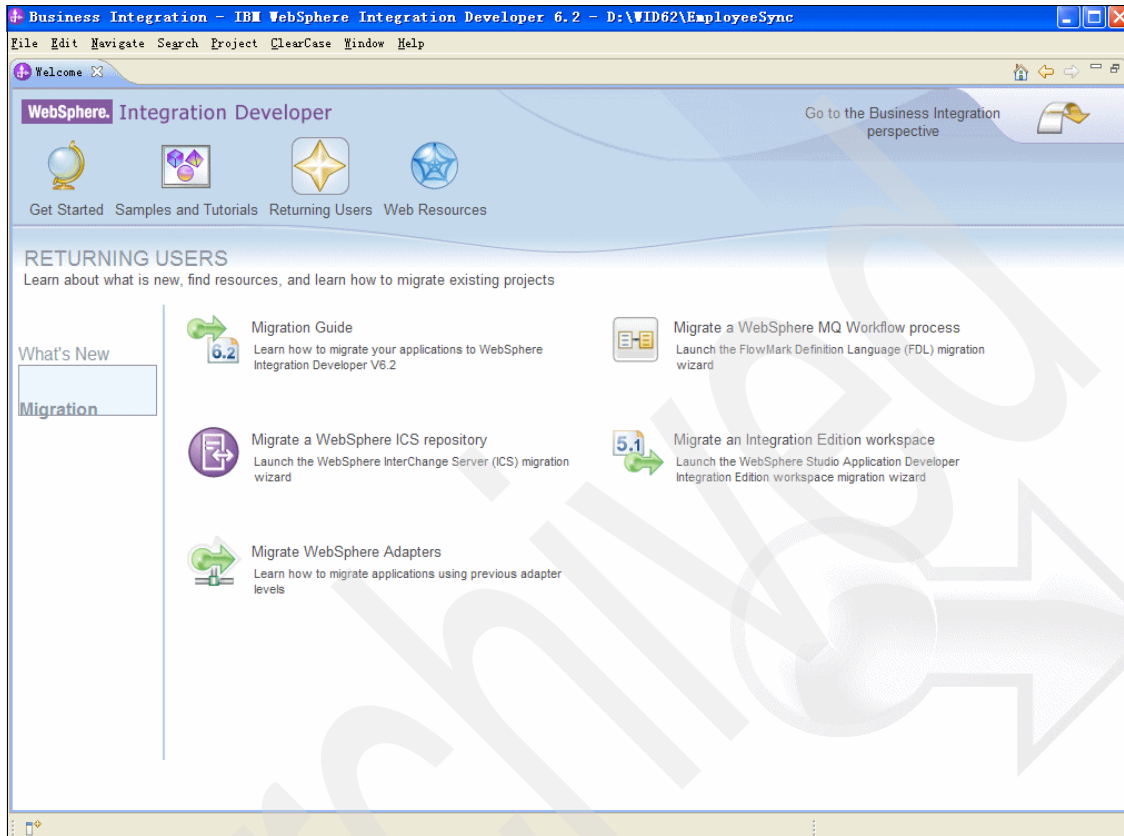


Figure 10-17 Migration page

5. The Migration Wizard window (Figure 10-18 on page 197) opens by using either approach.

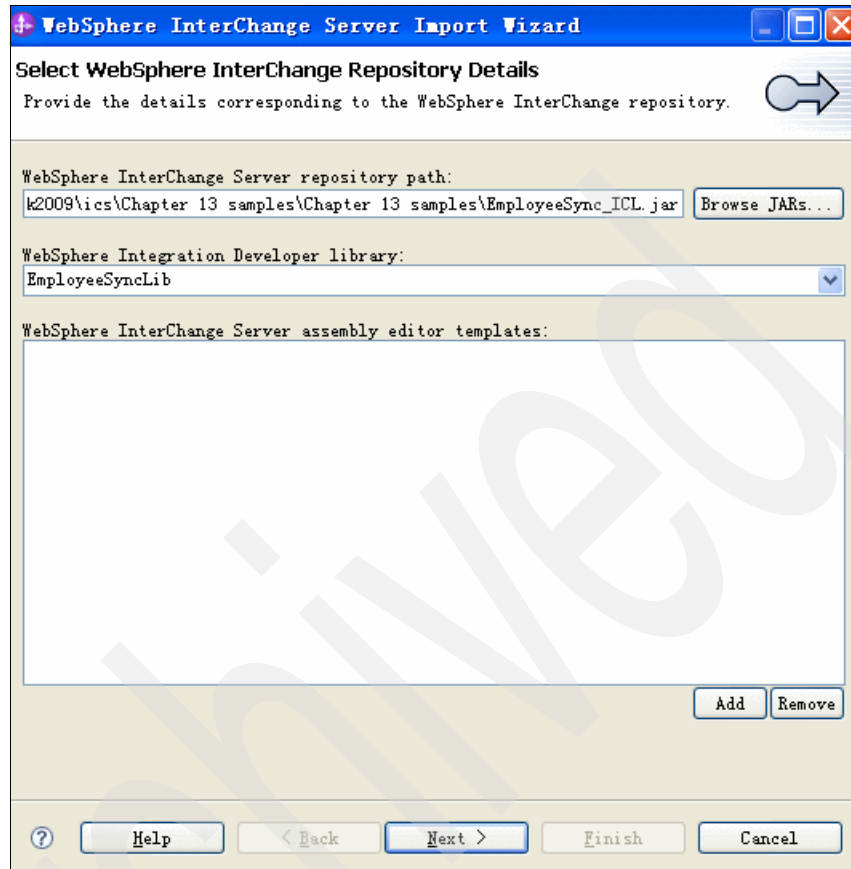


Figure 10-18 Select WebSphere InterChange Repository Details

6. In Figure 10-18:
 - a. Enter the WebSphere InterChange Server repository path or click Browse JARs to select the JAR file to use in the migration process.
 - b. Enter the name of a new or existing WebSphere Integration Developer library.

You can also add a WebSphere InterChange Server Assembly Editor template to be loaded and used for XML to Java conversion. If custom APIs (My Library) are created for use in the Activity Editor, the migration tool must refer to the custom activity editor template to determine how to migrate the custom API to Java. These templates can be found in the root directory `.wbiActEditorSettings` located in the WebSphere InterChange Server workspace directory. Generally, the template files have a `.bbt` file extension. Ensure that you load these templates in the template box. If

you do not add a template, the Standard Assembly Editor Template V4.3.3 will be used for XML to Java conversion.

- c. Click **Next** to configure the migration settings for each connector.
7. From this page as shown in Figure 10-19, you can select the appropriate target binding and data handlers for each connector being migrated. Select the connector from the list and choose a target binding and then a custom data handler JAR file if it applies. Note that the Migration Wizard will not allow the migration process to proceed until all connector values are set.

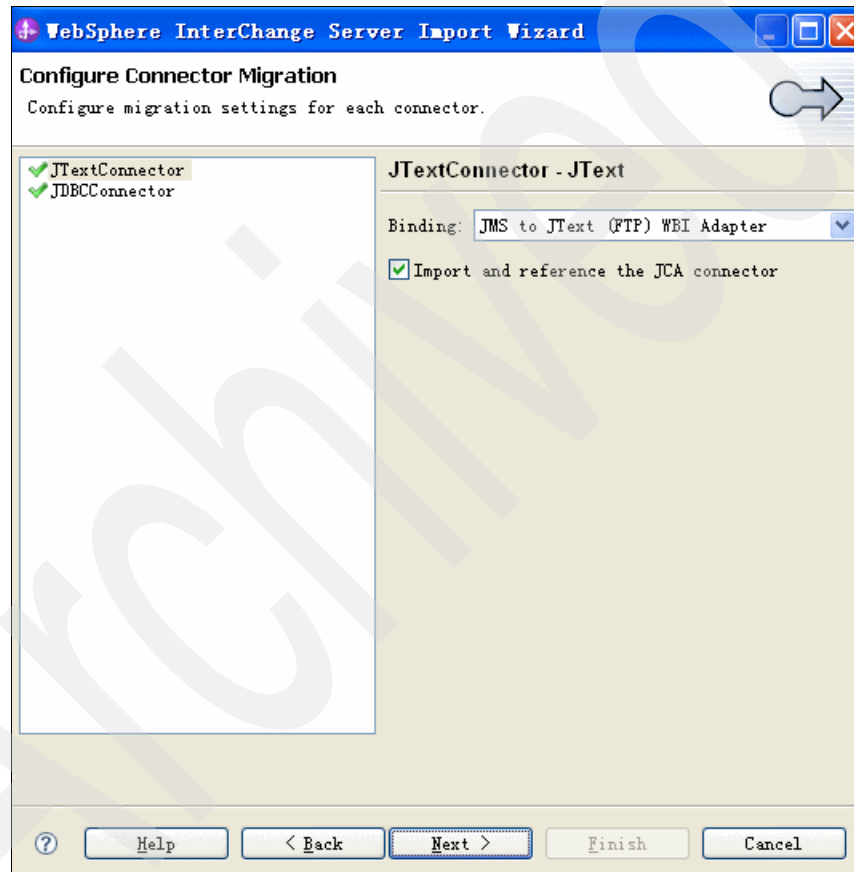


Figure 10-19 Configure Connector Migration

Note the adapter migration is a two-step process. The first step is represented by the Configure Connector Migration window (Figure 10-19), and it consists of defining the target implementation for the adapter. The first step is a prerequisite for the second step.

Here are more details about the two steps for adapter migration:

- The first step allows you to keep and reuse the old WebSphere Business Integration Adapter and can provide a Java Message Service (JMS) binding for this purpose.
- The first step directly converts WebSphere Business Integration Adapters, which have equivalent native binding counterparts in WebSphere Process Server. This step applies to the following WebSphere Business Integration Adapters: MQ, JMS, HTTP, Web Services, and Enterprise JavaBeans (EJBs). More details are given in Table 10-2 on page 200 and also in 12.3, “Bindings” on page 258.
- The first step cannot directly convert WebSphere Business Integration Adapters that have a WebSphere Java EE Connector Architecture (JCA) adapter counterpart in WebSphere Process Server. But, it is able to prepare the work for step two by importing the necessary JCA adapter libraries.
- The second step converts WebSphere Business Integration Adapters that have a WebSphere JCA adapter counterpart in WebSphere Process Server. It currently supports the following WebSphere Business Integration Adapters: Java Database Connectivity (JDBC), Email, JText, SAP, and PeopleSoft. More adapters will be covered with upcoming fix packs.
- The second step is done later, using a dedicated adapter Migration Wizard, after the first step of the migration is completed. It is described in detail in “Using the Migration Wizard” on page 192.

Table 10-2 on page 200 summarizes the migration options proposed by the migration tools for WebSphere Business Integration Adapters.

Table 10-2 Migration options for WebSphere Business Integration Adapters

WebSphere Business Integration Adapter type	Migration options
WebSphere Business Integration Adapter for JMS	Option 1: WebSphere Process Server JMS binding (covered by step 1) Option 2: WebSphere Process Server generic JMS binding (covered by step 1) Option 3: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for JMS (covered by step 1)
WebSphere Business Integration Adapter for MQ	Option 1: WebSphere Process Server MQ binding (covered by step 1) Option 2: WebSphere Process Server MQ JMS binding (covered by step 1) Option 3: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for MQ (covered by step 1)
WebSphere Business Integration Adapter for WebServices	Option 1: WebSphere Process Server HTTP binding (covered by step 1). Note: Migration to Web Services binding is targeted for an upcoming fix pack. Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for WebServices (covered by step 1)
WebSphere Business Integration Adapter for HTTP	Option 1: WebSphere Process Server HTTP binding (covered by step 1) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for HTTP (covered by step 1)
WebSphere Business Integration Adapter for EJB	Option 1: WebSphere Process Server Stateless Session Bean (covered by step 1) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for EJB (covered by step 1)

WebSphere Business Integration Adapter type	Migration options
WebSphere Business Integration Adapter for JDBC	Option 1: WebSphere Adapter for JDBC (covered by steps 1 and 2) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for JDBC (covered by step 1)
WebSphere Business Integration Adapter for JText	Option 1: WebSphere Adapter for FlatFile (covered by steps 1 and 2) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for JText (covered by step 1)
WebSphere Business Integration Adapter for EMail	Option 1: WebSphere Adapter for EMail (covered by steps 1 and 2) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for EMail (covered by step 1)
WebSphere Business Integration Adapter for SAP	Option 1: WebSphere Adapter for SAP (covered by steps 1 and 2) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for SAP (covered by step 1)
WebSphere Business Integration Adapter for PeopleSoft	Option 1: WebSphere Adapter for PeopleSoft (covered by steps 1 and 2) Option 2: WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter for PeopleSoft (covered by step 1)
Any other WebSphere Business Integration Adapter	WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter (covered by step 1). IBM plans to support the migration of more adapters in upcoming fix packs.

In Figure 10-19 on page 198, click **Next**.

8. The Conversion Options page opens. From here, you can accept the recommended options or change them as shown in Figure 10-20.

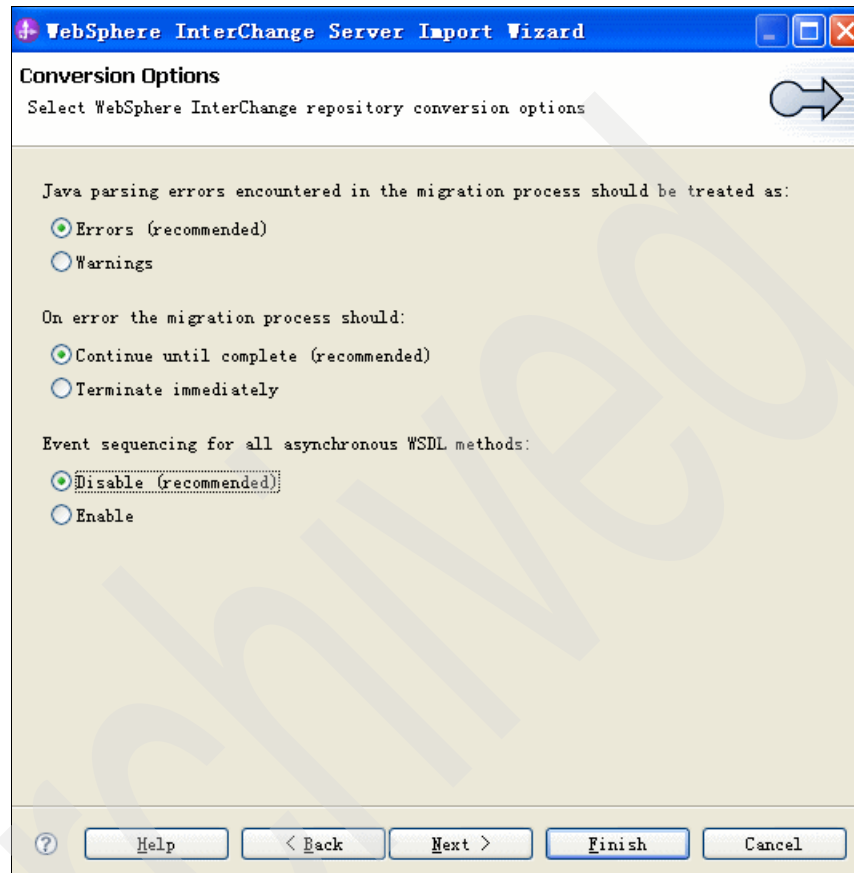
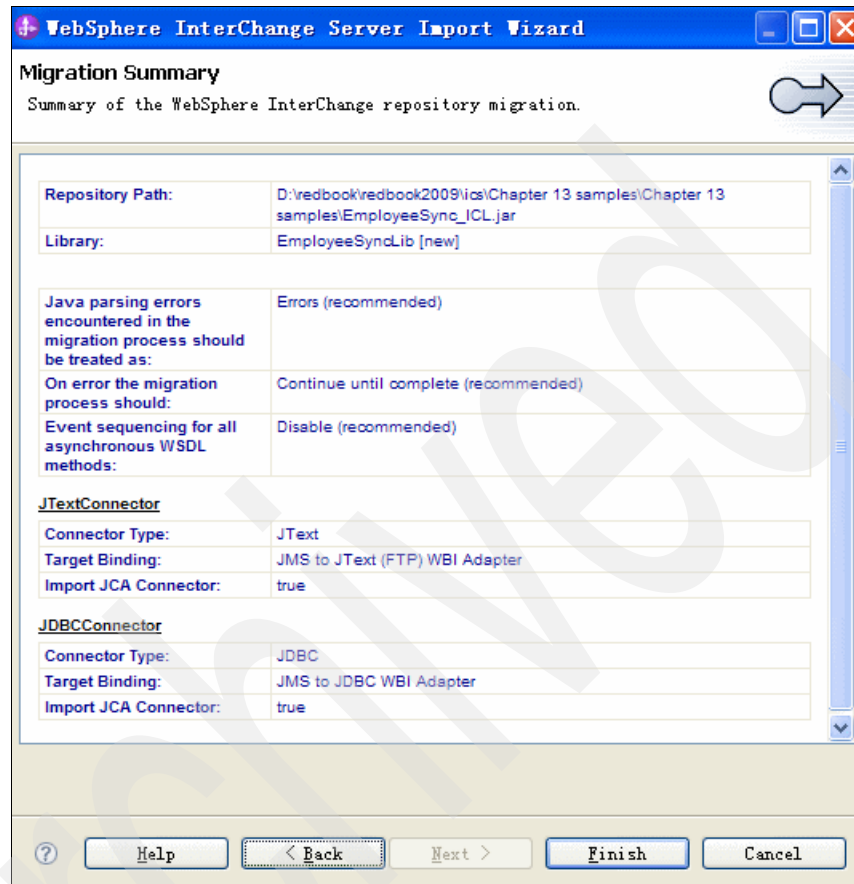


Figure 10-20 Conversion Options

Click **Next** on Figure 10-20.

9. A Migration Summary page opens similar to Figure 10-21.



The image shows a screenshot of the 'WebSphere InterChange Server Import Wizard' window, specifically the 'Migration Summary' step. The window has a blue title bar with the text 'WebSphere InterChange Server Import Wizard' and standard Windows window controls. Below the title bar, the text 'Migration Summary' is displayed, followed by a subtitle 'Summary of the WebSphere InterChange repository migration.' and a right-pointing arrow icon.

The main content area contains several sections:

- Repository Path:** D:\redbook\redbook2009\ics\Chapter 13 samples\Chapter 13 samples\EmployeeSync_ICL.jar
- Library:** EmployeeSyncLib [new]
- Java parsing errors encountered in the migration process should be treated as:** Errors (recommended)
- On error the migration process should:** Continue until complete (recommended)
- Event sequencing for all asynchronous WSDL methods:** Disable (recommended)
- JTextConnector**
 - Connector Type:** JText
 - Target Binding:** JMS to JText (FTP) WBI Adapter
 - Import JCA Connector:** true
- JDBCCConnector**
 - Connector Type:** JDBC
 - Target Binding:** JMS to JDBC WBI Adapter
 - Import JCA Connector:** true

At the bottom of the window, there is a row of buttons: a question mark icon, 'Help', '< Back' (disabled), 'Next >' (disabled), 'Finish' (highlighted), and 'Cancel'.

Figure 10-21 Migration Summary

10. After you have reviewed the summary details, click **Finish** to begin the migration process.

11. A progress bar at the bottom of the migration dialog indicates the progress of the migration. When the process completes, the dialog disappears and the Migration Results window opens as shown in Figure 10-22.

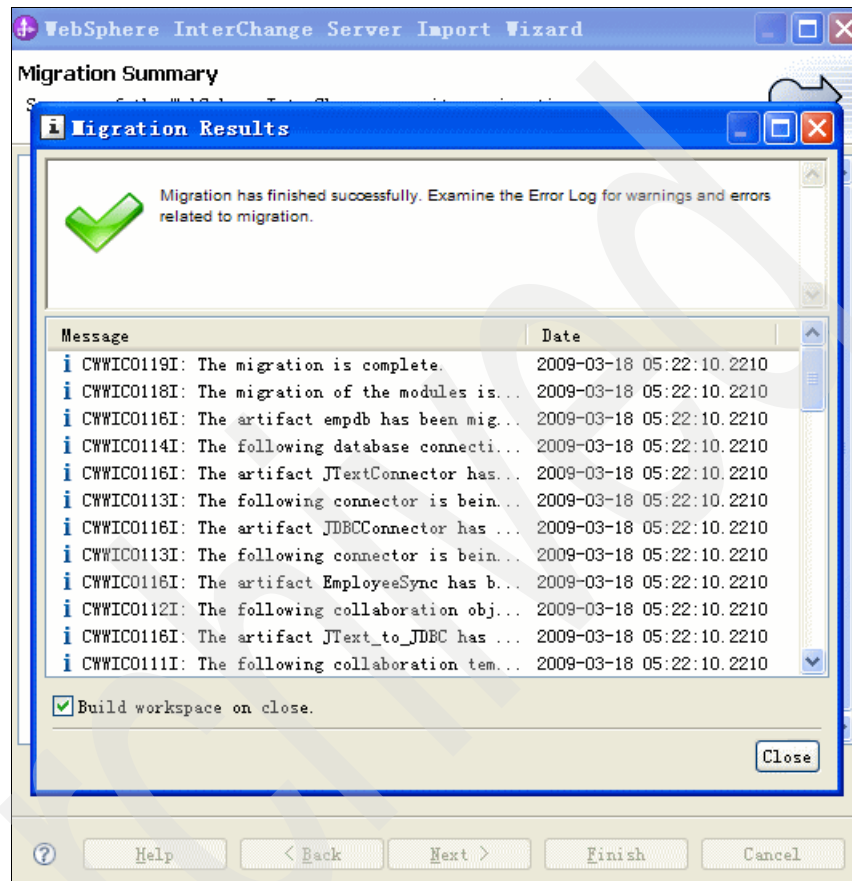


Figure 10-22 Migration Results

Click **Close** to finish the process and to build the new workspace. You can clear the check box “Build workspace on close” if you do not want the workspace to build at this time.

12. At this stage, you have finished the first step of the migration. You have to apply the second step to completely migrate WebSphere Business Integration Adapters that have a WebSphere JCA adapter counterpart. We describe the second step in “Migrating using the Adapter Migration Wizard” on page 211.

10.4 The reposMigrate utility

The **reposMigrate** utility is the scripting equivalent to the graphical Migration Wizard. You can invoke the **reposMigrate** utility from a command line. In the following section, we explain how to use the **reposMigrate** utility during the migration in more detail.

10.4.1 Migrating by using the reposMigrate utility

The **reposMigrate** utility is in the `bin` directory of WebSphere Process Server runtime installation. You invoke this utility from a command line.

To use the **reposMigrate** utility:

1. Identify the Java archive (JAR) file that contains the pre-exported WebSphere InterChange Server artifacts that are to be converted to WebSphere Process Server deployable artifacts.
2. Invoke the **reposMigrate** utility from a command-line prompt. Specify the required arguments and any optional arguments that are required.
3. Modify the resulting JAR file if applicable.
4. Run the **serviceDeploy** command to create a deployable EAR file for each JAR file.
5. Use the **wsadmin** command to install the EAR files on WebSphere Process Server. By invoking **wsadmin** with the generated Jython script, resources are created in the WebSphere Process Server system for all target resources, such as JDBC data sources and WBI Scheduler entries.

The **reposMigrate** utility requires a WebSphere InterChange Server repository JAR file as input. This JAR file needs to be self-contained with respect to the applications that are being migrated. That is, all artifacts referenced by any of the artifacts in the JAR file must also be contained in the JAR file.

Export these artifacts and create the JAR file by using the WebSphere InterChange Server **repos_copy** command with the **-o** option as shown in Example 10-1.

Example 10-1 Exporting WebSphere InterChange Server artifacts by using the repos_copy command

```
repos_copy.bat -sWICS43Server -uadmin -pnull  
-eCollaboration:EmployeeSync+ConnectionPool:empdb+Relationship:ContactR  
-oEmployeeSync.jar -deep
```

See “Using repos_copy” in the IBM WebSphere InterChange Server V4.3 information center at the following address for more details about the **repos_copy** command:

http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wics_user_administrator.doc/doc/administration/administration33.htm

The **reposMigrate** utility converts all of the WebSphere InterChange Server artifacts in a repository JAR file into WebSphere Process Server deployable artifacts.

For each migrated collaboration object and connector definition, a JAR file is created by the **reposMigrate** utility as shown in Figure 10-23. For other artifacts, such as business objects, maps, and relationships, a copy of the artifacts that are generated from the input JAR file is included in each JAR file. If no collaboration objects or connectors are migrated, a single JAR file is created that contains a module of all the shared artifacts.

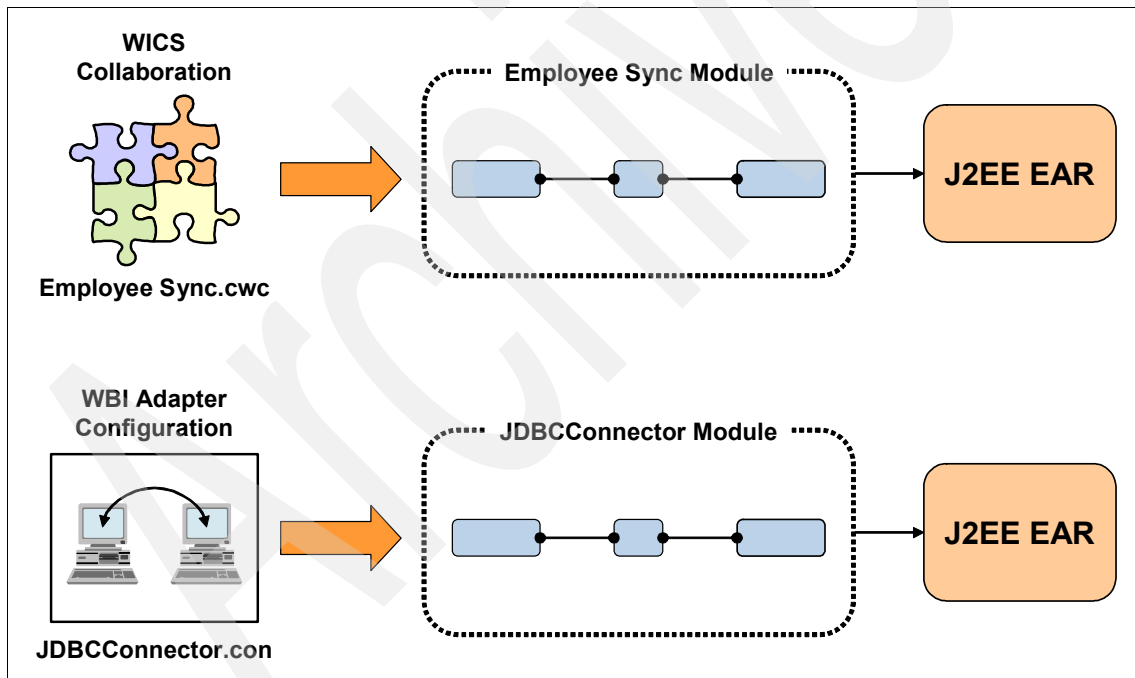


Figure 10-23 Artifact conversion

After the new JAR files are created, use the **serviceDeploy** command to deploy these artifacts to the WebSphere Process Server. The **serviceDeploy** command utility for WebSphere Process Server is used to compile and package the SCA

and J2EE artifacts programmatically. The **reposMigrate** utility receives the WebSphere InterChange Server JAR file, creates the SCA components, and then invokes the **serviceDeploy** utility to create the J2EE EAR file.

For WebSphere InterChange Server artifacts that have no corresponding artifact in WebSphere Process Server, a Jython script is generated during migration that can be run by using the **wsadmin** command to create WebSphere Process Server configuration definitions that correspond to the original WebSphere InterChange Server artifacts. The **wsadmin** tool in WebSphere Application Server V6 runs the administrative scripts.

The **reposMigrate** utility has the following syntax:

```
reposMigrate [-options] input repository module output directory
```

Note the following explanation:

- ▶ The *input repository* indicates the input JAR file.
- ▶ The *module output directory* indicates the output file directory.

Table 10-3 shows the command-line options for the **reposMigrate** utility.

Table 10-3 Command-line options of the *reposMigrate* utility

Command-line option	Explanation
-lv	Sets the log level to verbose.
-wi	Ignores Java conversion errors (only displays warnings).
-fh	Halts the migration at first failure.
-es	Enables event sequencing on all generated artifacts.
-ml	Maintains a loop structure when migrating collaboration templates.
-td <i>template directory</i>	Specifies the directory that contains the custom template files.

The following example shows a successful **reposMigrate** command:

```
reposmigrate Custom.jar C:\Modules -lv -wi
```

Important: The **reposMigrate** command only supports migrating a WebSphere Business Integration Adapter to WebSphere Process Server JMS binding connecting to the existing WebSphere Business Integration Adapter. In other words, you have to use WebSphere Integration Developer to migrate the WebSphere Business Integration Adapter to an equivalent WebSphere Process Server native binding or JCA adapter.

10.5 Migration tools specific to WebSphere Business Integration Adapters

The WebSphere Business Integration Adapter Artifact Importer and the Adapter Migration Wizard are two separate tools that are dedicated to helping with the migration of WebSphere Business Integration Adapters.

You can use the Adapter Artifacts Importer to import a WebSphere Business Integration Adapter definition into WebSphere Integration Developer. Therefore, the WebSphere Business Integration Adapter can be reused with a WebSphere Process Server environment through Java Message Service (JMS) communication. The functionality provided by this tool is similar to what is available in the Migration Wizard. The only difference is that the Adapter Artifacts Importer relies on the adapter configuration file (.cfg) and the schemas of the business object definitions as separate files. The Migration Wizard relies on the definitions that are stored in the repos_copy JAR file.

Generally, the Adapter Artifacts Importer is not needed for a WebSphere InterChange Server project migration, because the Migration Wizard already does everything that is needed. It is useful for projects that use different broker products, such as WebSphere Business Integration Server Foundation or WebSphere Business Integration Message Broker.

The Adapter Migration Wizard is extremely useful to migrate WebSphere Business Integration Adapters that have a WebSphere JCA adapter counterpart. However, the Adapter Migration Wizard cannot migrate all kinds of WebSphere Business Integration Adapters to all kinds of WebSphere JCA adapters. The list of supported adapters is given in Table 10-2 on page 200. The Adapter Migration Wizard can be run on any WebSphere Process Server module that contains WebSphere Business Integration Adapter artifacts. It updates specific artifacts, such as the schemas and service definition files (.import, .export, .xsd, and .wsdl), for use with the new JCA adapter. When you finish running the Adapter Migration wizard, the migration is complete and the module uses a new WebSphere JCA adapter.

Before running the Adapter Migration Wizard, run the WebSphere InterChange Server Migration Wizard to get project modules ready for the adapter upgrade.

10.5.1 WebSphere Business Integration Adapter Artifact Importer

WebSphere Business Integration Adapter Artifact Importer is provided by WebSphere Integration Developer to clients, who only want to create WebSphere Process Server configurations for the existing WebSphere Business

Integration Adapter. These clients do not want to migrate the entire repository or project to WebSphere Process Server.

To start WebSphere Business Integration Adapter Artifact Importer:

1. In WebSphere Integration Developer, select **File** → **New** → **External Service**.
2. In the External Service window (Figure 10-24), click **Messaging**, select **WebSphere Business Integration Adapters**, and click **Next**.

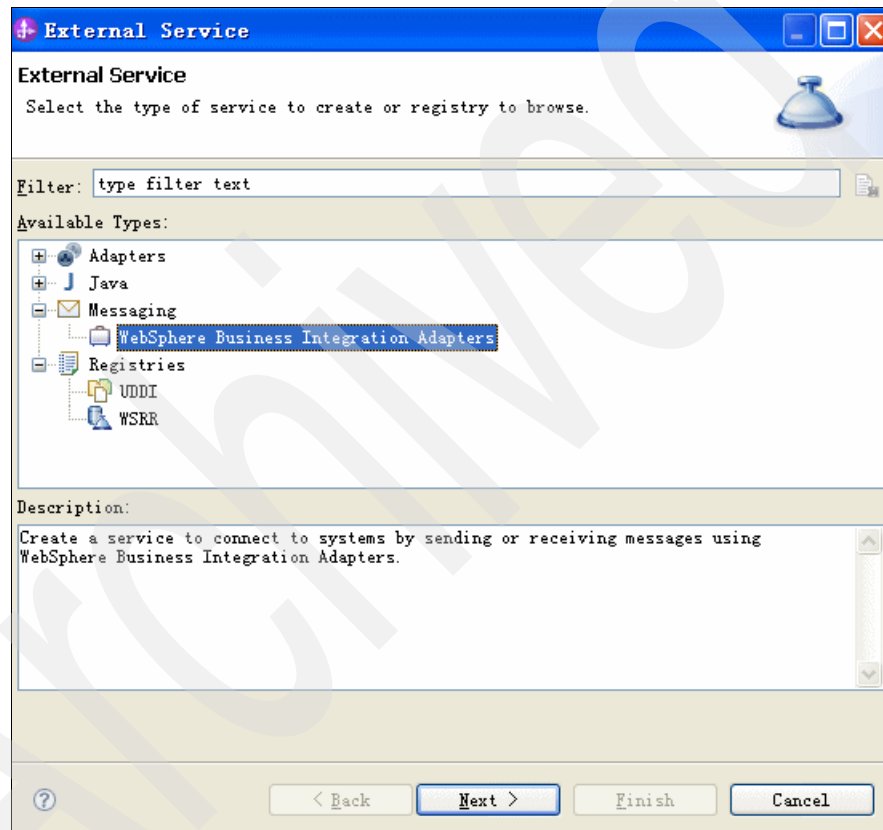


Figure 10-24 External Service

3. In the Select a Messaging Agent window (Figure 10-25), select the **WBI Adapter Artifact Importer**, and click **Next**.

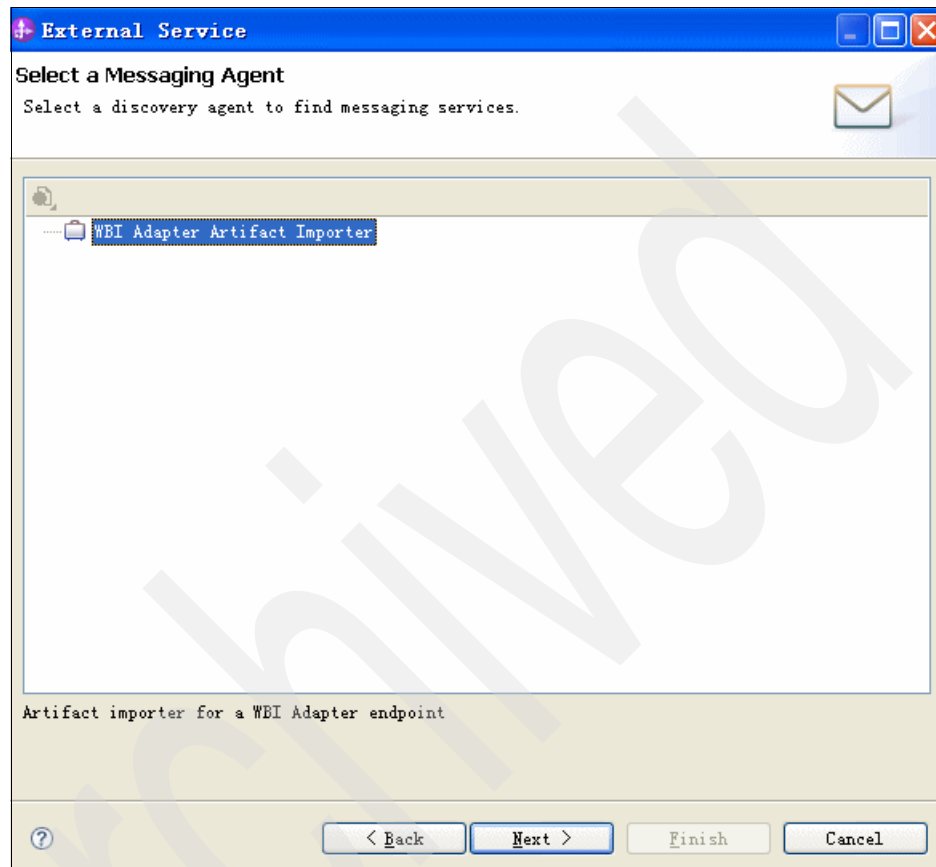


Figure 10-25 Select a Messaging Agent

4. In the Discovery Configuration window (Figure 10-26), click **Browse**, and specify the connector configuration file and business object schema directory. Then, click **Finish**.

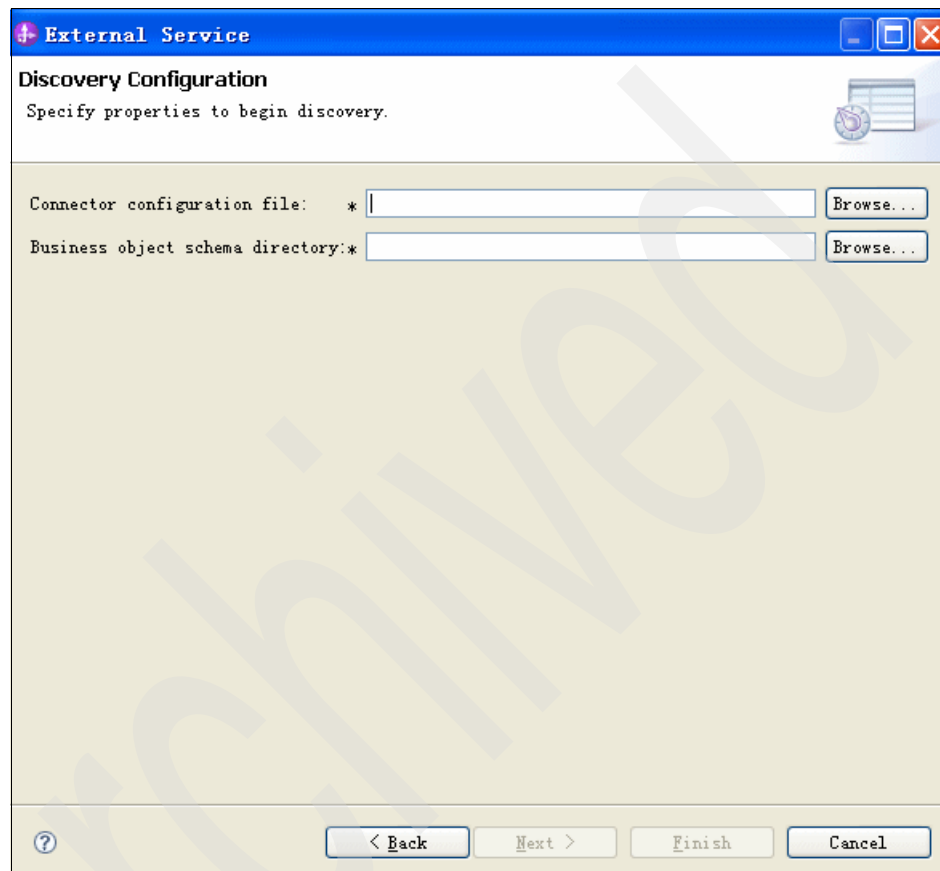


Figure 10-26 Discovery Configuration

10.5.2 Migrating using the Adapter Migration Wizard

In the WebSphere Integration Developer workspace, you can launch the Adapter Migration Wizard by right-clicking the module project in the Business Integration perspective and selecting **Update** → **Migrate Adapter Artifacts**. Figure 10-27 on page 212 describes the functional areas of the wizard.

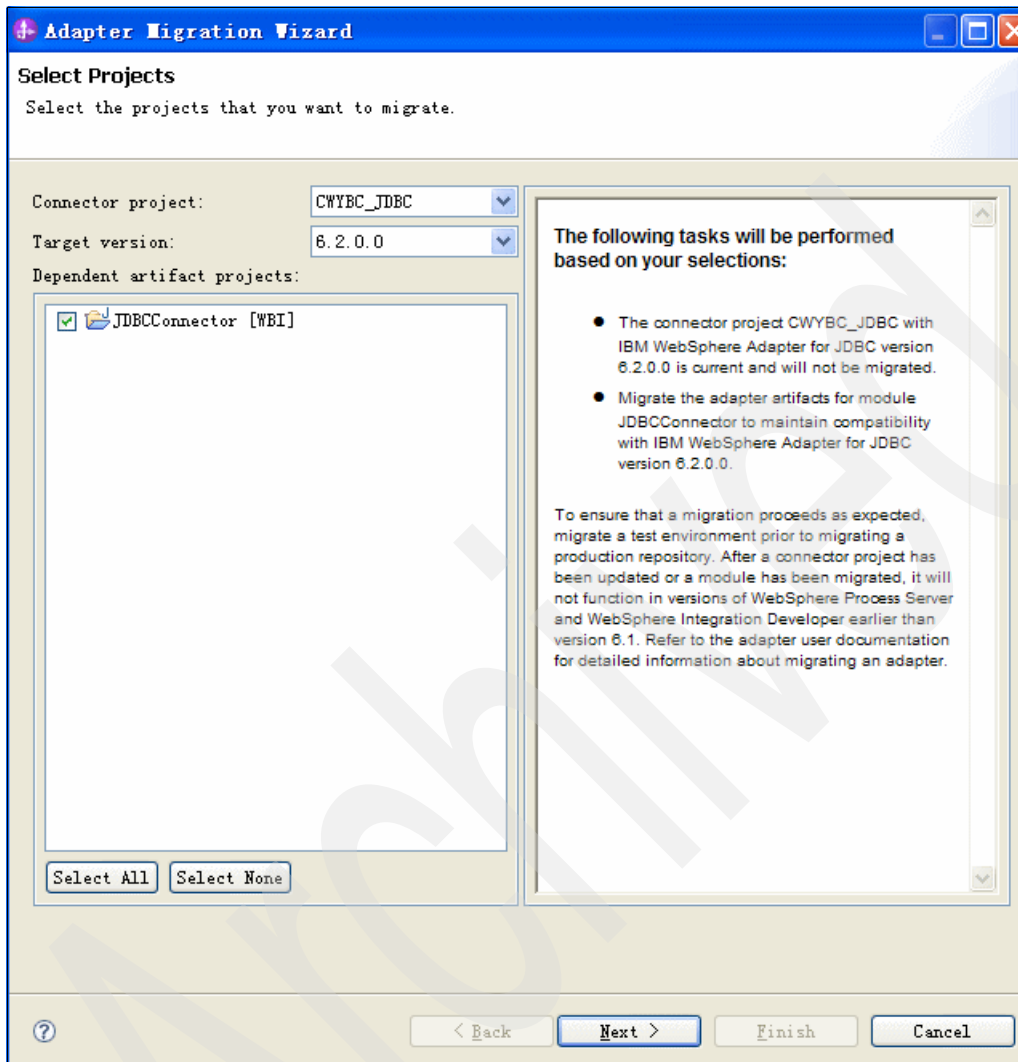


Figure 10-27 Adapter Migration Wizard

When you launch the Migration Wizard, by default all the dependent artifact projects are selected. If you clear the check mark next to a dependent artifact project, that project is not migrated.

As shown in Figure 10-28 on page 213 on the Review Changes window, you can review the migration changes that will occur in each of the artifacts that you migrate by clicking the + (plus) sign.

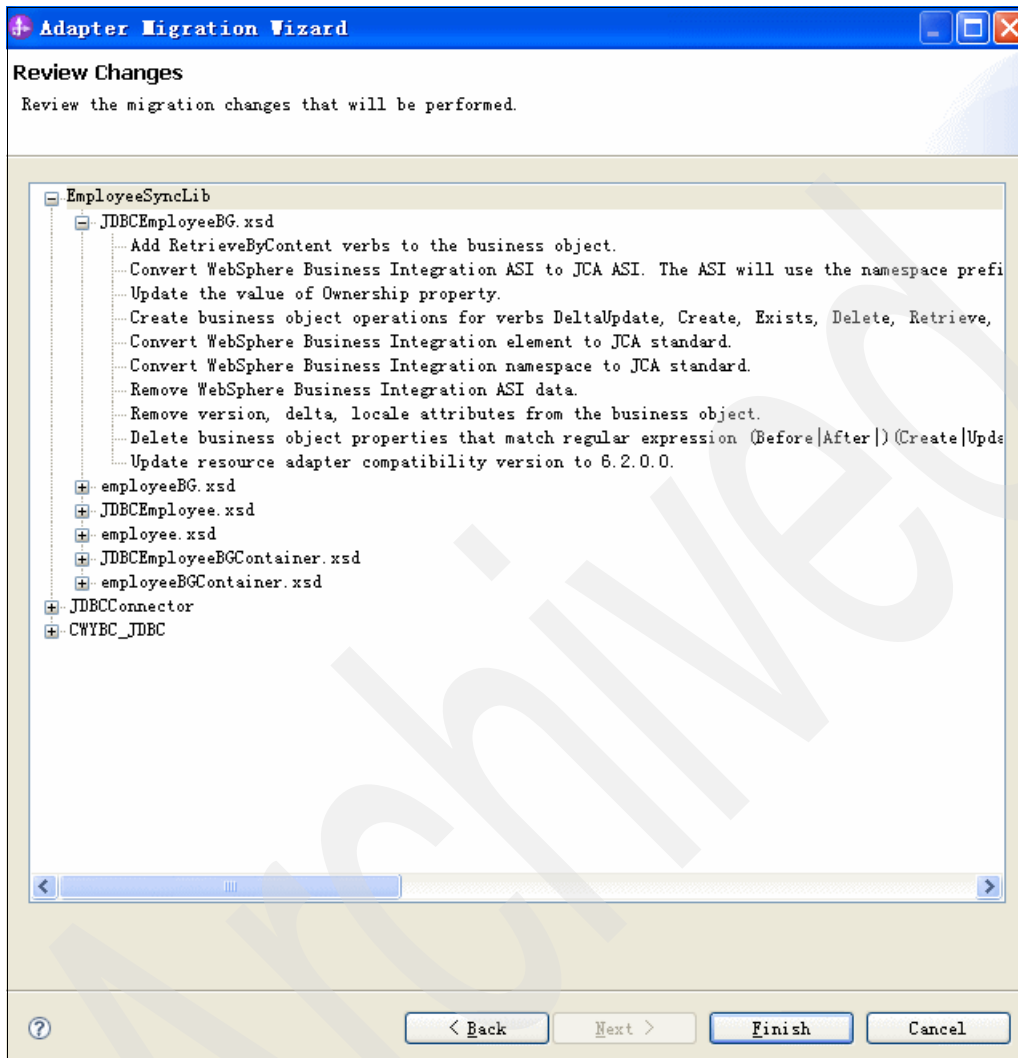


Figure 10-28 Review Changes

Click **Finish** to perform the migration.

Before performing the migration process, the wizard backs up all of the projects that are affected by the migration. The projects are backed up to a temporary folder within the workspace. If the migration fails for any reason, or if you decide to cancel the migration before it completes, the wizard deletes the modified projects and replaces them with the backup projects.

Artifacts migration

In this chapter, we provide details about the standard migration tools support for server artifacts that is provided by the Migration Wizard and **reposMigrate** tools. We discuss the major WebSphere InterChange Server artifacts and features, such as the business objects, maps, relationships, collaborations, WebSphere Business Integration Adapter definitions, and the Server Access interface. We also show how you can upgrade these artifacts and features to an equivalent component or functionality in WebSphere Process Server.

We include the following sections:

- ▶ 11.1, “Migration prerequisites and constraints” on page 216
- ▶ 11.2, “Files generated during migration” on page 216
- ▶ 11.3, “Business objects” on page 217
- ▶ 11.4, “Maps” on page 222
- ▶ 11.5, “Collaborations” on page 227
- ▶ 11.6, “Relationships” on page 237
- ▶ 11.7, “WebSphere Business Integration Adapters” on page 240
- ▶ 11.8, “WebSphere InterChange Server Access Interface” on page 248

11.1 Migration prerequisites and constraints

The following prerequisites and constraints, among others, of the migration tools are known:

- ▶ The WebSphere Process Server only supports migration from WebSphere InterChange Server Version 4.3.
- ▶ When migrating large repository files with the Migration Wizard, you might need to increase the heap size in WebSphere Integration Developer so that the migration completes without running out of memory. If you do this, a typical setting is `-Xvmx1024m`. The **reposMigrate** command-line tool consumes less memory and can be another option to circumvent this potential issue.
- ▶ The migration tools rely on best practices that are described in Chapter 13, “Best practices” on page 263.

For information about other specific limitations, see “Limitations when migrating from WebSphere InterChange Server” in the WebSphere Process Server, Version 6.2 information center at the following Web address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/cmig_limitations_wics.html

11.2 Files generated during migration

Table 11-1 on page 217 shows the artifacts that are created in WebSphere Process Server after migrating a typical WebSphere InterChange Server repository file.

Table 11-1 Generated artifacts after migrating

Generated file	File extension	File content
XML Schema Definition (XSD)	.xsd	XSD of WebSphere InterChange Server data objects
Web services Description Language (WSDL)	.wsdl	Interface, binding, and address information of Web services
Business Process Execution Language (BPEL)	.bpel and .bpelex	Business process modeled by using BPEL standards and the IBM BPEL extension definitions
Service Component Architecture (SCA) component	.component	SCA definitions
Java	.java	Java implementation for an SCA component
Map	.map	Business object mapping definitions
Mediation	.mfc	Mediation flow definitions
Interface map	.ifm	Interface mapping definitions
Relationship	.rel	XML description of a relationship, role, roleObject, and key attributes
WebSphere Business Integration Adapter	.import .export .wbia	WebSphere Business Integration adapter configuration definitions

11.3 Business objects

In the following sections, we describe the concept of business objects. We explain how they are migrated from WebSphere InterChange Server to WebSphere Process Server when using the standard migration tools.

11.3.1 Business objects in WebSphere InterChange Server

In WebSphere InterChange Server, information that is sent or received between components is packaged in the form of a business object:

- For data that is transferred between an adapter and WebSphere InterChange Server, application-specific business objects (ASBOs) are used that model the appropriate application entities.

- ▶ For data that is processed within the business logic of WebSphere InterChange Server collaboration, generic business objects (GBOs) are used. They contain a superset of information for the application-specific entities.
- ▶ Maps transform data between GBOs and ASBOs so that adapters can communicate with their applications by using application-specific entities. Collaboration objects can apply business logic in an application-independent way.
- ▶ Both ASBOs and GBOs are modeled during design time as business object definitions, which are stored in the business integration system. At run time, data is populated in a business object instance, which is based on the appropriate definition. The business object moves through the business integration system as dictated by its routing and integration or business logic rules.

11.3.2 Service Data Objects in WebSphere Process Server

The WebSphere Process Server business object framework provides a data abstraction for the SCA. It has capabilities that are similar to WebSphere InterChange Server business objects and mitigates complexities that are related to working with federated and disparate business data.

Service Data Objects (SDOs) are used as a unified pattern for data representation, transport, and persistence across application layers. WebSphere Process Server introduces business objects, which are enhanced SDOs. Business objects are part of the service-oriented architecture (SOA) core of WebSphere Process Server components as shown in Figure 11-1.

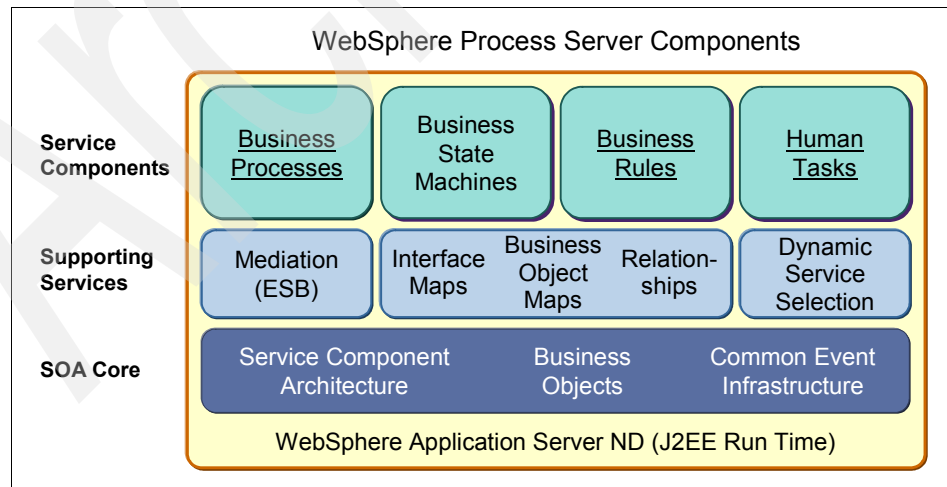


Figure 11-1 WebSphere Process Server components architecture

Business objects include extensions that are important for integration solutions and are used to further describe the data that is being exchanged between SCA services. A *business object* is a set of attributes that represents a business entity, such as an employee; an action on the data, such as create, read, update, and delete operations; and instructions for processing the data.

Business objects are flexible, because they can represent many kinds of data. For example, in addition to supporting the data synchronization model of traditional integration servers, they also can represent data that is returned from a synchronous Enterprise JavaBeans (EJB) session bean facade or a synchronous business process. Then, they can be bound to WebSphere Portal portlets and JavaServer Faces (JSF) components.

Business objects are the primary data abstraction for the SCA. Figure 11-2 illustrates how the SCA provides the framework to define service components and compose these services into an integrated application. It also shows that business objects represent the data that flows between each service. Regardless of whether the interface associated with a particular service component is defined as a Java interface or a WSDL port type, the input and output parameters are represented by using business objects.

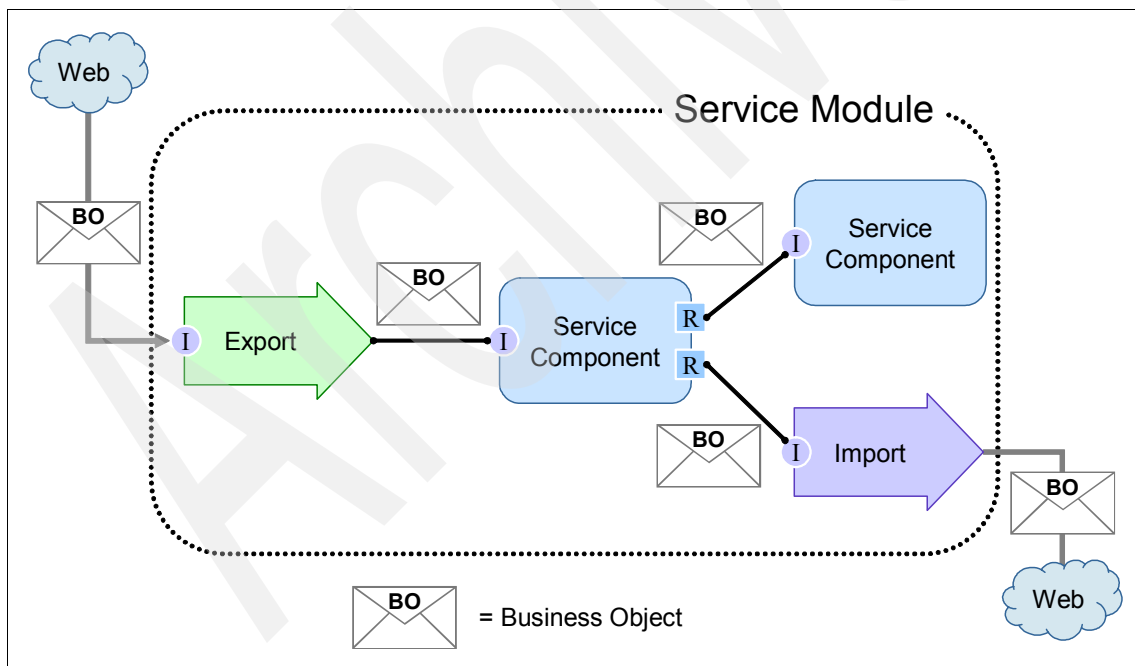


Figure 11-2 WebSphere Process Server business objects

11.3.3 Migration of business objects

The migration utility generates a WebSphere Process Server business object and a business graph for each WebSphere InterChange Server business object packaged in a repository JAR file. The business graph is a wrapper around business objects that contain change and event summaries along with verb information. The business graph is defined by the SDO JSR235 specification. It is used in WebSphere Process Server to provide increased quality of service (QoS) support.

Figure 11-3 shows business objects in a project under Data Types.

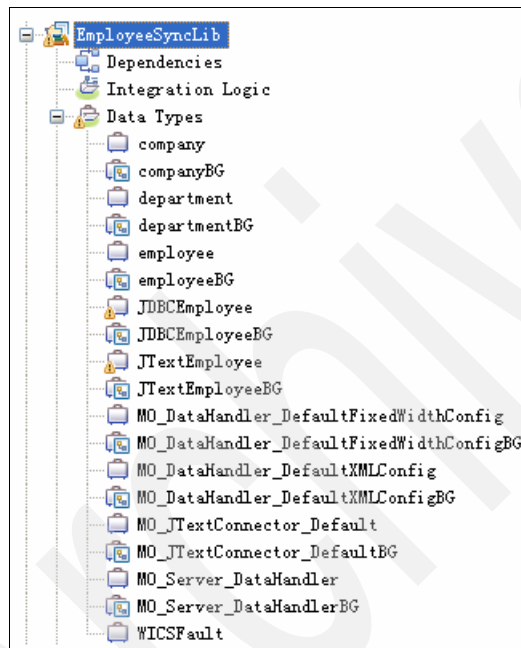


Figure 11-3 Business objects that are listed under Data Types

Figure 11-4 shows the business object of Employee.

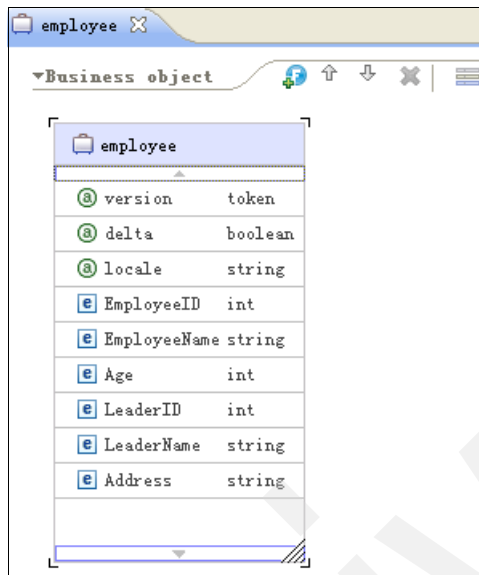


Figure 11-4 Employee business object

Figure 11-5 shows the corresponding business graph.

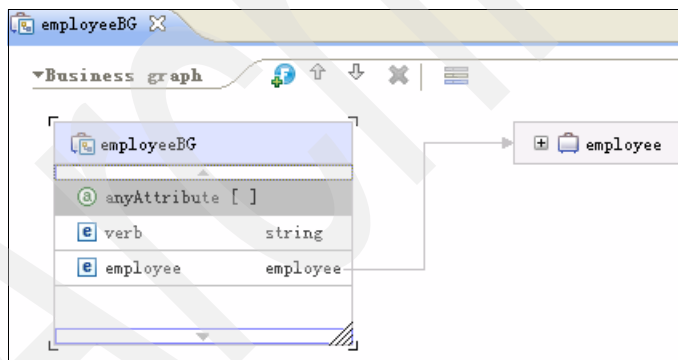


Figure 11-5 Employee business graph

Table 11-2 summarizes each of these concepts.

Table 11-2 SDO business object key concepts

Concept	Description
Business object	A business object contains attributes, each of which has a name, a type (scalar type of another business object), a default value (for scalar types), and cardinality.
Business graph	A business graph is a wrapper that is added around a simple business object or a hierarchy of business objects to provide additional capabilities, such as carrying a change summary, event summary, and verb information that is related to the business objects in the business graph.
Business object metadata	Metadata provides the ability to annotate business objects with application-specific information. This metadata is added to the business object's XML schema definition, which is also known as the <code>xs:annotation</code> and <code>xs:appinfo</code> elements.
Business object services	Business object services are the SDO API. It is the set of services for creating and manipulating business objects. Examples include services, such as create, copy, equality, and serialization.

For more information about business objects, see the following resources:

- ▶ WebSphere Process Server information center
<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>
- ▶ The SDO JSR 235 specification
<http://jcp.org/en/jsr/detail?id=235>

11.4 Maps

In the following sections, we describe the concept of *maps* and how they are migrated from WebSphere InterChange Server to WebSphere Process Server.

11.4.1 Maps in WebSphere InterChange Server

Maps in WebSphere InterChange Server are used to transform one business object to another. Maps assign a value to the attribute of a target business object based on the value or values of the attribute or attributes in the source business object. The mapping transformation rules specify how the data is mapped from a source business object to the destination business object.

Table 11-3 outlines and describes the rules in WebSphere InterChange Server for mapping.

Table 11-3 Mapping rules in WebSphere InterChange Server and their descriptions

Mapping rule	Description
Move	Copies the value of the source attribute to the destination attribute.
Set value	Specifies a value in the destination attribute.
Join	Joins two or more source attribute values into a single destination attribute. Can also specify a delimiter that is used to join the two source attributes.
Split	Splits a source attribute into two or more destination attributes based on the delimiter specified.
Submap	Calls a map for child business objects.
Cross-reference	Maintains the identity relationships for the business objects.
Custom	Writes Java code for transformations other than the standard transformation rules.

11.4.2 Maps in WebSphere Process Server

WebSphere Process Server provides two kinds of maps: interface maps and the business object maps. *Interface maps* resolve the differences between components that have different interfaces and allow them to communicate.

Business object maps in WebSphere Process Server are used to specify a value in the attribute of a target business object based on the value of the attribute in the source business object. The data mapping can be a simple copy of a source attribute value to a destination attribute. Also, complex transformation to assign a value to the attribute in the destination business object, based on a value in the attribute of a source business object, can be involved.

Table 11-4 outlines and describes the rules in WebSphere Process Server for business object mapping.

Table 11-4 Mapping rules in WebSphere Process Server and their description

Mapping rule	Description
Move	Copies the value of the source attribute to the destination attribute.
Extract	Extracts a portion of the source attribute. The source attribute must be of type String.
Join	Joins two or more source attribute values into a single destination attribute. The target of a join transform must be of type String.
Submap	The source and the target must be business objects. This transform calls a map that has the same source and the target business objects.
Assign	Sets a constant value to the target attribute.
Relationship	Performs identity relationship management. The source and the target must be business objects.
Custom	Allows the user to write logic for transformation by using Java code.
Custom assign	This rule is similar to the Custom rule, but it does not take any input.
Custom callout	This callout is similar to Custom, but it does not provide any output. It is useful for initializing before executing other transformations.

11.4.3 Migration of maps

Business object maps in WebSphere Process Server are equivalent to the maps in WebSphere InterChange Server with a slight difference because of the difference in the business objects of the two systems. Business objects in WebSphere InterChange Server represent the messages that need to flow between one system to another. They contain the verb information, which represents how the system must process the data, such as by using the create, delete, update, and retrieve operations.

In WebSphere Process Server, the verb information is stored in the business graphs, which are the wrappers added around business objects. In addition to the verb, the business graphs provide capabilities, such as the event summary and the change summary.

When a map is migrated, for every map in WebSphere InterChange Server, two maps are generated in WebSphere Process Server:

- ▶ One map for the mapping between source and the target business graph
- ▶ One map for the mapping of business objects

Figure 11-6 shows a map in WebSphere InterChange Server.

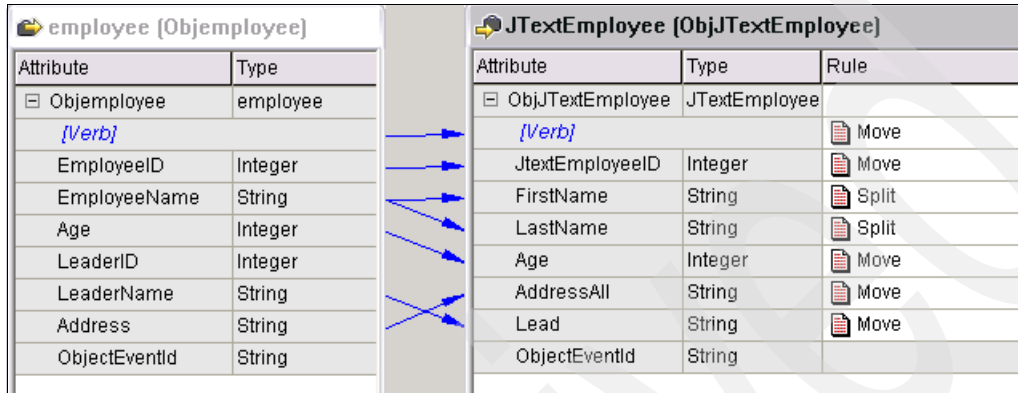


Figure 11-6 Map in WebSphere InterChange Server

Figure 11-7 on page 226 and Figure 11-8 on page 227 show the generated maps in WebSphere Process Server, when the map in Figure 11-6 is migrated from WebSphere InterChange Server.

Figure 11-7 shows the business object map in WebSphere Process Server.

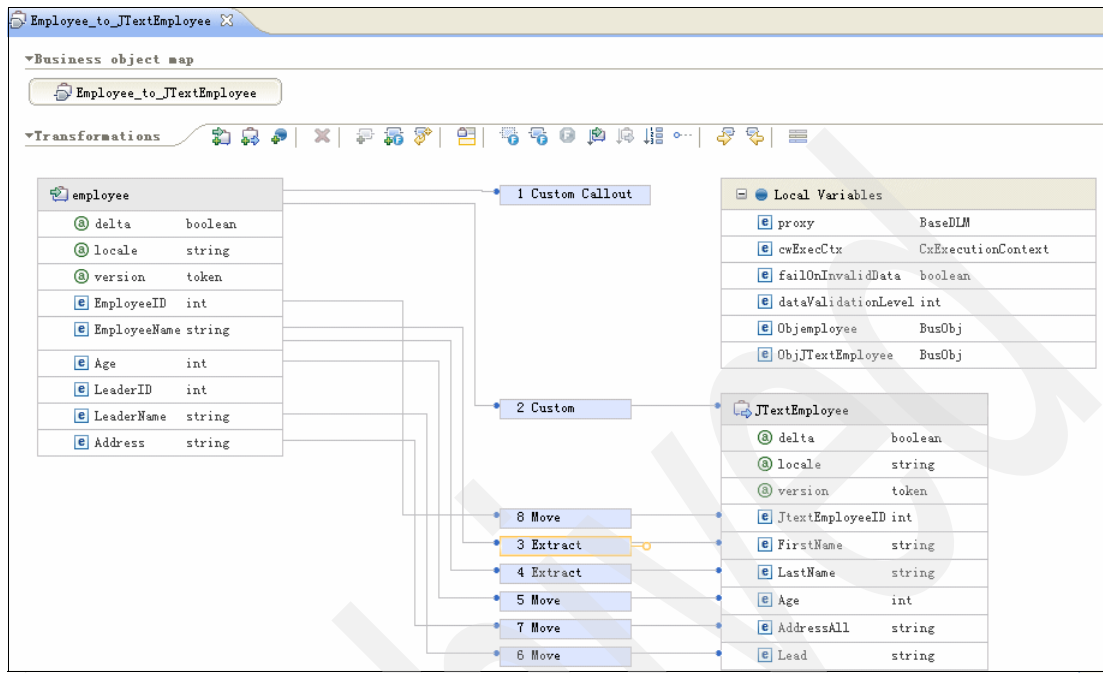


Figure 11-7 Business object map in WebSphere Process Server

Figure 11-8 on page 227 shows the business graph map in WebSphere Process Server.

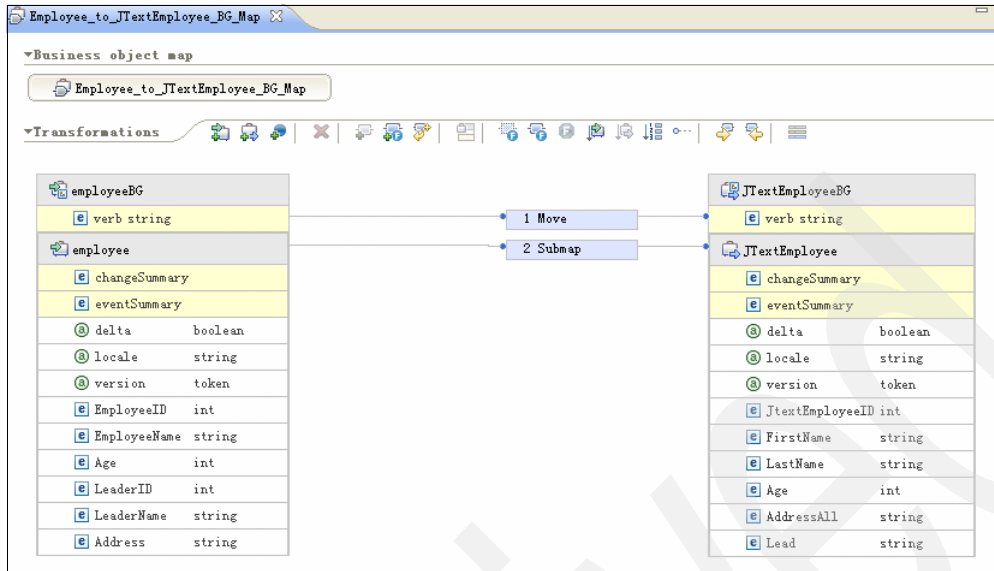


Figure 11-8 Business graph map in WebSphere Process Server

WebSphere Process Server provides support for usage of custom libraries in maps and process components. Custom transformations in WebSphere InterChange Server maps are usually coded in Java and can use custom libraries. In such cases, in the migrated WebSphere Process Server map, the custom libraries must be added explicitly under the custom transformation so that the custom APIs continue to work.

11.5 Collaborations

In this section, we describe the concept of collaborations and how are they migrated from WebSphere InterChange Server to WebSphere Process Server.

11.5.1 Collaborations in WebSphere InterChange Server

Collaborations contain the business and integration logic that is necessary to allow proper interactions between applications. Collaborations in WebSphere InterChange Server typically receive a business object from the source application and send the business objects to one or more destination applications. The verb within the business object tells the application the action, such as create or delete, that it needs to take on the data in the business object.

A *collaboration* is a two part entity: the collaboration template and the collaboration object. A *collaboration template* consists of all the collaboration's logic, but it is not executable. A *collaboration object* makes the collaboration executable by providing all the bindings to the adapters or other collaboration objects and other configuration properties.

Figure 11-9 shows a sample collaboration template in WebSphere InterChange Server.

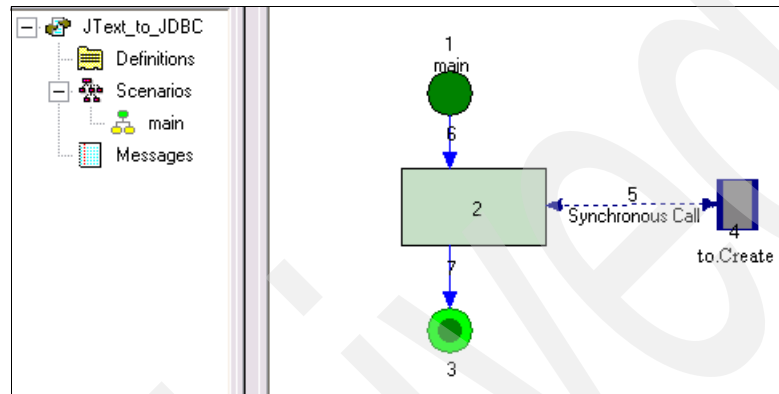


Figure 11-9 Sample collaboration template in WebSphere InterChange Server

Figure 11-10 shows a sample collaboration object in WebSphere InterChange Server.

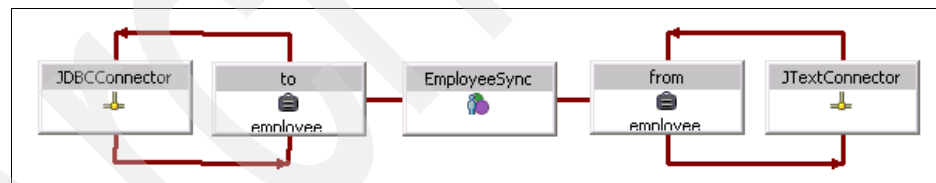


Figure 11-10 Sample collaboration object in WebSphere InterChange Server

11.5.2 Business processes in WebSphere Process Server

Business processes in WebSphere Process Server provide the means through which applications are integrated. Business processes are written based on the Web services BPEL industry standard in WebSphere Process Server. A business process is an SCA component that enables it to interact with other SCA components.

Business processes in WebSphere Process Server can be interruptible and non-interruptible processes. An *interruptible process* is a long running business

process and waits at specific activities in a process until an appropriate action is taken. An activity can wait until it receives a response from the external system. Such a process is an *uninterruptible process*. Business processes in WebSphere Process Server support compensations that allow for actions to occur when an undesirable state is reached during the normal execution of the process.

WebSphere Process Server also has human task activities that you can use in a business process to allow for manual action by a human being. Figure 11-11 on page 230 shows a sample business process in WebSphere Process Server.

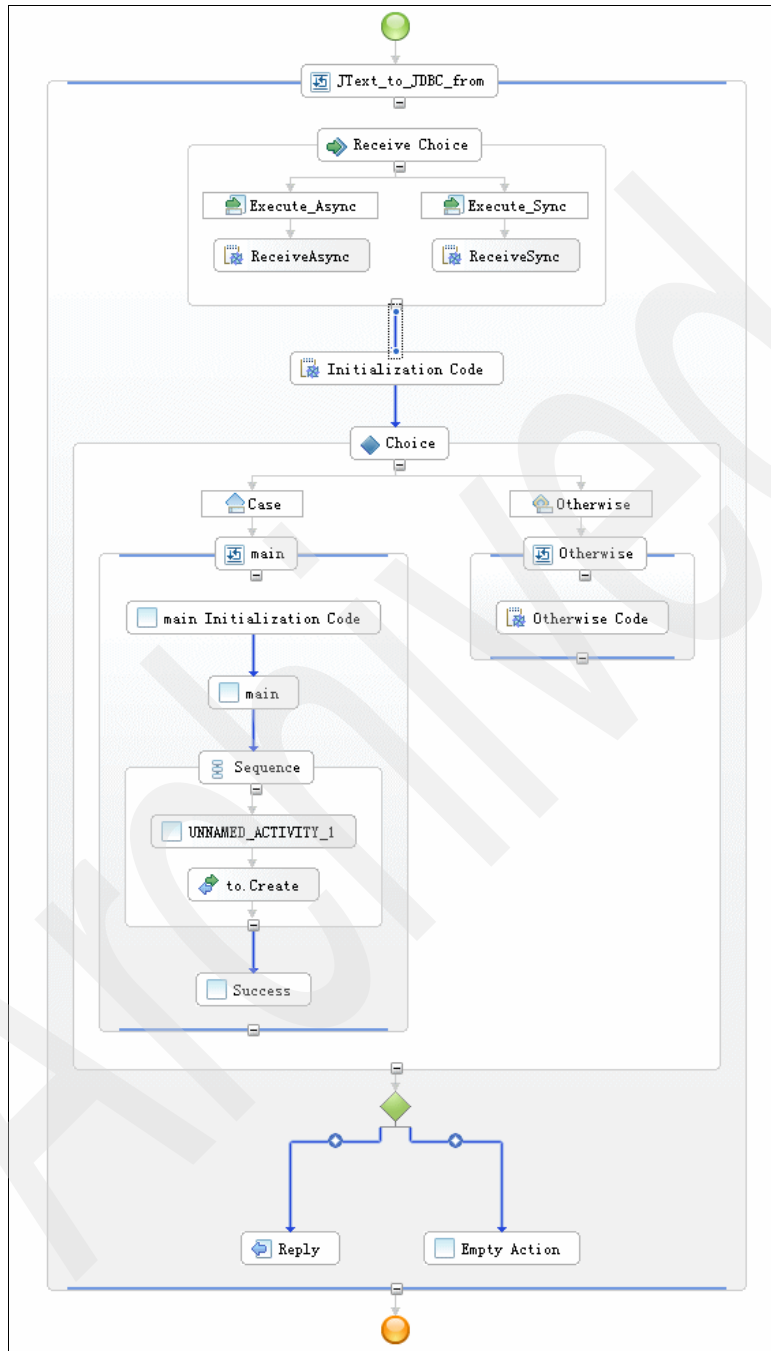


Figure 11-11 Sample business process in WebSphere Process Server

11.5.3 Migration of collaborations

The migration tool in WebSphere Process Server generates a business process and interface maps in WebSphere Process Server for collaboration in WebSphere InterChange Server. For each port in the collaboration template, the migration tool generates an SCA binding in WebSphere Process Server. Decision nodes in the collaborations are converted to links with conditions in BPEL. Exception paths in collaborations are migrated as fault handlers in WebSphere Process Server. To handle the asynchronous and synchronous execution of a collaboration, the migrated BPEL creates a “receive choice” activity.

Collaborations with decision nodes

Decision nodes are used in collaboration templates to control the flow of the process. A decision node has a normal path, a default path, and an exception path. The normal path allows for a condition to be evaluated and executes if the condition is *true*. An exception path is executed when an exception occurs. The default path is executed when none of the other conditions evaluate to *true*.

Figure 11-12 shows a collaboration with a decision node.

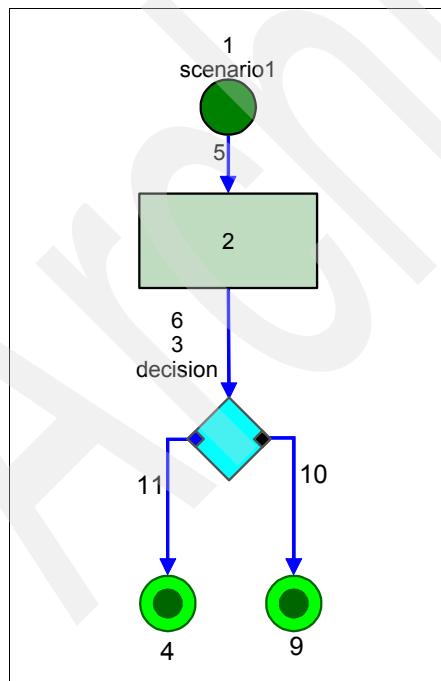


Figure 11-12 WebSphere InterChange Server collaboration with a decision node

Figure 11-13 shows the decision node criteria that is used in the collaboration in Figure 11-12 on page 231.





	Type	Condition	Branch Label	Comment
1 	Normal	5 < 25		
2 	Default			
3 				
4 				

Figure 11-13 Decision node criteria

WebSphere Process Server does not have nodes specifically for decisions. It has links that have conditions associated to them. As shown in Figure 11-14 on page 233, two success nodes have links with small diamonds on the link. The diamonds on the link indicate that there is a condition associated with the link. WebSphere InterChange Server decision node conditions are migrated to the link conditions in WebSphere Process Server except for the exception paths.

Exception paths are migrated as fault handlers to WebSphere Process Server. A WICSFault business object is created when a collaboration is migrated to WebSphere Process Server. Fault handling is handled by surrounding a block of nodes that can throw an exception within a sequence. BPEL checks the type of WICSFault to determine the variables to set in the WICSFault business object.

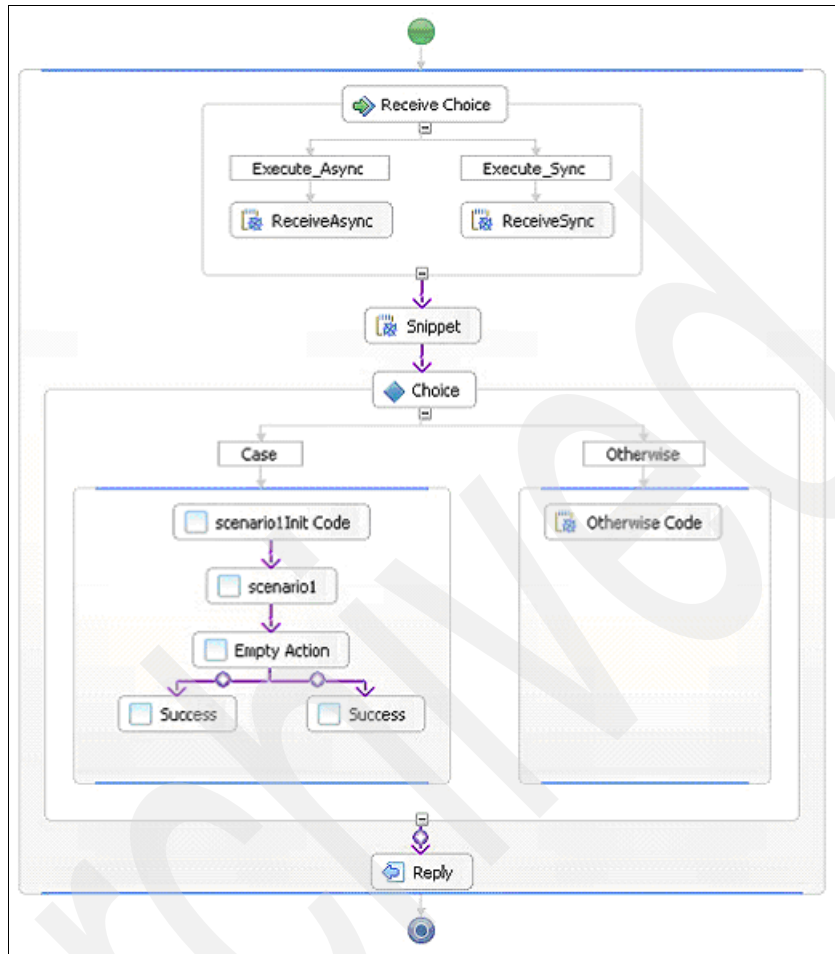


Figure 11-14 Business process with a condition on two of the links in WebSphere Process Server

Collaborations with loops

WebSphere InterChange Server provides an *iterator node* for looping. A For loop can be specified in the iterator node by specifying a start value, a condition, and an increment to loop through a block of activities. The loop conditions are used to create the conditions for the while loop in business processes in WebSphere Process Server when migrated.

Collaborations with break nodes

WebSphere InterChange Server has *break nodes* to break out of the loops when the break nodes are encountered and continue with the next node after the loop.

Break nodes are only allowed in the iterator nodes within a collaboration. Figure 11-15 shows a simple iterator node within a collaboration with a break node.

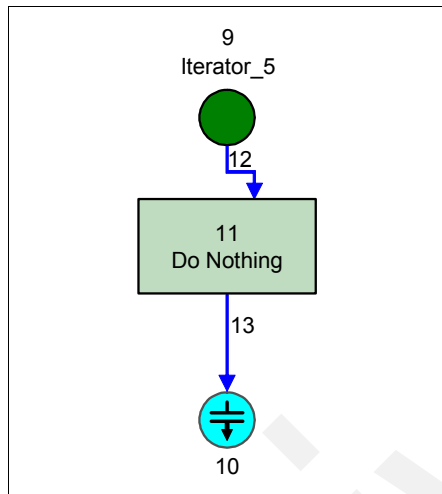


Figure 11-15 Collaboration with a break node

WebSphere Process Server does not have a break node equivalent. Therefore, during migration, the break node in WebSphere InterChange Server is converted to a Java snippet that declares a Boolean variable to break out of the loop when it is true. The while loop in WebSphere Process Server iterated with the logic that it did in WebSphere InterChange Server, with additional logic in which the Boolean variable is not true.

Collaborations with asynchronous inbound service calls

Asynchronous inbound service calls in WebSphere InterChange Server are converted to receive choice activities in WebSphere Process Server, because asynchronous inbound service calls have a timeout set. If no timeout is set, receive activities can be used in WebSphere Process Server.

Figure 11-16 shows an asynchronous inbound service call in a collaboration template.

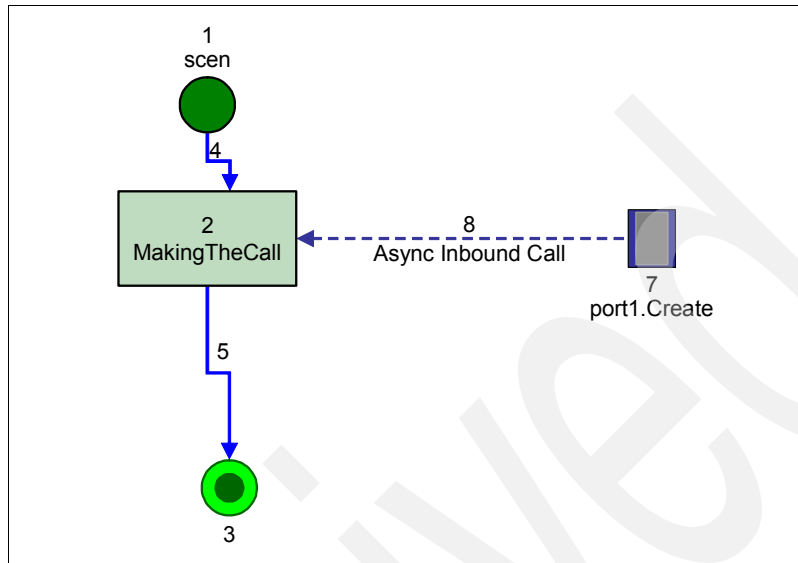


Figure 11-16 Collaboration with asynchronous inbound service call

Figure 11-17 shows the receive choice activity in WebSphere Process Server when a collaboration with an asynchronous inbound service call is migrated.

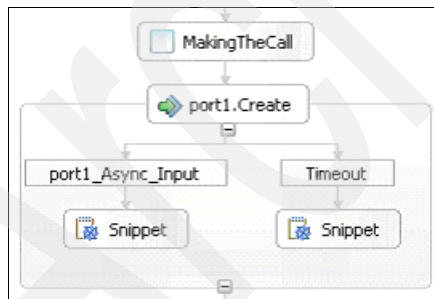


Figure 11-17 Receive choice activity in WebSphere Process Server

Collaborations with multiple triggering ports

Collaborations in WebSphere InterChange Server have multiple triggering ports. When migrated to WebSphere Process Server, a BPEL is created for each triggering port. All the scenario code is repeated in all the BPELs. The only difference is the triggering logic. The migrated BPELs have a case logic to determine which scenario is triggered.

Collaborations with custom properties

Custom properties can be defined in a collaboration template, and a value can be assigned in the collaboration object. When migrated to WebSphere Process Server, this information is captured as an environment variable for the resulting BPEL.

Long-lived business processes

Collaborations can be long-lived business processes (LLBPs) in WebSphere InterChange Server. The collaborations are migrated as long running business processes in WebSphere Process Server. The instance data for a long-lived business process is not migrated for in-flight processes. Therefore, these processes must run in WebSphere InterChange Server until the process is complete.

Custom code in collaborations

WebSphere Process Server provides backward compatibility to support WebSphere InterChange Server APIs. Any custom code in WebSphere InterChange Server collaborations is migrated to WebSphere Process Server. However, if custom libraries other than the standard Java Developer Kit (JDK™) libraries are used in collaborations, the libraries must be migrated to WebSphere Process Server for the migrated collaborations to work. Use only the documented WebSphere InterChange Server API calls to ensure a smooth migration.

Additional information

See the following developerWorks articles for more information about migration of collaborations:

- ▶ “Migrating WebSphere InterChange Server artifacts to WebSphere Process Server artifacts, Part 1: Migrating collaboration templates to BPEL”
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0612_seacat/0612_seacat.html
- ▶ “Migrating WebSphere InterChange Server artifacts to WebSphere Process Server artifacts, Part 2: Understanding the WebSphere Process Server SCA components”
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0703_seacat/0703_seacat.html

11.6 Relationships

In the following sections, we describe the concept of relationships and how are they are migrated from WebSphere InterChange Server to WebSphere Process Server.

11.6.1 Relationships in WebSphere InterChange Server

When attributes in a source and destination business object contain equivalent data that is represented differently, the transformation step employs a relationship. A *relationship* establishes an association between data from two or more business objects. Each business object is called a *participant* in the relationship.

WebSphere InterChange Server relationships are classified into the following categories:

- ▶ A *lookup relationship* establishes an association between data, such as attributes in business objects. The data can be related on a one-to-one, one-to-many, or many-to-many basis. Lookup relationships typically transform non-key attributes whose values are represented with codes.
- ▶ An *identity relationship* establishes an association between business objects or other data on a one-to-one basis. For each relationship instance, each participant can have only one instance. Identity relationships typically transform the key attributes of business objects, such as ID numbers and product codes.
- ▶ A *nonidentity relationship* establishes an association between business objects or other data on a one-to-many or many-to-many basis. For each relationship instance, there can be one or more instances of each participant.

11.6.2 Relationships in WebSphere Process Server

Relationships are supporting services in WebSphere Process Server applications that establish an association between data from two or more data types. Each data type is represented as a role in the relationship. Each role in a relationship is distinguished from other roles based on the function that they serve in the given relationship. Several roles ultimately contribute to build the relationship.

Table 11-5 on page 238 outlines the key concepts of WebSphere Process Server relationships.

Table 11-5 WebSphere Process Server relationship key concepts

Concept	Description
Relationship definition	Describes how a relationship needs to look. It also identifies each role and specifies how the roles are related. The base for a role is a data type that can be either a business object, an XSD simple type, or types used in interfaces.
Role definitions	Have exactly one RoleObject that is defined that represents the data type. Each role might have one to several KeyAttributes defined, which represent important attributes of the data type.
Relationship role	Describes how entities can participate in a given relationship. That is, these role definitions are used to capture structure and constraint requirements on participating entities and their manner of participation.
Relationship instance data	Used for a static mapping of attributes of a data type. Static mappings are used for data that does not change further, which usually applies to data, such as postal codes, country codes, or area codes.

11.6.3 Migration of relationship artifacts

The migration utility generates a WebSphere Process Server relationship definition for each WebSphere InterChange Server relationship that is defined in the repository JAR file. Figure 11-18 shows a sample relationship, called `contactR`, in WebSphere Integration Developer.

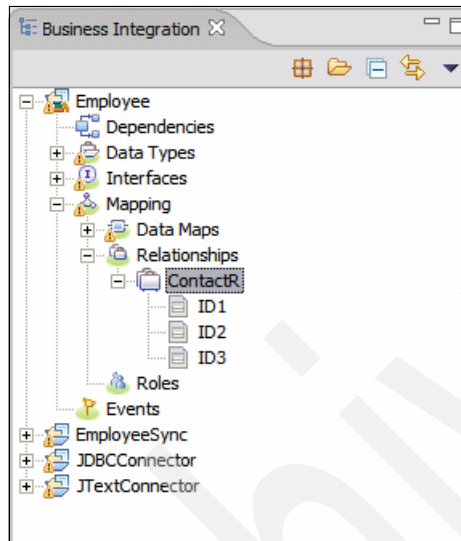


Figure 11-18 Example of a relationship

Figure 11-19 shows the graphical view of the contact relationship.

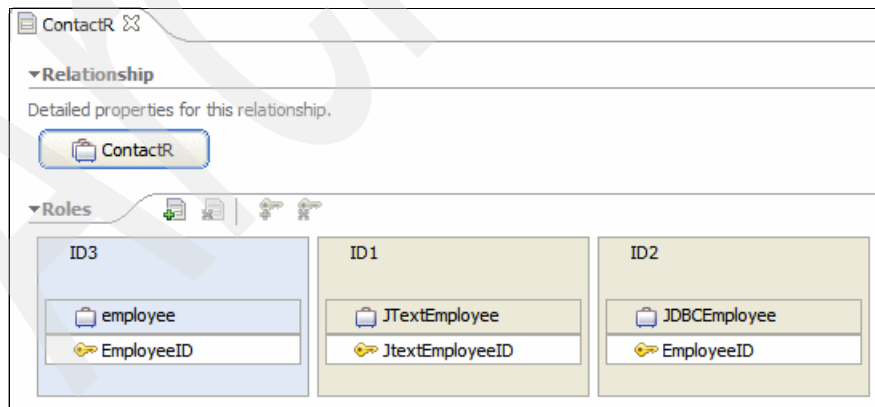


Figure 11-19 ContactR relationship

Coexistence of relationships

Relationship table schema and instance data can be reused by WebSphere Process Server and shared concurrently between WebSphere InterChange Server and WebSphere Process Server.

In WebSphere Process Server, a Jython script is generated during migration. It has references to relationships and DBConnection pools of WebSphere InterChange Server. You must run this script by using the **wsadmin** command to create WebSphere Process Server configuration definitions that correspond to the original WebSphere InterChange Server artifacts.

11.7 WebSphere Business Integration Adapters

In this section, we discuss the support provided by the migration tools that are dedicated to server artifacts only.

Another tool called the *Adapter Migration Wizard* helps to upgrade WebSphere Business Integration Adapter into WebSphere Adapters. Because we do not discuss this tool in this section, instead see 10.5, “Migration tools specific to WebSphere Business Integration Adapters” on page 208 and 10.5.2, “Migrating using the Adapter Migration Wizard” on page 211.

11.7.1 WebSphere Business Integration Adapters with WebSphere InterChange Server

The WebSphere Business Integration Adapter architecture in Figure 11-20 on page 241 outlines the components of the solution. It also illustrates the following steps for a solution with WebSphere Business Integration Adapter in WebSphere InterChange Server:

1. In this case, install the adapter framework and adapter at target application A.
2. Create business objects for the transactions of interest by using the Business Object Designer and Object Discovery Agent (ODA) wizards.
3. Configure the adapter and transport to communicate with the integration server, as well as the target Application A.
4. If using event notification, install an event store in the application and triggers to detect events. Insert event records into the event store.

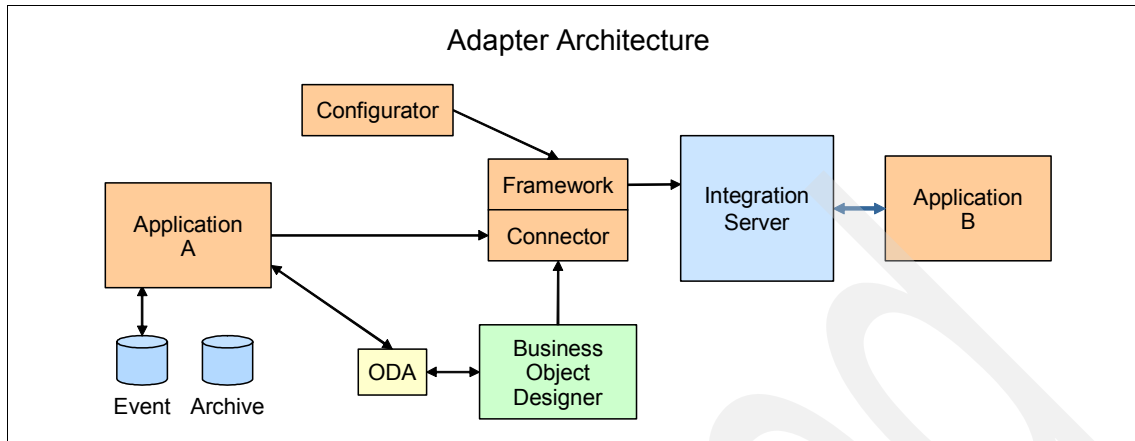


Figure 11-20 WebSphere Business Integration Adapter architecture

Figure 11-21 on page 242 illustrates the following WebSphere Business Integration Adapter deployment environment:

1. The adapter runs as a separate Java virtual machine (JVM) process.
2. The connector controller manages the exchange of business objects between the server and adapter.
3. The adapter framework requests the application-specific component to check for changes to the application's event store.
4. The application-specific component polls for changes to the application's event store.
5. The application-specific component determines whether the changed data maps to a supported business object definition.
6. The application-specific component instantiates a business object and uses it to retrieve the changed data.
7. The application-specific component initiates an event delivery to transfer the business object to the adapter framework.
8. By using the adapter data handlers, serial application data is transformed into business objects. Maps are used between a business object that is structured for the data model of a specific application and a business object that is generically structured for use by collaborations at the hub.

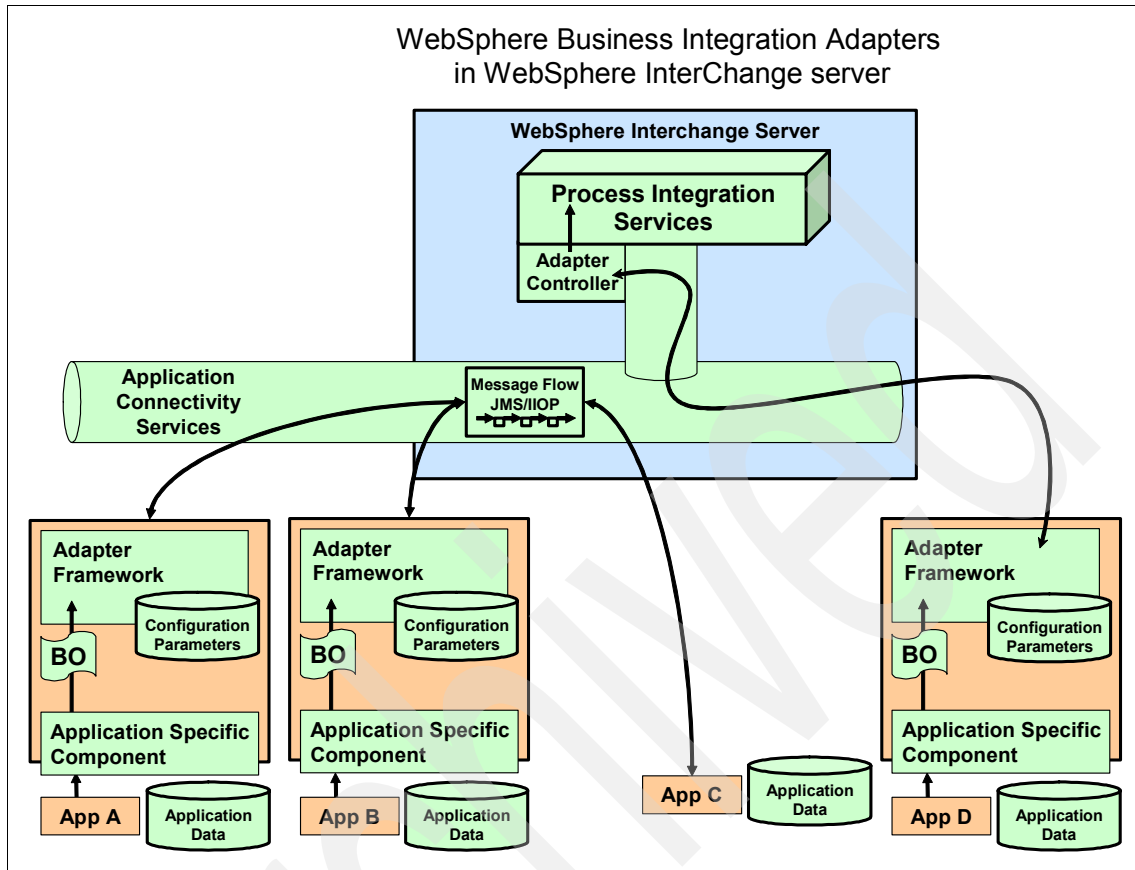


Figure 11-21 WebSphere Business Integration Adapter deployment in WebSphere InterChange Server

11.7.2 WebSphere Business Integration Adapters in WebSphere Process Server

WebSphere Process Server has built-in functionality to provide support and work with the existing portfolio of WebSphere Business Integration Adapters.

Figure 11-22 on page 243 shows an integration scenario in WebSphere Process Server, where WebSphere Business Integration Adapters are represented in the form of SCA import and export components. In this scenario, they are wired to a BPEL process component.

The WebSphere Business Integration Adapter runs as a separate process, outside of the WebSphere Process Server, and communicates with the server by using the Java Message Service (JMS) protocol.

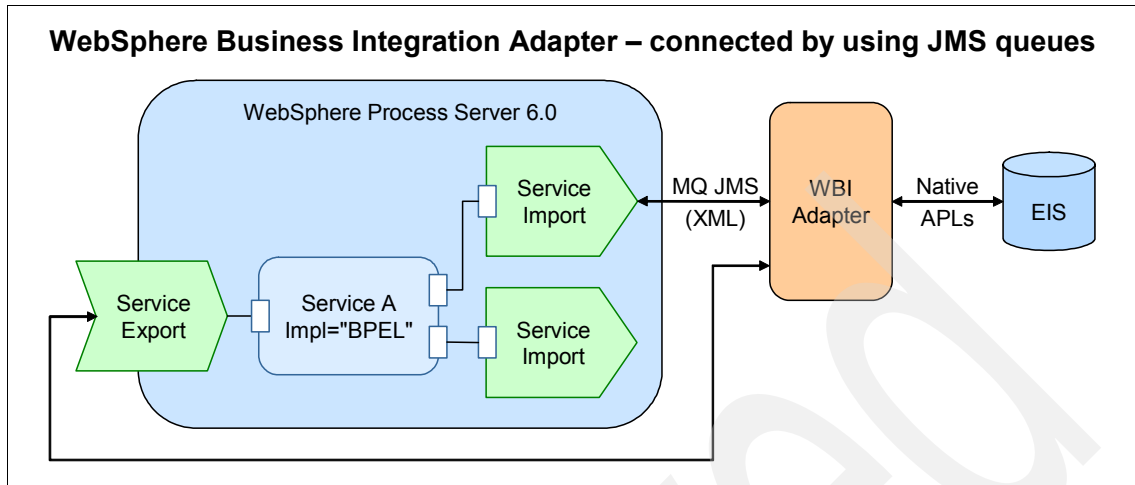


Figure 11-22 Using WebSphere Business Integration Adapter in WebSphere Process Server

Using the WebSphere Business Integration Adapter in WebSphere Process Server

To configure the WebSphere Business Integration Adapter to work with WebSphere Process Server:

- ▶ If you are using the Migration Wizard (preferred method generally), import the repos_copy JAR file through the wizard, which then generates a module per adapter definition.
- ▶ If you are using the WebSphere Business Integration Adapter Artifact Importer:
 - a. Generate the ASBO XSD file by using the ODA tool that is provided by the WebSphere Business Integration toolset.
 - b. Create a connector configuration file to be used by WebSphere Process Server by using the Connector Configurator tool in the WebSphere Business Integration toolset.
 - c. Create the adapter module in WebSphere Integration Developer by using the Enterprise Service Discovery wizard.

The wizard prompts for the location of the business objects (XSD files) that are created using ODA and the connector configuration file that is created in the previous step. The wizard builds the adapter module by using the business objects and connector configuration file that are provided.

11.7.3 Migration of WebSphere Business Integration Adapter artifacts

The Migration Wizard generates the WebSphere Process Server Adapter artifacts as separate projects for each connector definition in the WebSphere InterChange Server repository JAR file and a common library that is shared by other projects as shown in Figure 11-23. Each migrated adapter project contains business objects and adapter definition artifacts, such as a .wbia file.

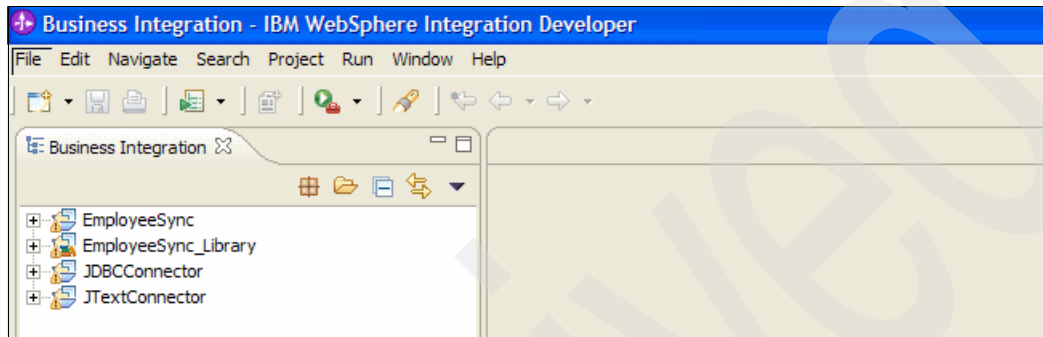


Figure 11-23 Generated project after migration

Part of the manual reconfiguration presented in the following section is required for WebSphere Business Integration Adapter to work with WebSphere Process Server.

Modifying the business objects

The business objects that are part of the migrated JAR file require minor modifications, such as renaming the XML files in the BusinessObject folder, to have a file extension of XSD. These changes are made so that the adapters that are configured with WebSphere Process Server as the broker can use these business objects successfully.

Modifying the adapter configuration by the using the configurator tool

To modify the adapter configuration:

1. Set the BrokerType parameter value to **WebSphere Application Server**.

An MQ client link is created on the WebSphere Process Server Service Integration Bus. This link matches the `jms.MessageBrokerName` that is defined in the WebSphere Business Integration Adapter connector configuration for the file.

2. Change the `jms.MessageBrokerName` field:

```
WBIA_QM:WBIA.JMS.SVRCONN:host name:SIB_MQ_ENDPOINT_ADDRESS
```

To view the value of the `SIB_MQ_ENDPOINT_ADDRESS` port for your installation, in the administrative console, select **Application servers** → **server1** → **Ports**.

Example

Refer to Appendix B, “Access EJB migration” on page 591, for a detailed example that shows how to reuse the WebSphere Business Integration Adapters in a WebSphere Process Server environment.

11.7.4 WebSphere Adapters

In this section, we provide background information about the WebSphere Adapters, also called *J2EE Connector Architecture (JCA) adapters*. The current migration tools provided with WebSphere Process Server V6.2 support migration from WebSphere Business Integration Adapters to WebSphere Adapters by providing a two-step migration process. The second step is handled by a migration tool called *Adapter Migration Wizard*. For more information about this tool, see 10.5, “Migration tools specific to WebSphere Business Integration Adapters” on page 208 and 10.5.2, “Migrating using the Adapter Migration Wizard” on page 211.

WebSphere Adapter portfolio is a new generation of adapters based on the JCA. Enterprise information systems (EISs) provide native APIs for identifying a function to call, specifying its input data, and processing its output data. The goal of the JCA is to provide an independent API for coding these functions, facilitate data sharing, and integrate J2EE applications with existing and other EISs. The JCA standard accomplishes this goal by defining a series of contracts that govern interactions between an EIS and J2EE components within an application server. Figure 11-24 on page 246 shows the detail of the JCA adapter architecture.

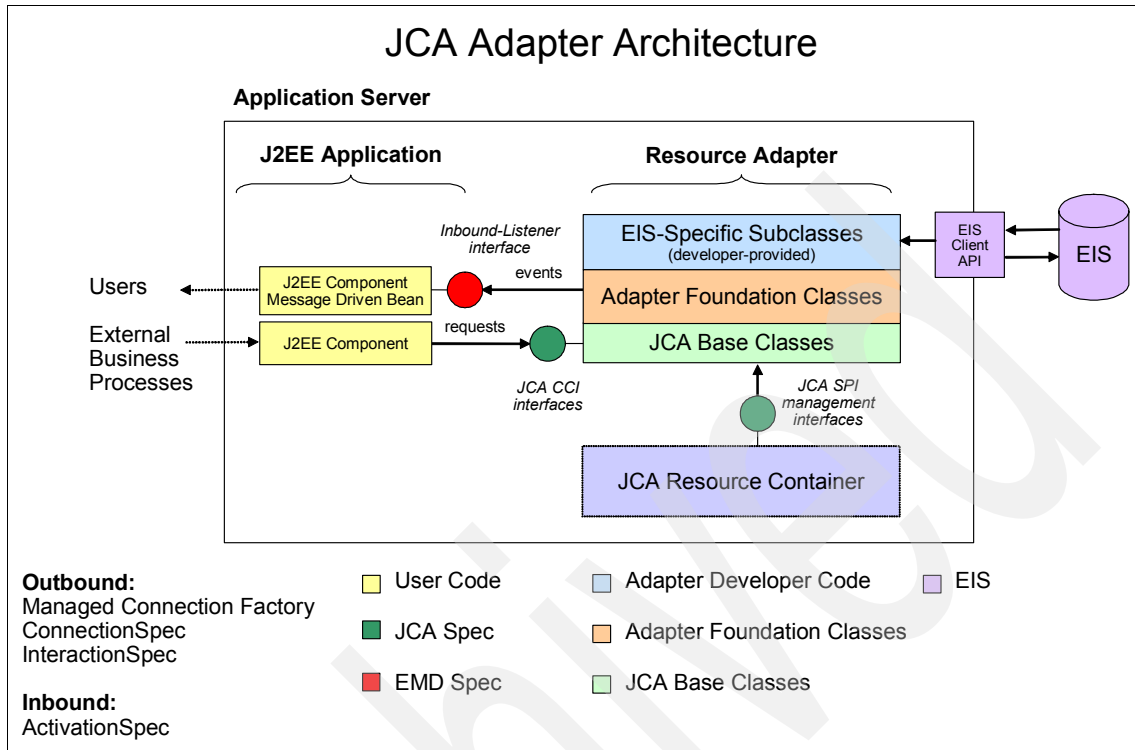


Figure 11-24 JCA adapter architecture

WebSphere Adapter components

The WebSphere Adapter includes the following components:

- ▶ **Foundation classes:** A set of class extensions that is built on top of the JCA specifications that implement functions that are common to all adapters
- ▶ **EIS subclasses:** Subclasses that implement the Common Client Interface (CCI) and EIS API contracts
- ▶ **Enterprise Metadata Discovery (EMD):** Utility that introspects the EIS to discover business objects and services and generates the SCA artifacts for the adapters

Figure 11-25 on page 247 illustrates an inbound request that is initiated by the back-end EIS based on several create, read, update, and delete events on the business objects. The adapter gets the event either by push or poll, based on how the specific adapter interact with the EIS. Then, the adapter fetches the business objects from the EIS before sending it to the listener that handles the event. An outbound client can be an SCA component in the same module or a different module or an external client, such as Java Server Pages (JSP).

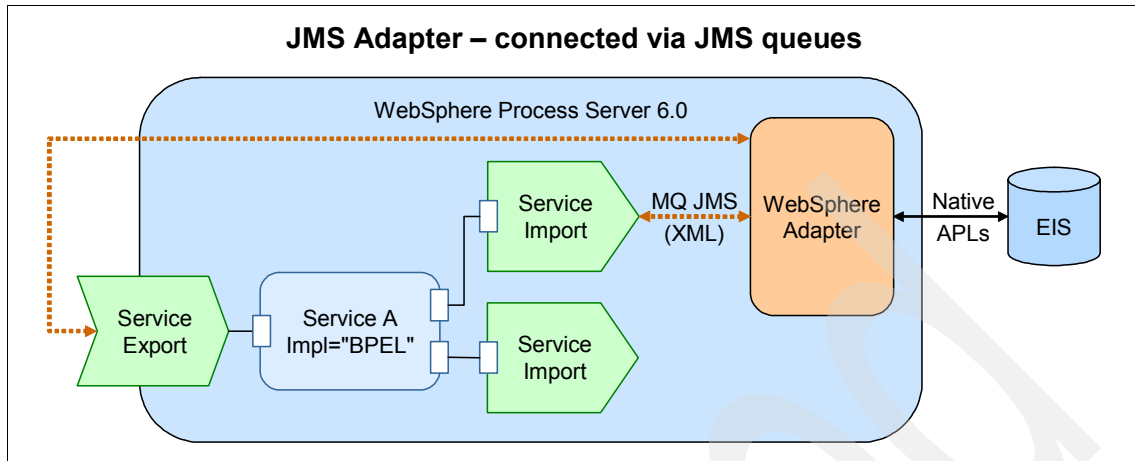


Figure 11-25 WebSphere JCA Adapter process

WebSphere JCA adapters

WebSphere JCA adapters include the following application adapters:

- ▶ JD Edwards EnterpriseOne
- ▶ Oracle E-Business Suite
- ▶ PeopleSoft Enterprise
- ▶ SAP Software
- ▶ Siebel Business Applications

WebSphere JCA adapters include the following technology adapters:

- ▶ E-mail
- ▶ Flat files
- ▶ FTP
- ▶ JDBC

Differences between WebSphere Business Integration Adapters and WebSphere Adapters

The WebSphere Business Integration Adapters and WebSphere Adapters have the following differences:

- ▶ WebSphere Business Integration adapters are proprietary software written based on the WebSphere InterChange Server architecture. WebSphere Adapters are based on the JCA version.
- ▶ WebSphere Business Integration Adapters are broker independent, but WebSphere Adapters currently support only WebSphere Process Server and WebSphere Application Server as a broker.

- ▶ WebSphere Adapters are all based on Java, but several of the WebSphere Business Integration Adapters are written in C++.
- ▶ WebSphere Business Integration Adapters run as separate JVM processes outside of WebSphere Process Server. WebSphere Adapters run inside the J2EE Connector architecture (J2C) container in WebSphere Application Server.

11.7.5 WebSphere Process Server bindings

The current migration tools that are provided with WebSphere Process Server V6.2 support direct migration from WebSphere Business Integration Adapters to WebSphere Process Server bindings with WebSphere InterChange Server data handlers or WebSphere Process Server data handlers.

For a general presentation of WebSphere Process Server bindings, see 5.6, “WebSphere Process Server bindings” on page 101.

For a general presentation of WebSphere InterChange Server data handlers, see 5.3, “WebSphere Business Integration data handlers” on page 87.

See Chapter 15, “Data access scenario with technology adapters” on page 293 for detailed examples that show how to migrate WebSphere Business Integration Adapters to WebSphere Process Server bindings.

For information about support from the WebSphere Business Integration Adapters to WebSphere Process Server bindings, see “WebSphere InterChange Server migration scenarios” in the WebSphere Integration Developer, Version 6.2 information center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.migration.ui.doc/topics/cmigwicsscen.html>

11.8 WebSphere InterChange Server Access Interface

In this section, we discuss the Server Access Interface in WebSphere InterChange Server and Access EJB support in WebSphere Process Server.

11.8.1 Overview of the Server Access feature in WebSphere InterChange Server

In support of the EJB specification 1.1, IBM provides WebSphere InterChange Server Access for EJBs. With this feature, a session bean called the *WebSphere Access EJB* or *Access EJB* is created and enables J2EE client components to

access the WebSphere InterChange Server. Access EJB enables a client component to access and use collaborations and other services. Collaborations represent business processes, which can involve multiple EISs.

To communicate with a WebSphere InterChange Server-managed EIS, a J2EE client component sends a request to a WebSphere Access EJB. In turn, the Access EJB sends data that represents the triggering event of a collaboration to WebSphere InterChange Server. WebSphere InterChange Server handles any communication with WebSphere InterChange Server-managed EISs by controlling execution of the collaboration. To provide this communication, WebSphere Server Access for EJB includes the following components:

- ▶ Implementation of the WebSphere Access EJB

To provide communication between the client component and the Access EJB, the client component calls either the home or remote interface of the Access EJB. The client component, which is in the client tier, interacts with the Access EJB to request execution of a business process, in the form of collaboration, in a WebSphere InterChange Server-managed EIS.

- ▶ Client-side version of WebSphere InterChange Server Access

To provide communication between the Access EJB and WebSphere InterChange Server, each of these components uses WebSphere InterChange Server Access Interface. The Access EJB, which is located in the middle tier, communicates with the WebSphere InterChange Server through the Server Access Interface to send the request for collaboration execution to WebSphere InterChange Server, which is in the server tier.

Figure 11-26 on page 250 illustrates how to access a WebSphere InterChange Server-managed EIS through the WebSphere Access EJB.

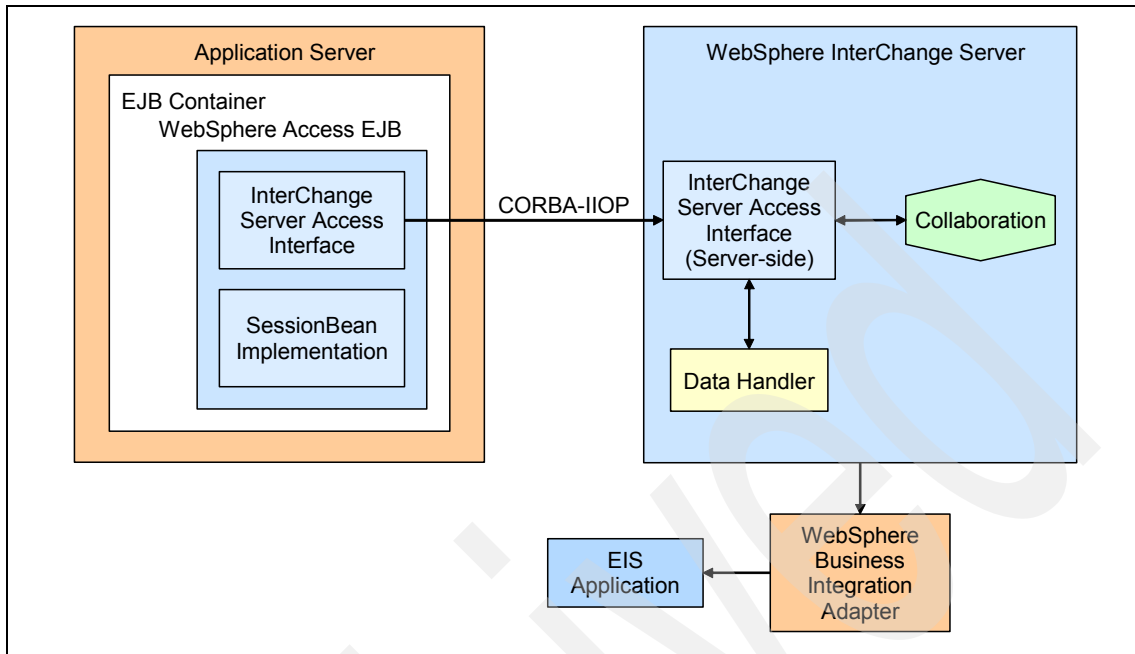


Figure 11-26 Access EJB communication with WebSphere InterChange Server

11.8.2 Access EJB support in WebSphere Process Server

For backward compatibility, WebSphere Process Server provides deprecated support for Access EJB. The Access EJB support assumes that the SCA BPEL modules to be invoked are already generated by the WebSphere InterChange Server migration tools. The mapping from the collaboration name and port name, that is, the input parameters for the Access EJB, to the SCA module name, interfaces, and business object types assume the conventions that are used by the migration tools.

The Access EJB support in WebSphere Process Server is delivered in the `AccessEJB.zip` project interchange file and is located in the `WPS_Install_root/HeritageAPI` directory. Access EJB support consists of an EJB (Access EJB) that references an SCA module project in charge of DynamicRouting that invokes the SCA BPEL module. This SCA BPEL module is the migrated version of the collaboration that was invoked in WebSphere InterChange Server. The DynamicRouting module uses a selector component to choose the correct SCA target based on the collaboration name and port name passed to the Access EJB.

The Access EJB is replicated in WebSphere Process Server as an EJB. This way, all external J2EE clients can maintain the same EJB calls, invoking the target BPEL in the migrated module instead of the target collaboration in WebSphere InterChange Server.

See “Postmigration considerations” in the WebSphere Process Server V6.2 information center at the following address for more information about the Access EJB support:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/topic/com.ibm.websphere.wps.620.doc/doc/rmig_post_cons_wics.html

Post-migration tasks

When applications have been migrated from IBM WebSphere InterChange Server to WebSphere Process Server with the standard server migration tools, certain areas require attention to make migrated applications work in WebSphere Process Server consistently with their intended function. These areas are a result of the differences between the architectures of WebSphere Process Server and WebSphere InterChange Server.

In this chapter, we discuss the necessary tasks to execute post-migration. We include the following sections:

- ▶ 12.1, “Jython script” on page 254
- ▶ 12.2, “Components covered by the Jython script” on page 255
- ▶ 12.3, “Bindings” on page 258

12.1 Jython script

A Jython script is generated by the server migration tools during the migration, for all WebSphere InterChange Server artifacts that require further customization in WebSphere Process Server. You can run the script by using the **wsadmin** command to create the WebSphere Process Server configuration definitions that correspond to the original WebSphere InterChange Server artifacts. The WebSphere InterChange Server artifacts that are handled in this manner are database connection pools, relationship definitions, and scheduler entries.

The Jython script that is generated is called *InstallAdministrativeObjects.py*. A copy of this script is included in every Java archive (JAR) file that is created by the **reposMigrate** command. The script is placed in the library project that is specified during import in WebSphere Integration Developer. This script is always generated even if no artifact requires it.

You can find the *InstallAdministrativeObjects.py* script and additional information about other post-migration tasks in “Postmigration considerations” in the WebSphere Process Server V6.2 information center at the following Web address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/rmig_post_cons_wics.html

12.1.1 Running the script

You can view the Jython script by using a Jython Editor from the WebSphere Integration Developer Java perspective as shown in Figure 12-1 on page 255. The script is self-explanatory and contains details about how to run the file.

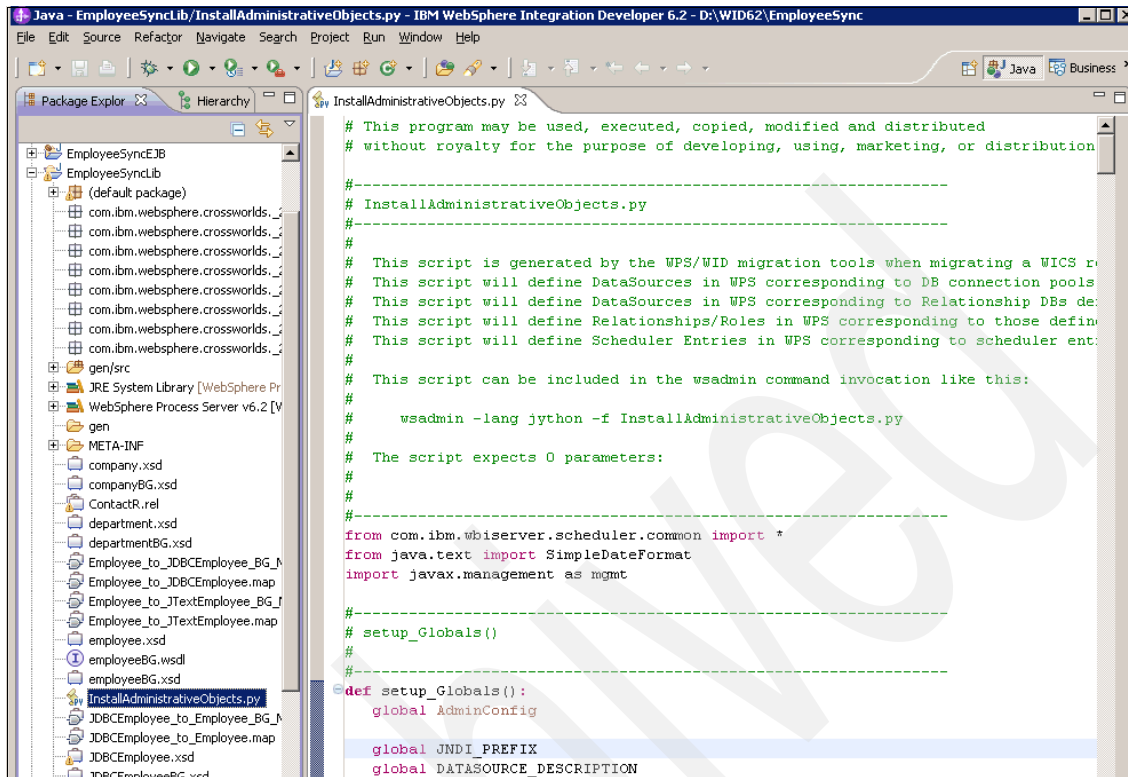


Figure 12-1 Jython script

Run the Jython script by using the following command from the bin folder of WebSphere Process Server installation:

```
wsadmin -lang jython -f \path\InstallAdministrativeObjects.py
```

In this command, *path* is the location of the script folder.

12.2 Components covered by the Jython script

In this section, we discuss the database connection pools, relationships access handling, and scheduler.

12.2.1 Database connection pools

In WebSphere InterChange Server, user-defined database connection pools make it possible for developers to directly access relational databases from within a collaboration or map. A *database connection pool* from WebSphere InterChange Server is rendered as a standard Java Database Connectivity (JDBC) resource in WebSphere Process Server. The basic function is the same. However, the way in which connections and transactions are managed might differ.

The InstallAdministrativeObjects.py script stores information about the JDBC data sources that are needed in WebSphere Process Server that correspond to the user-defined database connection pools in WebSphere InterChange Server. When this script is run, it creates a new JDBC provider and a data source, with the same name as that of the connection pool from WebSphere InterChange Server.

Therefore, if a map or a collaboration uses a connection pool in WebSphere InterChange Server, after migration to WebSphere Process Server, the components continue to work as before. This continuity is possible by using the newly created JDBC data source, without any other modification of the migrated artifacts.

Database provider: If the database provider of WebSphere Process Server differs from the database provider of WebSphere InterChange Server, modify the database-related parameters in WebSphere Process Server by using the administration console. Also, update the Jython script with the WebSphere Process Server database information.

12.2.2 Relationship reuse

A partial or progressive migration is an applicable scenario in a WebSphere InterChange Server to WebSphere Process Server migration, because migration is usually performed in phases. For those circumstances, WebSphere Process Server provides support for reuse or parallel use of existing relationships. With this feature, both WebSphere InterChange Server and WebSphere Process Server can access and update the same relationship instance database.

As in the case of connection pools, the InstallAdministrativeObjects.py script stores information about the JDBC data sources that is needed in WebSphere Process Server. When this script is run, it creates a new JDBC provider and a data source, along with relationship and role definitions. Therefore, you must update the script properly to have the definitions of the relationships in WebSphere Process Server to point to the existing relationships' instances.

Important: You must run the script *before* you deploy the modules that contain the relationship definitions to the WebSphere Process Server Runtime.

Be aware that the Relationship Manager in WebSphere Process Server does not allow administration of the relationship instances that exist in a WebSphere InterChange Server database. If a WebSphere InterChange Server database instance is used for the WebSphere Process Server Relationship Service, Relationship Manager reports an incompatibility error when you try to query or create a relationship instance. In these circumstances, use the WebSphere InterChange Server Relationship Manager to administer the relationship instances.

12.2.3 Scheduler

By scheduling jobs with WebSphere InterChange Server, you can create schedules to manipulate the operational states (start, stop, and pause) of connectors and collaborations. By manipulating component states, you can better manage WebSphere InterChange Server process events. WebSphere InterChange Server can distribute the server's workload over scheduled time periods, thereby reducing traffic and allowing for more efficient resource management.

The application scheduler service in WebSphere Process Server is a WebSphere programming extension that is responsible for starting actions at specific times or intervals. Schedulers are persistent and transactional timer services. Schedulers run Enterprise JavaBeans (EJB) methods or send Java Message Service (JMS) messages by using any Java 2 Platform, Enterprise Edition (J2EE) server application. The scheduler service helps to minimize IT costs and increase application speed and responsiveness by maximizing the utilization of existing computing resources. The scheduler service provides the ability to reliably process workloads, by using parallel processing, and schedule resource-intensive tasks to process when there is low traffic during off hours.

The migration tools generate the Jython script's `InstallAdministrativeObjects.py` file, which contains the WebSphere InterChange Server Scheduler information with a corresponding WebSphere Process Server Scheduler data source. The post migration step requires you to run the **wsadmin** command-line tool to configure these schedules in WebSphere Process Server.

12.3 Bindings

In WebSphere Process Server V6.2, the Migration Wizard supports migrating WebSphere Business Integration Adapters to WebSphere Process Server bindings. See Table 10-2 on page 200 for the details.

After migration, you might need to perform additional tasks. We provide more detail about each type of binding in the next section.

12.3.1 Migrating a WebSphere Business Integration Adapter for EJB to a WebSphere Process Server Stateless Session Bean

The Migration Wizard provides the option to migrate to either a JMS binding connecting to the existing WebSphere Business Integration Adapter or to a new Stateless Session Bean as shown in Figure 12-2. The migration will generate a skeleton Java component, and the component name is Output as shown in Figure 12-2. You need to implement the Output component by double-clicking it as shown in Figure 12-3 on page 259.

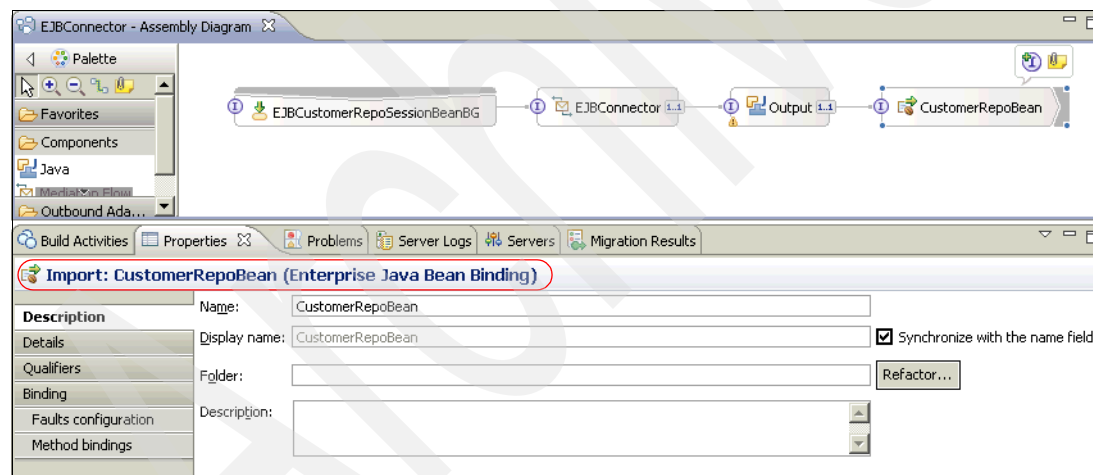
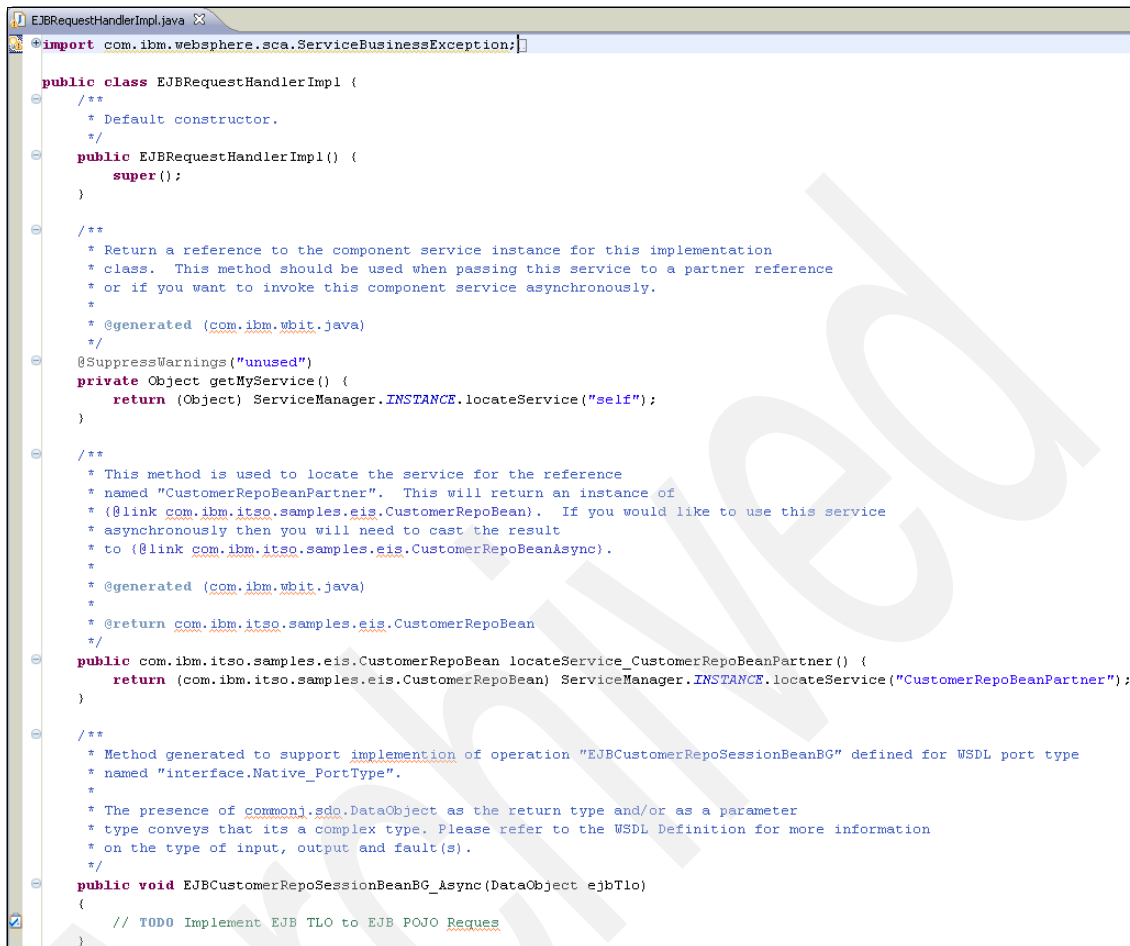


Figure 12-2 Enterprise JavaBean Binding



```

import com.ibm.websphere.sca.ServiceBusinessException;

public class EJBRequestHandlerImpl {
    /**
     * Default constructor.
     */
    public EJBRequestHandlerImpl() {
        super();
    }

    /**
     * Return a reference to the component service instance for this implementation
     * class. This method should be used when passing this service to a partner reference
     * or if you want to invoke this component service asynchronously.
     *
     * @generated (com.ibm.wbit.java)
     */
    @SuppressWarnings("unused")
    private Object getMyService() {
        return (Object) ServiceManager.INSTANCE.locateService("self");
    }

    /**
     * This method is used to locate the service for the reference
     * named "CustomerRepoBeanPartner". This will return an instance of
     * {@link com.ibm.itso.samples.eis.CustomerRepoBean}. If you would like to use this service
     * asynchronously then you will need to cast the result
     * to {@link com.ibm.itso.samples.eis.CustomerRepoBeanAsync}.
     *
     * @generated (com.ibm.wbit.java)
     *
     * @return com.ibm.itso.samples.eis.CustomerRepoBean
     */
    public com.ibm.itso.samples.eis.CustomerRepoBean locateService_CustomerRepoBeanPartner() {
        return (com.ibm.itso.samples.eis.CustomerRepoBean) ServiceManager.INSTANCE.locateService("CustomerRepoBeanPartner");
    }

    /**
     * Method generated to support implementation of operation "EJBCustomerRepoSessionBeanBG" defined for WSDL port type
     * named "interface.Native_PortType".
     *
     * 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 void EJBCustomerRepoSessionBeanBG_Async(DataObject ejbTlo)
    {
        // TODO Implement EJB TLO to EJB POJO Request
    }
}

```

Figure 12-3 EJBRequestHandlerImpl.java

12.3.2 Migrating a WebSphere Business Integration Adapter for HTTP to a WebSphere Process Server HTTP binding

If the WebSphere Business Integration Adapter for HTTP was configured with a protocol listener, you must modify your client application before connecting to the migrated connector. For example, if the connector is called HTTPConnector, listening on port 8080, and using the Uniform Resource Locator (URL) /wbia/samples/webservices, when migrated to WebSphere Process Server, the port will be 9080 (the WC_defaulthost port) and the URL will change to /HTTPConnectorWeb/wbia/samples/http. In WebSphere InterChange Server, you can access this connector using http://localhost:8080/wbia/http/samples. In

WebSphere Process Server, you access this connector using `http://localhost:9080/HTTPConnectorWeb/wbia/http/samples`.

The Migration Wizard provides the option to migrate to either a JMS binding connecting to the existing WebSphere Business Integration Adapter or to a new HTTP binding. The wizard allows you to select a WebSphere InterChange Server default Data Handler, create a WebSphere Process Server Data Handler skeleton, or use a custom Data Handler. The default Data Handler in the Migration Wizard is the XML Data Handler. If you select to create a WebSphere Process Server Data Handler skeleton, implement it after migration.

12.3.3 Migrating a WebSphere Business Integration Adapter for JMS to a WebSphere Process Server JMS binding or generic JMS binding

The Migration Wizard provides the option to migrate to either a JMS binding connecting to the existing WebSphere Business Integration Adapter or to a new JMS binding or generic JMS binding. The wizard allows you to select a WebSphere InterChange Server default Data Handler, create a WebSphere Process Server Data Handler skeleton, or use a custom Data Handler. Choosing the default Data Handler option provides the `CwDataHandler`. If you select to create a WebSphere Process Server Data Handler skeleton, implement it after migration.

If you select the JMS binding, you need to create the generic JMS, Queue Connection Factory, JMS Queues, and JMS Listener Ports, and you also need to generate a new bindings file that includes entries for the queues. See the WebSphere Process Server information center for more information about working with JMS bindings:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.soacore.620.doc/doc/cadm_jmsbinding.html

If you select the generic JMS binding, you need to create the generic JMS, Queue Connection Factory, and JMS Queues, and you also need to generate a new binding file that includes entries for the queues. The listener ports are created during deployment. See the WebSphere Process Server information center for more information about working with generic JMS bindings:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.soacore.620.doc/doc/cadm_scagenbinding.html

12.3.4 Migrating a WebSphere Business Integration Adapter for MQ to a WebSphere Process Server MQ binding or MQ JMS binding

The Migration Wizard provides the option to migrate to either a JMS binding connecting to the existing WebSphere Business Integration Adapter or to a new MQ binding or MQ JMS binding.

The MQ binding or MQ JMS binding supports dynamic endpoint routing. For this reason, the Request queue is not used, and the migration has left this value blank. You need to provide a valid queue in the Request queue field (but we recommend that you provide an unused queue) before the module can be deployed and started. Also, provide a valid Reply queue for the export.

The wizard allows you to select a WebSphere InterChange Server default Data Handler, create a WebSphere Process Server Data Handler skeleton, or use a custom Data Handler. Choosing the default Data Handler option provides the CwDataHandler. If you select to create a WebSphere Process Server Data Handler skeleton, implement it after migration.

If you select MQ JMS binding, you need to additionally configure the Destination Queue and Connection Factory. Open the outbound map for the Mediation Flow Component, and edit the URL in Custom Step #3 to include the Connection Factory name. A sample URL looks like Example 12-1.

Example 12-1 URL generated by the Migration Wizard

```
jms:/queue?destination=MQOUTPUT&connectionFactory=&targetService=Output
```

Enter the value between connectionFactory= and &. The final string looks like Example 12-2.

Example 12-2 Modified URL

```
jms:/queue?destination=MQOUTPUT&connectionFactory=MYCONNECTIONFACTORY&targetService=Output
```

12.3.5 Migrating a WebSphere Business Integration Adapter for Web services to a WebSphere Process Server HTTP binding

If the WebSphere Business Integration Adapter for Web services was configured with a protocol listener, you need to modify your client application before connecting to the migrated connector. For example, if the connector is called WSCorrelator, listening on port 8080, and using the URL /wbia/samples/webservices, then when migrated to WebSphere Process Server,

the port will be 9080 (the WC_defaulthost port), and the URL will change to /WSConnectorWeb/wbia/samples/webservices.

There are two possible scenarios for this type of migration:

- ▶ If the WebSphere Business Integration Adapter for Web services uses the HTTP transport protocol, the Migration Wizard provides the option to migrate to either a JMS binding connecting to the existing WebSphere Business Integration Adapter or to a new HTTP binding.
- ▶ If the WebSphere Business Integration Adapter for Web services uses the JMS transport protocol, the only option for migration is the JMS binding connecting to the existing WebSphere Business Integration Adapter.

Note that in this type of migration, the wizard will not allow you to select a custom Data Handler.

Best practices

In this chapter, we provide the best practices about the standard migration tools that are dedicated to the server artifacts. Best practices can be categorized into two stages. In the premigration stage, best practices define how WebSphere InterChange Server artifacts must be designed. This way, the migration tools are the most efficient to produce the migrated WebSphere Process Server artifacts. In the post-migration stage, best practices define how the generated artifacts can be modified to obtain an optimized solution.

We include the following sections:

- ▶ 13.1, “Artifact preparation” on page 264
- ▶ 13.2, “Post-migration review” on page 272
- ▶ 13.3, “Nonfunctional considerations” on page 273

13.1 Artifact preparation

A set of best practices can help you migrate more easily using the standard migration tools. These guidelines might affect the development of integration artifacts for WebSphere InterChange Server. By adhering to these guidelines, you can ease the migration of WebSphere InterChange Server artifacts to WebSphere Process Server. Use these best practices only as a guide.

Important: Existing WebSphere InterChange Server solutions might not adhere to these guidelines. Furthermore, there might be cases where it is necessary to deviate from these guidelines. In these cases, use care to limit the scope of the deviation to minimize the amount of rework that is required to migrate the artifacts.

The guidelines that are outlined in this section do not encompass best practices for the development of WebSphere InterChange Server artifacts in general. They are limited in scope to those considerations that might affect the ease in which artifacts can be migrated at a future time.

13.1.1 General development

The following best practices are for general development of WebSphere InterChange Server product-based solutions to help ease a future migration:

- ▶ Use WebSphere InterChange Server for real-time, automated process integration solutions.

It is important for integration solutions to adhere to the programming model and architecture that are provided by WebSphere InterChange Server. WebSphere InterChange Server is best suited to real-time, automated process integration solutions. Also, each of the integration components within WebSphere InterChange Server plays a well defined role within the architecture. Significant deviations from this model make it more challenging to migrate content to the appropriate artifacts on WebSphere Process Server.
- ▶ Document the system and component design.

This general best practice greatly improves the success of future migration projects. Be sure to capture the integration architecture and design, including functional design and quality of service (QoS) requirements, the interdependencies of artifacts shared across projects, and the design decisions that were made during the deployment. This guideline assists in system analysis during migration and minimizes any rework efforts.

- Use the development tools to edit integration artifacts.

To create, configure, and modify artifact definitions, it is essential that you use only the development tool that is provided in WebSphere InterChange Server. Avoid manual manipulation of artifact metadata, for example, editing XML files directly, which can corrupt the artifact for migration.

- Use best practices to define rules with the tools and Java snippets.

When developing Java code within collaboration templates, maps, common code utilities, and other components, you must make the following considerations:

- Use only the published APIs.

Use only the APIs that are published in the product documentation for the artifacts. These APIs are outlined in detail in the WebSphere InterChange Server development guides. In many cases, compatibility APIs are provided in WebSphere Process Server, but in certain cases, they are not provided. For the list of supported WebSphere InterChange Server APIs in WebSphere Process Server, see “Supported WebSphere InterChange Server APIs” in the WebSphere Integration Developer V6.2 information center at the following Web address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.migration.ui.doc/topics/rmigwicsapis.html>

- Use the Activity Editor.

Use the Activity Editor tool to the greatest extent possible to design business logic and transformation rules in maps and collaboration templates. By using this tool, you can ensure that the logic is described through metadata, which can more readily be converted to the new artifacts.

Try to avoid field-developed common code utility libraries, included as a Java archive (JAR) file in the classpath of WebSphere InterChange Server, because these libraries must be migrated manually.

- Use adapters to access Enterprise Information Systems (EIS).

In general, ensure that code can interface with an EIS that is placed within the adapters, and not within maps or collaboration templates. This rule is generally a best practice for architecture design.

Use of these adapters also helps avoid prerequisites for third-party libraries and related considerations within the code, such as connection management and possible Java Native Interface (JNI™) implementations.

- Avoid external dependencies in Java snippet code.

Make the Java code in the Java snippet as simple and atomic as possible. The level of sophistication in the Java code must be on the order of scripting, involving basic evaluations, operations, and computations, data formatting, and type conversions.

If more extensive or sophisticated application logic is required, consider using Enterprise JavaBeans (EJB) that run in WebSphere Application Server to encapsulate the logic, and use Web service calls to invoke it from WebSphere InterChange Server.

Use standard Java Developer Kit (JDK) libraries rather than third-party or external libraries that must be migrated separately. Also, collect all related logic within a single code snippet. Avoid using logic where connection and transaction contexts span multiple code snippets. With database operations, for example, place code for obtaining a connection, beginning and ending a transaction, and releasing the connection in one snippet.

- Adhere to Java 2 Platform, Enterprise Edition (J2EE) development practices for portability.

Make the code as safe as possible by using appropriate exception handling. Also, make the code compatible to run within a J2EE application server environment, even though it is currently running within a Java 2 Platform, Standard Edition (J2SE) environment.

Adhere to J2EE development practices, such as avoiding static variables, spawning threads, and disk Input/Output. These practices are excellent best practices to adhere to in general, but they also pertain to portability.

- Do not spawn threads or use thread synchronization primitives.

If you must spawn threads or use thread synchronization primitives, convert them to use asynchronous beans when migrated.

- Do not perform any disk Input/Output by using java.io.*. Use Java Database Connectivity (JDBC) to store any data.

- Do not perform any functions that can be reserved for an EJB container, such as socket Input/Output, classloading, loading native libraries, and so on.

If you must perform such functions, manually convert these snippets to use EJB container functions when migrated.

13.1.2 Common code utilities

Avoid the development of common code utility libraries for use across integration artifacts within the WebSphere InterChange Server environment. Consider using EJBs that run in WebSphere Application Server to encapsulate the logic, and use Web service calls to invoke them from WebSphere InterChange Server.

13.1.3 Business objects

The primary considerations for the development of business objects are to use only the tools that are provided to configure artifacts, to use explicit data types and lengths for data attributes, and to use only the documented APIs.

The following best practices are for business objects:

- Be explicit in the specification of data types.

Business objects within WebSphere Process Server are based on Service Data Objects (SDOs), which use data attributes that are strongly typed. For business objects in WebSphere InterChange Server and adapters, data attributes are not strongly typed. Sometimes, it is common to specify string data types for non-string data attributes. To avoid issues in WebSphere Process Server, be explicit in the specification of data types.

Because business objects within WebSphere Process Server can be serialized at run time as they are passed between components, be *explicit* with the required lengths for data attributes to minimize utilization of system resources. For this reason, do *not* use the maximum 255 character length for a string attribute, for example. Also, do not specify zero-length attributes, which currently default to 255 characters. Instead, specify the exact length that is required for attributes.

- Check for the no-colon-name (NCName) rule.

XSD NCName rules apply to business object attribute names in WebSphere Process Server. Therefore, do not use any spaces or colons (:) in names for business object attributes. Business object attribute names with spaces or colons are invalid in WebSphere Process Server. Rename invalid business object attributes before migrating.

- Use array indexing.

If you use an array in a business object, do not rely on the order of the array when indexing into the array in maps and relationships. The construct to which this array migrates in WebSphere Process Server does not guarantee index order, particularly when entries are deleted.

- Use Business Object Designer.

It is important to use only the Business Object Designer tool to edit business object definitions and to use only the published APIs for business objects within integration artifacts.

13.1.4 Database connection pools

User-defined database connection pools are useful within maps and collaboration templates for simple data lookups and for more sophisticated state management across process instances. A database connection pool in WebSphere InterChange Server is rendered as a standard JDBC resource in WebSphere Process Server. The basic function is the same. The way that connections and transactions are managed, however, might differ.

To maximize future portability, avoid keeping database transactions active across Java snippet nodes within a collaboration template or map. For example, place code that is related to obtaining a connection, beginning and ending a transaction, and releasing the connection in one code snippet.

13.1.5 Collaboration templates

The following best practices are for collaboration templates:

- Process Designer usage

To ensure that processes are described appropriately with metadata, always use the Process Designer tool for the creation and modification of collaboration templates, and avoid editing the metadata files directly. Use the Activity Editor tool wherever possible to maximize the use of metadata to describe the required logic.

- Documented API usage

To minimize the amount of manual rework that might be required in migration, use only the documented APIs within collaboration templates.

To maximize future portability, avoid using explicit connection release calls and explicit transaction bracketing, that is, explicit commits and explicit rollbacks, for user-defined database connection pools. Instead, use the container-managed implicit connection cleanup and implicit transaction bracketing.

Avoid keeping system connections and transactions active across Java snippet nodes within a collaboration template. This rule applies to any connection to an external system, as well as to user-defined database connection pools. As a best practice, manage operations with an external EIS

within an adapter, and ensure that code that is related to database operation is contained within one code snippet.

This best practice might be necessary within a collaboration, which when rendered as a Business Process Execution Language (BPEL) business process component, can be selectively deployed as an interruptible flow. In this case, the process can comprise several separate transactions, with only state and global variable information passed between the activities. The context for any system connection or related transaction that spanned these process transactions is lost.

► Variable scope

Avoid the use of static variables. Instead, use non-static variables or collaboration properties to address the requirements of the business logic.

Use class-level scoping on variables instead of scenario-scoped variables. Scenario scoping is not carried forward during migration.

► Java qualifiers

Avoid the use of the Java qualifiers *final*, *transient*, and *native* in Java snippets. These qualifiers cannot be enforced in the BPEL Java snippets that are the result of migrating the collaboration templates.

Do not use special characters in collaboration template property names. These special characters are invalid in the BPEL property names to which they are migrated. Rename properties to remove these special characters before migrating to avoid syntactical errors in the BPEL that is generated by the migration.

Do not reference variables by using *this* in the name. For example, instead of `this.inputBusObj`, use `inputBusObj`.

Initialize all variables that are declared in Java snippets with a default value, for example, `Object myObject = null`. Be sure that all variables are initialized during declaration before migrating.

Ensure that no Java import statements are in the user modifiable sections of your collaboration templates. In the definition of the collaboration template, use the import fields to specify Java packages to import.

13.1.6 Maps

The following best practices are for maps:

- ▶ Use Map Designer.

To ensure that maps are described appropriately with metadata, always use the Map Designer tool for the creation and modification of maps, and avoid editing the metadata files directly.

Use the Activity Editor tool wherever possible to maximize the use of metadata to describe the required logic.

- ▶ Use a submap when referencing child business objects in a map.
- ▶ Use constants for SET values instead of Java code.

Avoid using Java code as the *value* in a SET, which is not valid in WebSphere Process Server. Use constants instead. For example, the set value `"xml version=" + "1.0" + "encoding=" + "UTF-8"` does not validate in WebSphere Process Server. Instead, change it to `"xml version=1.0 encoding=UTF-8"` before migration.

13.1.7 Relationships

The following best practices are for relationships:

- ▶ Reuse and share relationship instance data.

Relationship definitions are migrated to WebSphere Process Server. The relationship table schema and instance data can be reused by WebSphere Process Server and shared concurrently between WebSphere InterChange Server and WebSphere Process Server.

- ▶ Use published APIs for relationships.

The first key consideration for relationships is to use only the tools that are provided to configure the related components. The second consideration is to use only the published APIs for relationships within integration artifacts.

- ▶ Use Relationship Designer to modify relationship definitions.

Use only the Relationship Designer tool to edit relationship definitions. Additionally, allow only WebSphere InterChange Server to configure the relationship schema, which is generated automatically upon deployment of relationship definitions. Do not alter the relationship table schema directly with the database tools or SQL scripts.

Also, if you must manually modify relationship instance data within the relationship table schema, use the facilities that are provided by Relationship Designer.

13.1.8 Access framework clients

Do not develop any new clients that adopt the Common Object Request Broker Architecture (CORBA) Interface Definition Language (IDL) interface APIs, which are not supported in WebSphere Process Server.

13.1.9 Naming convention

The following best practices are for naming conventions:

- ▶ A restriction exists in the length of the names and paths to the artifacts in Java. In certain operating systems, the length is 255 characters. Try to install the products in a folder with a short path, and do not use names that are too long.
- ▶ Even with the restrictions in the name length, try not to use acronyms and abbreviations unless the abbreviation is widely used.
- ▶ Follow the Java naming convention. The first letter of a word and the first letter of each internal word must be capitalized.
- ▶ For the artifacts, use English characters. These names are used to generate code artifacts. Sometimes, they are mapped by removing spaces and changing the capitalization.
- ▶ Be consistent in the naming of the artifacts. For example, use a suffix in each artifact type as shown:
 - Libraries: *Name* + “Lib”
 - Interfaces: *Name* + “IF”
 - Service Component Architecture (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`

13.1.10 Additional information

You can find additional information about premigration best practices in the WebSphere Process Server V6.2 information center at the following Web address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/doc/rmig_bestwics.html

13.2 Post-migration review

Consider refactoring the generated artifacts after using the migration tools, because they might not always be optimal. The following list provides considerations for this purpose:

- ▶ 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.
- ▶ Use 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. This approach is better from the performance and maintenance point of view.
- ▶ Specify one part per Web services Description Language (WSDL) message based on the Web services Interoperability (WS-I) specification and the Version 6.0 preferred style.
- ▶ Use the WSDL doc-literal style, which is the preferred style in WebSphere Process Server Version 6.0. Ensure that all complex types are given a name and that each complex type can be uniquely identified by its target namespace and name.
- ▶ Consider refactoring your process to use the Process Server Rules engine. If-then rules logic can be externalized by means of business rules. Business rules can be modified at run time without redeploying the applications. Business rules can reduce your process complexity.
- ▶ Consider refactoring your process to use the Process Server State machine. Think of a business process as a series of sequential actions. Also, think of the state machine as a series of loosely related stages that are acted upon. If your application is linear, use a business process. If it is primarily event-driven, contains cyclical patterns, or both, use a state machine. Both editors and the languages that they model are equally valid, and selection of either one depends on the application design.

- ▶ Be as specific in staff assignments as possible.
The more users that are assigned with a task, the more work lists must be updated.
- ▶ Choose between microflows and macroflows.
By default, a process is a microflow, because it is simple and executed much faster than a macroflow. Keep in mind the following considerations for changing a microflow to a macroflow:
 - The business process requires more than one transaction.
 - The business process needs to stop at any point and wait for external input, either in the form of an event or a human task.
 - The business process does not have IBM-specific BPEL extensions enabled for the process.
- ▶ Use compensation pairs and compensation handlers instead of exception handling that is hard-coded to your activities.
Remember that compensation handlers cannot be used in microflows and require you to turn on IBM-specific BPEL extensions to use the compensation pairs.

More information: For more technical considerations, see “Considerations: Post-migration” in the WebSphere Integration Developer, Version 6.2 information center:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.migration.ui.doc/topics/rpostcon.html>

13.3 Nonfunctional considerations

You can tune the generated artifacts after migration to optimize performance. Use the following hints:

- ▶ Separate namespaces in business objects.
If working with more than one business object, place each one in a separate namespace. If they are placed in the same namespace, all of the business objects are loaded each time that a namespace is called, and the overall performance of the tool degrades.
- ▶ Use microflows instead of long running processes.
If modelling a business process with a single transaction, consider making it a microflow. Microflows have great performance and run quickly in the runtime environment.

- ▶ Disable monitoring and tracing whenever possible.

Monitoring and tracing cause a significant performance impact. The default configuration WebSphere Process Server has performance critical logs disabled by default. However, you must double-check that tracing, debugging, and performance monitoring are disabled.

- ▶ Do not use the Apache Derby database for production environments.

WebSphere Process Server uses the Derby database by default. Use of the Derby database simplifies the configuration. However, it is not designed for a production environment where high availability and performance are important.

- ▶ Tune statement cache for long running processes.

BPEL macroflows (long-running processes) extensively use the database for persisting data that is relevant to the process. Persisting various data results in the usage of many separate statements, far more than the default capacity of the data source's cache. Persisting various data results in an excessive number of cache misses, making the caching ineffective. You can resolve this problem 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 System Programming Interface (SPI) Resource Adapter. Its custom property, `maxConcurrency`, under the J2EE Connector Architecture (J2C) activation specification, is used to specify the number of threads that are available to process messages by the application. If a work manager is used, set the maximum number of threads to a value that is high enough to prevent the thread pool from running out of threads.

One symptom of insufficient concurrency is CPU idleness. Vary the concurrency to achieve maximum CPU utilization and throughput.

- ▶ Use an appropriate Java heap size for production environments.

A small Java heap size results in frequent garbage collection. High Java heap size results in long garbage collection cycles. Modern Java virtual machines (JVM) provide many options to optimize garbage collection and memory usage.

- ▶ 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. Only up to 10 asynchronous SCA requests in the module can be processed

concurrently. If the server CPU has not reached its maximum capacity, it sometimes is caused by this setting being low. It can be increased.

- Configure the Object Request Broker (ORB) thread pool parameter.

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 must be configured to maximize the clustering throughput.

The general rule is to use the same number of ORB threads on all application nodes. Also, 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 six-node cluster must be 20.

- Configure the data source connection pools of the relationship and business process engine. The maximum connections property of the relationship data source must be large enough to allow concurrent access to the database from all threads.



Part 4

End-to-end technical solutions

In this part, we provide the step-by-step migration of three end-to-end technical solutions that are based on the commonly used data access and data synchronization interaction patterns. We follow a phased approach to migration. First, we migrate the WebSphere InterChange Server components and WebSphere Business Integration Adapters to WebSphere Process Server components and native bindings by using the migration tool. Next, we upgrade the WebSphere Business Integration Adapters to WebSphere Adapters as appropriate. Finally, we enhance and optimize the migrated artifacts based on the WebSphere Process Server implementation best practices when applicable.

This part includes the following chapters:

- ▶ In Chapter 14, “Preparation for the technical solutions” on page 279, we provide the premigration preparation, hardware and software prerequisites, and setup steps that you must perform before continuing with the migration examples in the subsequent chapters.
- ▶ In Chapter 15, “Data access scenario with technology adapters” on page 293, we provide step-by-step instructions to migrate a data access integration system with technology adapters to integrate Java Database Connectivity

(JDBC), MQ, and Web services that were originally designed for and implemented on WebSphere InterChange Server to WebSphere Process Server.

- ▶ In Chapter 16, “Data synchronization scenario with technology adapters” on page 343, we explain how to migrate a data synchronization integration system with technology adapters to integrate JDBC, MQ, and file systems that were originally designed for and implemented on WebSphere InterChange Server to WebSphere Process Server.
- ▶ In Chapter 17, “Data synchronization scenario with application adapters” on page 431, we explain how to migrate a data synchronization integration system with application adapters to integrate SAP and PeopleSoft that were originally designed for and implemented on WebSphere InterChange Server to WebSphere Process Server.
- ▶ In Chapter 18, “Customize and enhance the solutions” on page 511, we explain how to enhance and optimize the migrated artifacts based on the WebSphere Process Server implementation best practices when applicable.
- ▶ In Chapter 19, “Technical solutions: Troubleshooting” on page 559, we discuss the issues that were encountered when running the end-to-end technical solutions illustrated in this part. We also provide guidance about how to resolve these issues.

Preparation for the technical solutions

In this chapter, we provide details about the required preparation work that you must perform in order to follow the examples in Part 4, “End-to-end technical solutions” on page 277. You must have the required software products installed and running to execute the scenarios. However, you do not need a running WebSphere InterChange Server environment. Instead, exports of WebSphere InterChange Server repository Java archive (JAR) files are provided as starting points for the migration scenarios.

Note: This part does not include any instructions for running the data access and data synchronization scenarios in the WebSphere InterChange Server environment.

The solutions that are provided in Chapter 15, “Data access scenario with technology adapters” on page 293, Chapter 16, “Data synchronization scenario with technology adapters” on page 343, and Chapter 17, “Data synchronization scenario with application adapters” on page 431 use the embedded WebSphere Process Server Unit Test Environment (UTE). However, the solutions have also been tested in a stand-alone WebSphere Process Server environment.

We include the following sections:

- ▶ 14.1, “Setting up the testing environment” on page 281
- ▶ 14.2, “Configuring WebSphere MQ” on page 289
- ▶ 14.3, “Setting up a file system for the JText Connector” on page 290

14.1 Setting up the testing environment

In the following section, we explain the required steps to set up the environment for the migration scenarios.

14.1.1 Software environment

Table 14-1 indicates the required software products and versions to demonstrate the scenarios in Part 4, “End-to-end technical solutions” on page 277.

Table 14-1 Software versions used

Software	Version
WebSphere Integration Developer	6.2
WebSphere Process Server	6.2
WebSphere MQ	6.0.2 Fix Pack 3
WebSphere Business Integration Toolset	4.3.0 Fix Pack 5
DB2	8.1.14.292
WebSphere Business Integration Adapter Framework	2.6.0.11
WebSphere Business Integration Adapter for JText	5.6.5
WebSphere Business Integration Adapter for JDBC	2.6.9
WebSphere Business Integration Adapter for WebSphere MQ	2.8.2
WebSphere Business Integration Adapter for Web services	3.4.7
WebSphere Business Integration Adapter for mySAP.com	6.0.8
WebSphere Business Integration Adapter for PeopleSoft	3.0.1
SAP Front-End GUI Tool	640 Fix Pack 14
XML DataHandler	2.7.0

14.1.2 Setting up the database

The migration scenarios use one table and one sequence that are generated in a locally running IBM DB2 8.2 database. Two scripts are provided that have the SQL to create these database objects.

To set up the database environment:

Important: The ICSREPOS database is used in the migration scenarios for convenience, because WebSphere InterChange Server uses this database. In the following instructions, we add a new table and a new sequence to the database. We made no other modifications. However, do *not* use a production ICSREPOS database for the migration scenarios, because running these scenarios might affect your runtime data.

1. Download the files as explained in Appendix D, “Additional material” on page 651.
2. Store the files on your local drive in c:\setuptemp. Extract the archives to the following directories:
 - c:\setuptemp\Chapter 14 samples
 - c:\setuptemp\Chapter 15 samples
 - c:\setuptemp\Chapter 16 samples
 - c:\setuptemp\Chapter 17 samples

Figure 14-1 on page 283 shows an example of this directory structure and the extracted files.

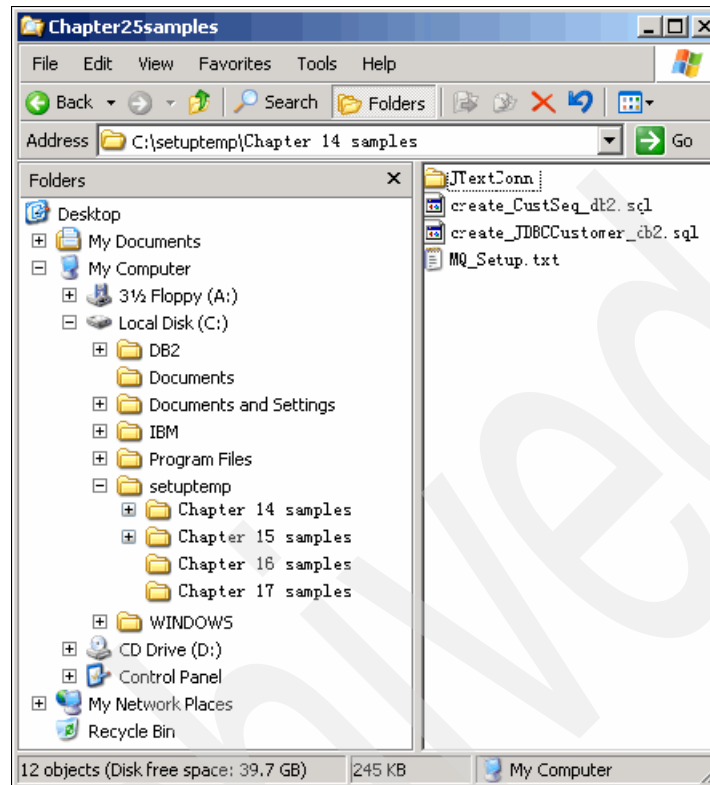


Figure 14-1 Downloading and extracting the files to the setuptemp folder

3. Start DB2 Control Center and connect to the database instance in use. Right-click the **Databases** folder and select **Create Database** → **Standard** (Figure 14-2).

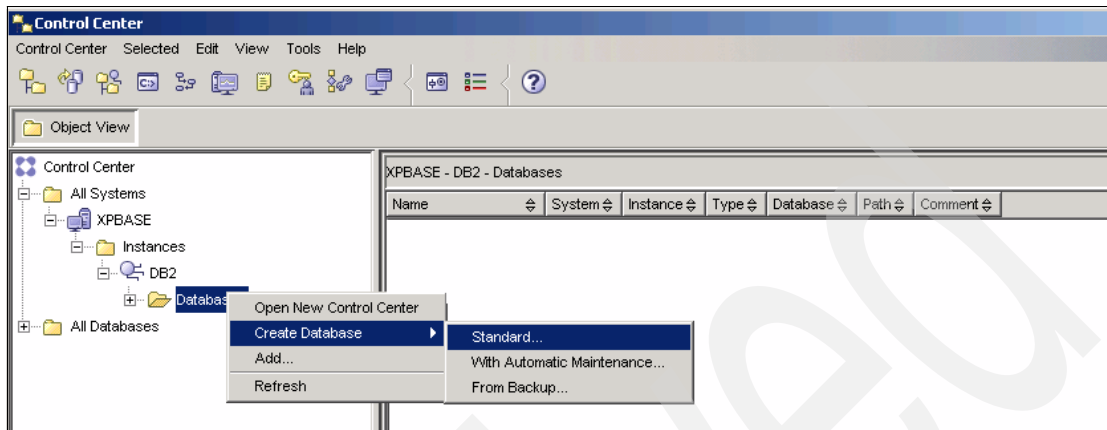


Figure 14-2 Creating a new standard database in DB2 Control Center

4. In the window that opens, for Database name, type ICSREPOS to create a database on your system. Select the Default directory to store this database data.
5. In the DB2 Message information window (Figure 14-3) about the completed task, click **Yes** to open the wizard.

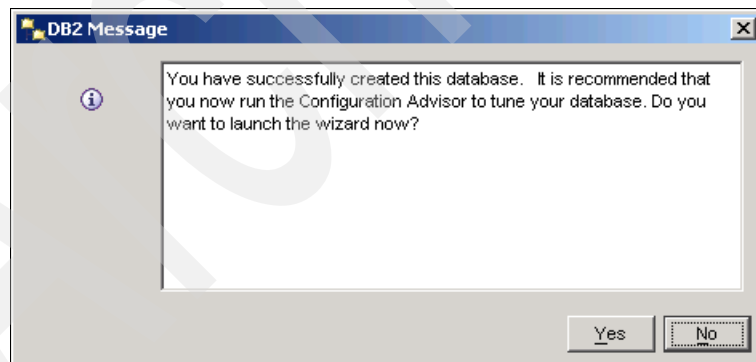


Figure 14-3 Database Configuration Advisor DB2 Message window

6. In the Configuration Advisor window, click **Next** to view configuration recommendations or click **Finish** to apply recommendations.

7. In the DB2 Message confirmation window (Figure 14-4), click **Close**.

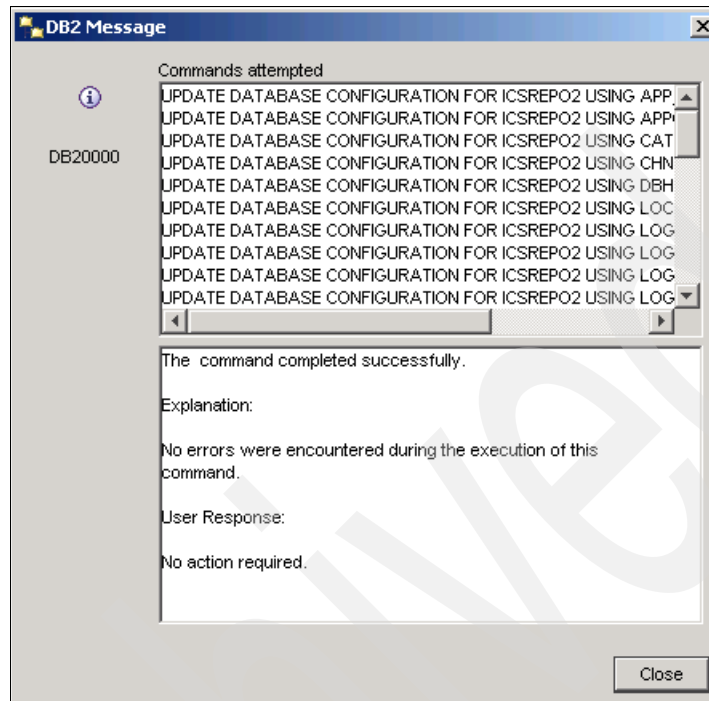


Figure 14-4 Configuration Advisor confirmation DB2 Message window

8. In the database information view (Figure 14-5), in which the newly created database is selected, click the **Query** hyperlink in the lower right pane.

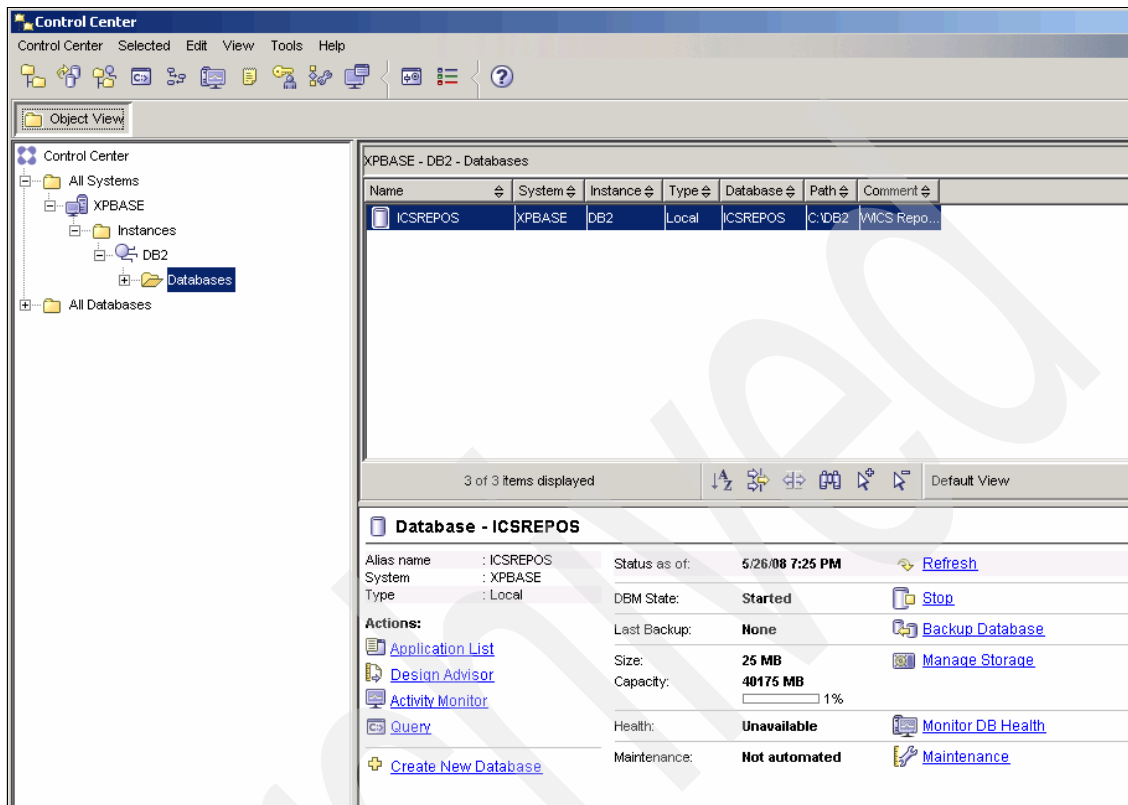


Figure 14-5 ICSREPOS database information view

9. In the Command Editor (Figure 14-6), select the **folder** icon to browse to your setup directory, and open the **create_JDBCCustomer_db2.sql** file to load the contents into the DB2 Command Editor.

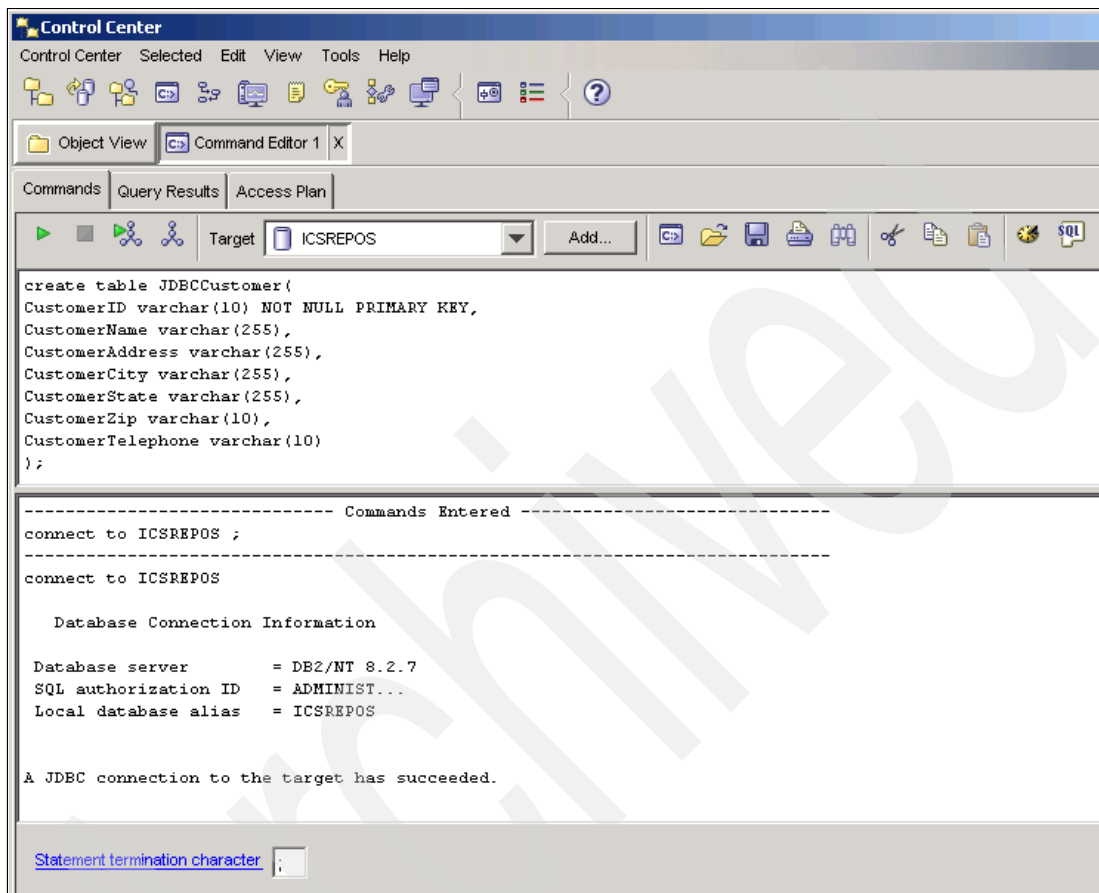


Figure 14-6 DB2 Command Editor

10. Click the **Execute** icon (green triangle arrow) to run the command and create the JDBCCustomer table. Figure 14-7 shows the successful creation of the JDBCCustomer database table.

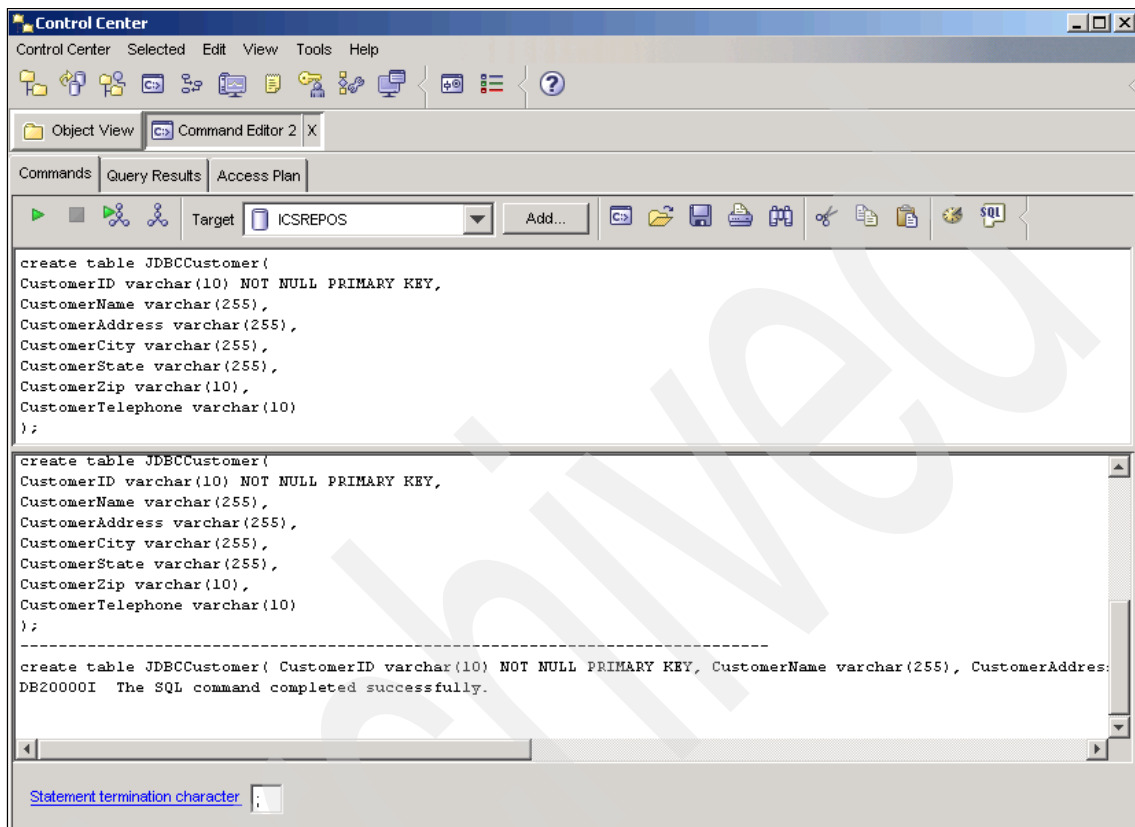


Figure 14-7 Successful creation of the JDBCCustomer database table

11. Repeat steps 9 on page 287 and 10 to create the DB2 sequence CustSeq by using the create_CustSeq_db2.sql script.

14.2 Configuring WebSphere MQ

The migration scenarios use WebSphere MQ as an endpoint application. You can use the WebSphere MQ installation that comes with WebSphere InterChange Server. A script creates a queue manager and the required queues, a channel, and a listener. If there is an existing WebSphere MQ listener, note the port number, and optionally, reuse it in the environment.

To configure WebSphere MQ:

1. From c:\setuptemp\Chapter 14 samples, open the MQ_Setup.txt file and review the contents, which are shown in Example 14-1.

Example 14-1 MQ_Setup.txt

```
DEFINE QLOCAL(CUST.CREATE.Q)
DEFINE QLOCAL(CUST.INPUT.Q)
DEFINE QLOCAL(CUST.UPDATE.Q)
DEFINE QLOCAL(WICS.ARCHIVE.Q)
DEFINE QLOCAL(WICS.ERROR.Q)
DEFINE QLOCAL(WICS.INPROGRESS.Q)
DEFINE QLOCAL(WICS.INPUT.Q)
DEFINE QLOCAL(WICS.OUTPUT.Q)
DEFINE QLOCAL(WICS.REPLYTO.Q)
DEFINE QLOCAL(WICS.UNSUBSCRIBED.Q)
DEFINE QLOCAL(WICS.DUMMY.Q)
DEFINE CHANNEL(CUST.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
DEFINE LISTENER(LISTENER.TCP) trptype(TCP) port(1416)
```

2. Open a command prompt and type the command that is shown in Example 14-2.

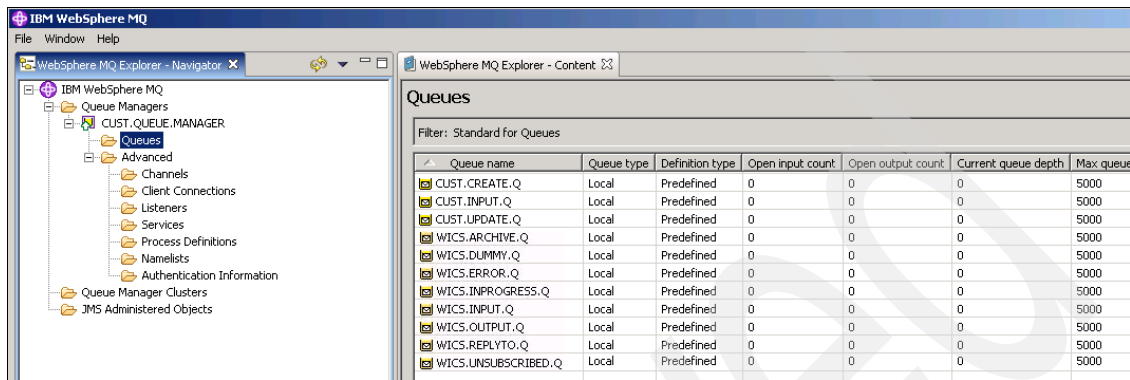
Example 14-2 MQ command

```
cd C:\setuptemp\Chapter 14 samples\

C:\IBM\WebSphereAdapters\templates\configure_mq.bat
CUST.QUEUE.MANAGER MQ_Setup.txt
```

Note: In Example 14-2, we assume that the WebSphere Business Integration Adapters were installed in the C:\IBM\WebSphereAdapters directory. This example creates a queue manager named CUST.QUEUE.MANAGER.

- When the command executes successfully, open WebSphere MQ Explorer. Check to see that the queue manager runs and all queues are created as shown in Figure 14-8. Also, confirm that the listener is running in the Listeners folder. If not, right-click the listener and start it.



The screenshot shows the IBM WebSphere MQ Explorer interface. On the left, a tree view displays the hierarchy: IBM WebSphere MQ > Queue Managers > CUST.QUEUE.MANAGER > Queues. The main pane on the right, titled 'Queues', shows a table of queues for the selected queue manager. The table has columns for Queue name, Queue type, Definition type, Open input count, Open output count, Current queue depth, and Max queue. The filter is set to 'Standard for Queues'.

Queue name	Queue type	Definition type	Open input count	Open output count	Current queue depth	Max queue
CUST.CREATE.Q	Local	Predefined	0	0	0	5000
CUST.INPUT.Q	Local	Predefined	0	0	0	5000
CUST.UPDATE.Q	Local	Predefined	0	0	0	5000
WICS.ARCHIVE.Q	Local	Predefined	0	0	0	5000
WICS.DUMMY.Q	Local	Predefined	0	0	0	5000
WICS.ERROR.Q	Local	Predefined	0	0	0	5000
WICS.INPROGRESS.Q	Local	Predefined	0	0	0	5000
WICS.INPUT.Q	Local	Predefined	0	0	0	5000
WICS.OUTPUT.Q	Local	Predefined	0	0	0	5000
WICS.REPLYTO.Q	Local	Predefined	0	0	0	5000
WICS.UNSUBSCRIBED.Q	Local	Predefined	0	0	0	5000

Figure 14-8 CUST.QUEUE.MANAGER in WebSphere MQ Explorer

14.3 Setting up a file system for the JText Connector

The WebSphere Business Integration Adapter for JText uses a configurable file system structure to receive events, create output, and archive events. We prepared a directory with the appropriate folders to complement the supplied WebSphere Business Integration Adapter for JText configuration file.

To set up a file system for the JText Connector, create the `c:\temp` directory if it does not already exist. Then, from the `C:\setuptemp\Chapter 14 samples\` directory, copy the file system structure `JTextConn` to the `c:\temp` directory. Figure 14-9 shows the folder structure.

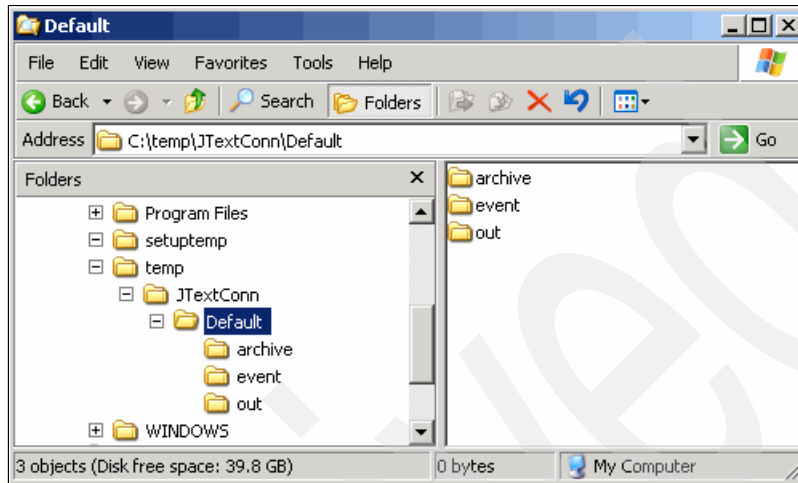


Figure 14-9 JText Connector directory setup

Data access scenario with technology adapters

The migration scenario outlined in this chapter illustrates the steps that are necessary to migrate a data access integration system that was originally designed for WebSphere InterChange Server to WebSphere Process Server. The scenario represents an end-to-end integration system in which back-end systems are accessed by a front-end application through a typical set of WebSphere InterChange Server System components. Together with the WebSphere InterChange Server, these components form an integration system.

The procedure that we demonstrate in this chapter defines the steps to migrate this complete integration system, upgrade certain key components, and tune the runtime environment. By performing the steps that are illustrated in this chapter, you can completely migrate an end-to-end data access scenario to WebSphere Process Server, and then test it.

The scenario uses three WebSphere Business Integration Adapters, which are the WebSphere Business Integration Adapter for Web Services, the WebSphere Business Integration Adapter for JDBC, and the WebSphere Business Integration Adapter for WebSphere MQ. The scenario also uses the WebSphere XML DataHandler. In addition, the scenario uses several business objects, maps, and collaborations.

The data access scenario simulates an integration system that is designed to provide access to two back-end systems:

- ▶ A Customer Relationship Management (CRM) system that is internally accessed through Java Database Connectivity (JDBC)
- ▶ An accounting system that is internally accessed through WebSphere MQ

The integration system exposes a Web services interface that includes two Web service operations, which are `createCustomer` and `retrieveCustomer`. By invoking these Web services, client applications can create new customer records in both the accounting and CRM systems and retrieve existing customer records.

The scenario that is outlined in this chapter is completely independent of other examples and scenarios in this book. However, we use many of the concepts and steps that are necessary to migrate individual artifacts to illustrate the migration of the complete end-to-end scenario.

As a reminder, see Part 2, “Migration implementation concepts” on page 71, for an overall discussion about the possible options for migration implementation. Part 3, “Migration tooling” on page 177, provides detailed information about the standard migration tools.

In this chapter, we include the following sections:

- ▶ 15.1, “Target environment” on page 295
- ▶ 15.2, “Implementation” on page 295
- ▶ 15.3, “Testing the end-to-end solution” on page 335
- ▶ 15.4, “Conclusion” on page 341

15.1 Target environment

In this chapter, we illustrate the migration of an end-to-end scenario from WebSphere InterChange Server to WebSphere Process Server. The original scenario was created with the products that are listed in Table 15-1. The WebSphere InterChange Server System Repository file, which represents the end-to-end integration scenario to be migrated to WebSphere Process Server, is available for download as explained in Appendix D, “Additional material” on page 651.

15.1.1 Software environment

Table 15-1 lists the software products and versions that we used to demonstrate the example in this chapter.

Table 15-1 Software versions used

Software	Version
WebSphere Integration Developer	6.2
WebSphere Process Server	6.2
WebSphere MQ	6.0.2 Fix Pack 2
WebSphere InterChange Server	4.3.0 Fix Pack 5
DB2	8.1.14.292
WebSphere Business Integration Adapter Framework	2.6.0.11
WebSphere Business Integration Adapter JText	5.6.5
WebSphere Business Integration Adapter JDBC	2.6.9
WebSphere Business Integration Adapter WebSphere MQ	2.8.2
WebSphere Business Integration Adapter Web services	3.4.7
XML DataHandler	2.7.0

15.2 Implementation

In this section, we describe the process of migrating the data access scenario (already exported as a WebSphere InterChange Server Repository file) to WebSphere Integration Developer. The resulting modules in WebSphere

Integration Developer can be deployed in WebSphere Process Server as a fully functional system.

More information: See Chapter 14, “Preparation for the technical solutions” on page 279, for information about setting up the required migration development environment.

15.2.1 Premigration overview

The data access scenario is a functional implementation of the data access pattern that is typically implemented in WebSphere InterChange Server. This scenario is restricted to two collaborations, CreateCustomerCollab and RetrieveCustomerCollab, as a means of limiting the scope of the sample migration.

The Create Customer Collaboration (CreateCustomerCollab) exposes a single inbound port (FromCreate) and two outbound ports (ToMQXML and ToJDBC). The ports are bound to instances of the WebSphere Business Integration Adapter for Web Services, WebSphere Business Integration Adapter for WebSphere MQ, and WebSphere Business Integration Adapter for JDBC as illustrated in Figure 15-1.

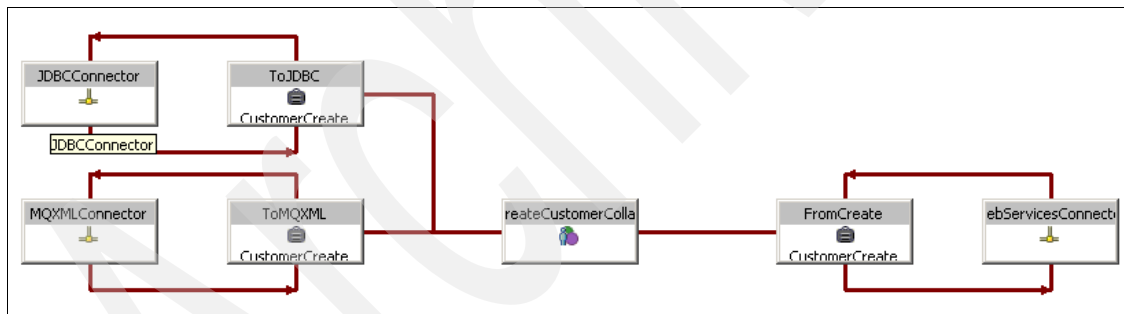


Figure 15-1 Create Customer Collaboration flow diagram

The Retrieve Customer Collaboration (RetrieveCustomerCollab) exposes a single inbound port (FromRetrieve) and a single outbound port (ToJDBC). The ports are bound to instances of the WebSphere Business Integration Adapter for Web Services and the WebSphere Business Integration Adapter for JDBC as illustrated in Figure 15-2.

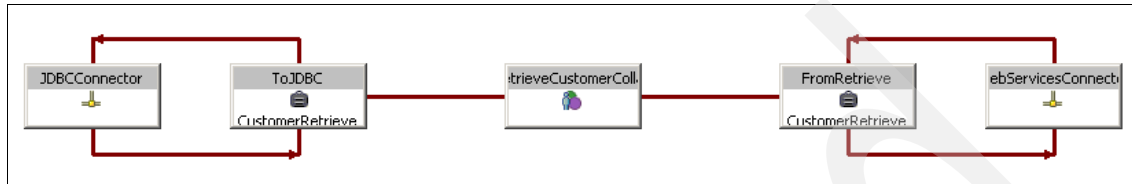


Figure 15-2 Retrieve Customer Collaboration flow diagram

15.2.2 Development

You can migrate the data access scenario, which is already exported as a WebSphere InterChange Server Repository file, as we explain in this section. The ICSDataAccessScenario.jar file represents the WebSphere InterChange Server Repository file.

Exporting a WebSphere InterChange Server repository file: From the WebSphere Business Integration System Manager, select an appropriate integration component library, right-click, and select **Export as** → **Repository File**.

Files for download: The samples in this section are available for download as explained in Appendix D, “Additional material” on page 651.

To migrate the business objects and maps of the data access scenario:

1. Create a new workspace in WebSphere Integration Developer.
2. Switch to the Business Integration Perspective View.

3. Create a new library in the workspace:
 - a. Select **File** → **New** → **Library**.
 - b. In the New Library window (Figure 15-3), for Library Name, type `DataAccessScenarioLibrary` for the new library. Select **Use default location** and click **Finish**.

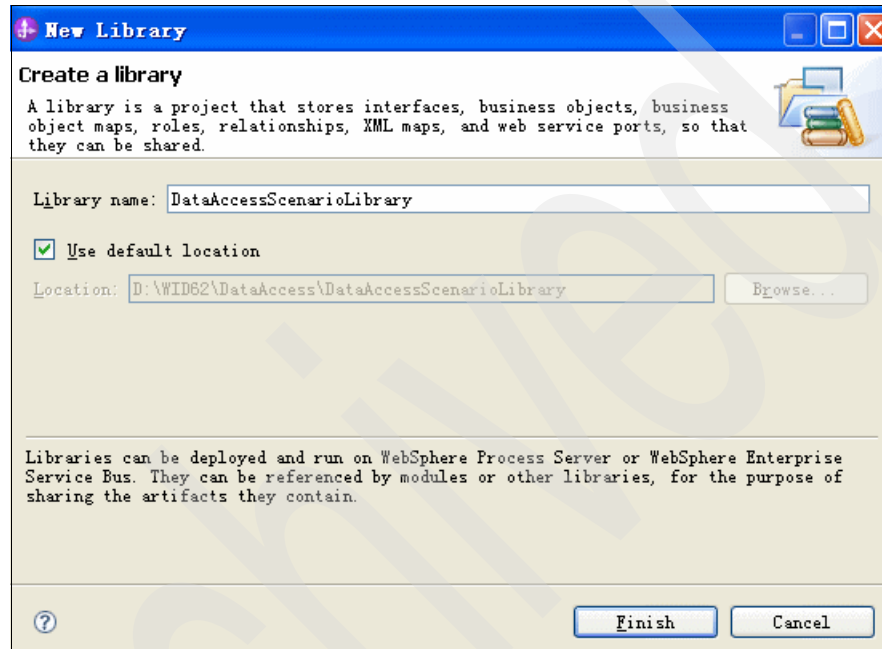


Figure 15-3 New data access scenario library

4. Import the WebSphere InterChange Server Repository file named ICSDataAccessScenario.jar by using the WebSphere InterChange Server Migration Wizard:
 - a. Select **File** → **Import**.
 - b. In the Import – Select window (Figure 15-4), under Business Integration, select **WebSphere InterChange Server Repository**, and click **Next**.

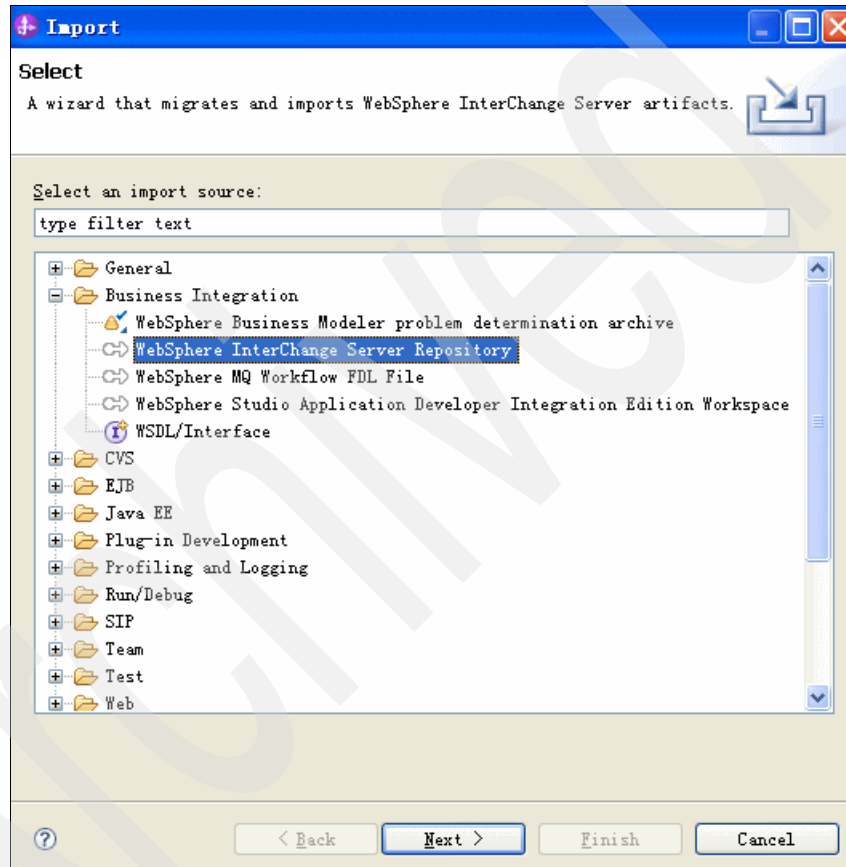


Figure 15-4 WebSphere InterChange Server JAR file Import wizard

5. In the WebSphere InterChange Server Import Wizard migration window (Figure 15-5):
 - a. For WebSphere InterChange Server repository path, select the **ICSDDataAccessScenario.jar** file.
 - b. For WebSphere Integration Developer library name, select the **DataAccessScenarioLibrary**.
 - c. Click **Next** to continue.

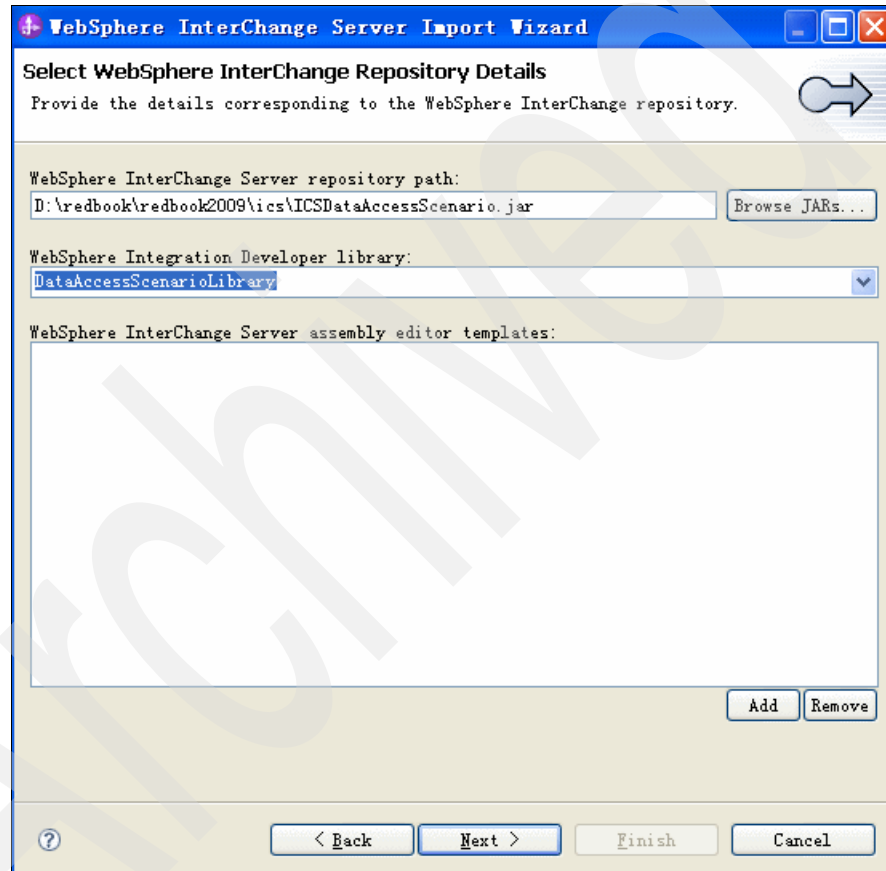


Figure 15-5 Migration Import wizard

For more details about how to run the Migration Wizard, see 10.3, “Migration Wizard” on page 192.

6. Configure the migration settings for each connector as shown in Figure 15-6:
 - a. For WebServicesConnector, select **HTTP Binding for WebServices**.
 - b. For MQXMLConnector, select **MQ Binding** and choose **Use the default WICS data handler**.
 - c. For JDBCConnector, select **JMS to JDBC WBI Adapter** and select **Import and reference the JCA connector**.
 - d. Click **Next** to continue.

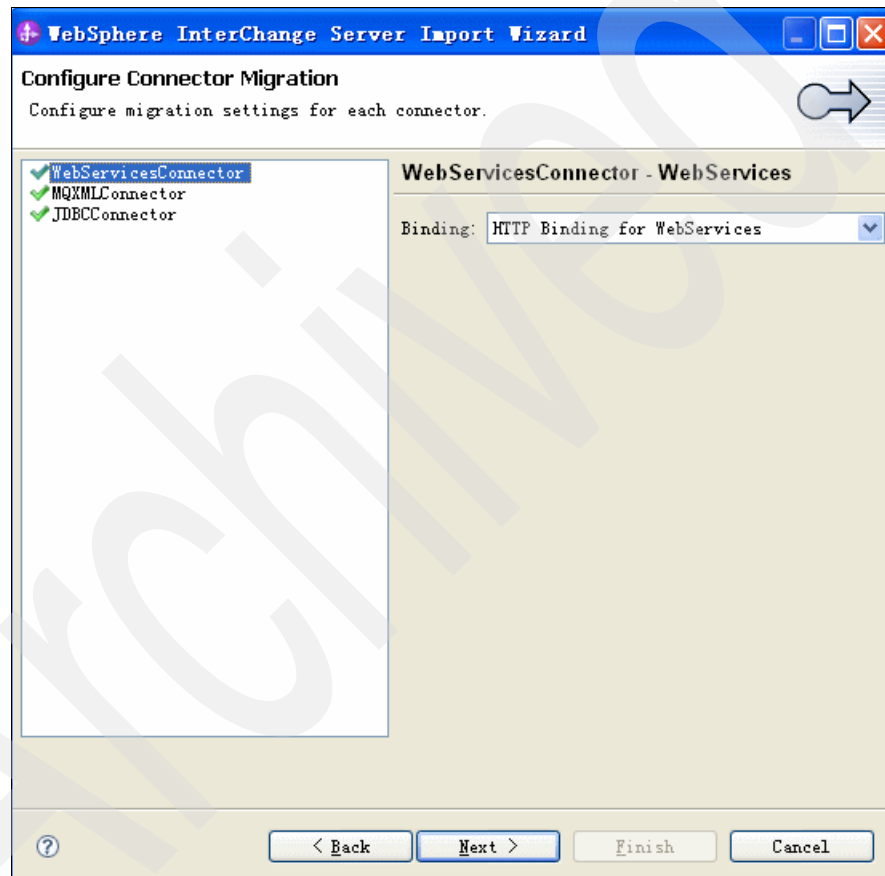


Figure 15-6 Configure Connector Migration

7. Click **Next**. The Conversion Options page opens. From here, you accept the recommended options as shown in Figure 15-7 on page 302.

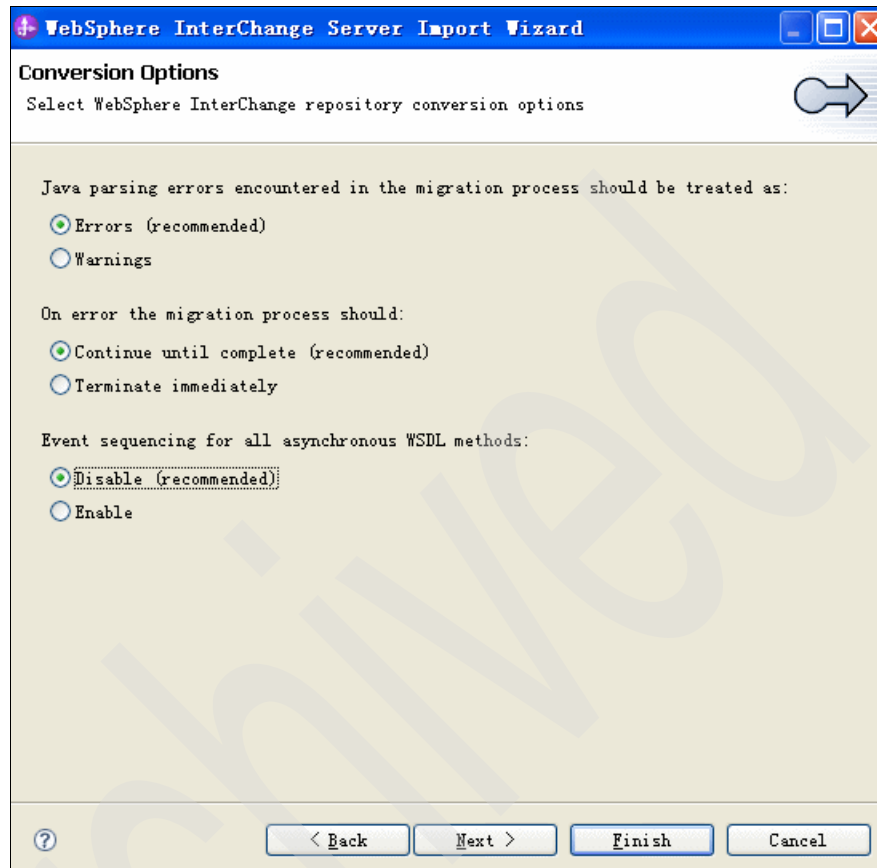


Figure 15-7 Conversion Options

8. Click **Next**. A Migration Summary page opens similar to Figure 15-8 on page 303.

WebSphere InterChange Server Import Wizard

Migration Summary
Summary of the WebSphere InterChange repository migration.

Repository Path: D:\redbook\redbook2009\ics\ICSDDataAccessScenario.jar
Library: DataAccessScenarioLibrary

Java parsing errors encountered in the migration process should be treated as: Errors (recommended)
On error the migration process should: Continue until complete (recommended)
Event sequencing for all asynchronous WSDL methods: Disable (recommended)

WebServicesConnector
Connector Type: WebServices
Target Binding: HTTP Binding for WebServices
Data Handler Type: Default Data Handler

MQXMLConnector
Connector Type: MQ
Target Binding: MQ Binding
Data Handler Type: Default Data Handler

JDBCConnector
Connector Type: JDBC
Target Binding: JMS to JDBC WBI Adapter
Import JCA Connector: true

? < Back Next > Finish Cancel

Figure 15-8 Migration Summary

9. After you have reviewed the summary details, click **Finish** to begin the migration process.

A progress bar at the bottom of the migration dialog indicates the progress of the migration. After the process has completed, the dialog disappears and the Migration Results window opens as shown in Figure 15-9 on page 304.

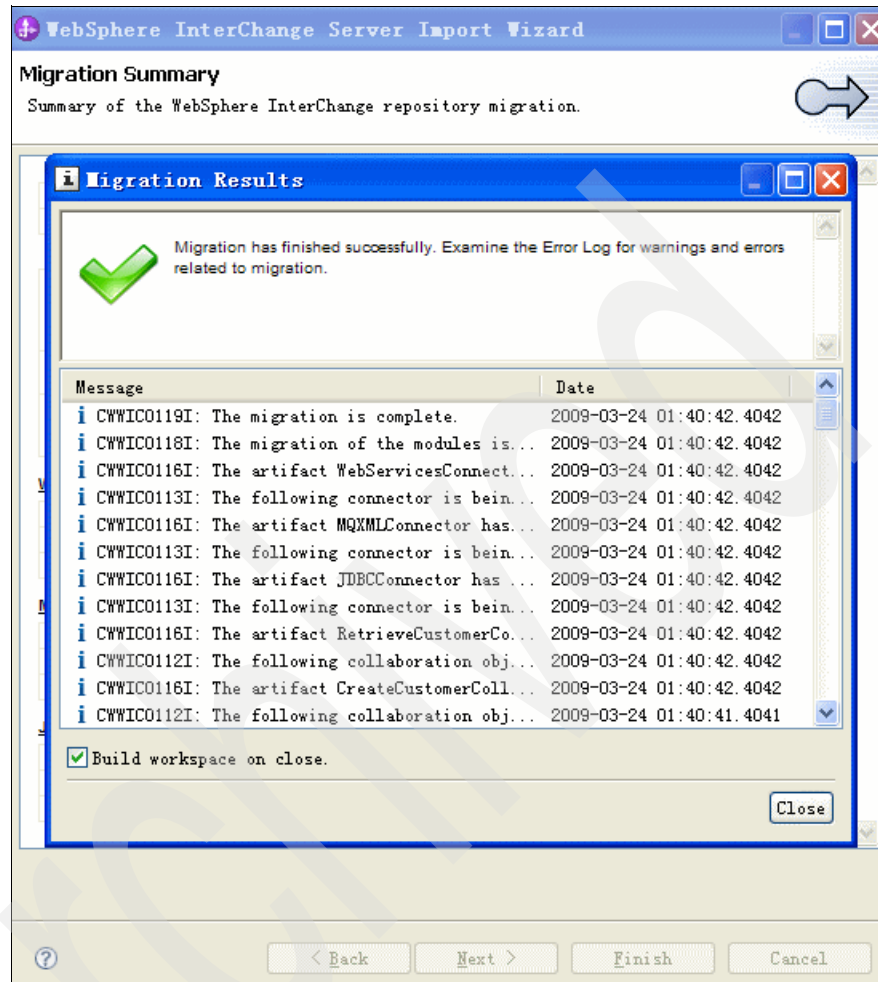


Figure 15-9 Migration Results

10. Click **Close** to finish the process and to build the new workspace.

After running the WebSphere InterChange Server Migration Wizard, the workspace contains the six new modules in addition to the DataAccessScenario library that you previously created (Figure 15-10 on page 305). Three of the new modules correspond to the three WebSphere Business Integration Adapters that are used in the scenario. The modules are JDBCConnector, MQXMLConnector, and WebServicesConnector. The MQXMLConnector module includes the MQ binding with WebSphere InterChange Server Data Handler. The WebServicesConnector includes the HTTP binding. The other two new modules, CreateCustomerCollab and RetrieveCustomerCollab, correspond to the

collaborations from the WebSphere InterChange Server system. The CWYBC_JDBC module includes the package WebSphere Adapter for JDBC. We will use the CWYBC_JDBC module for migrating WebSphere Business Integration Adapter for JDBC to WebSphere Adapter for JDBC later.

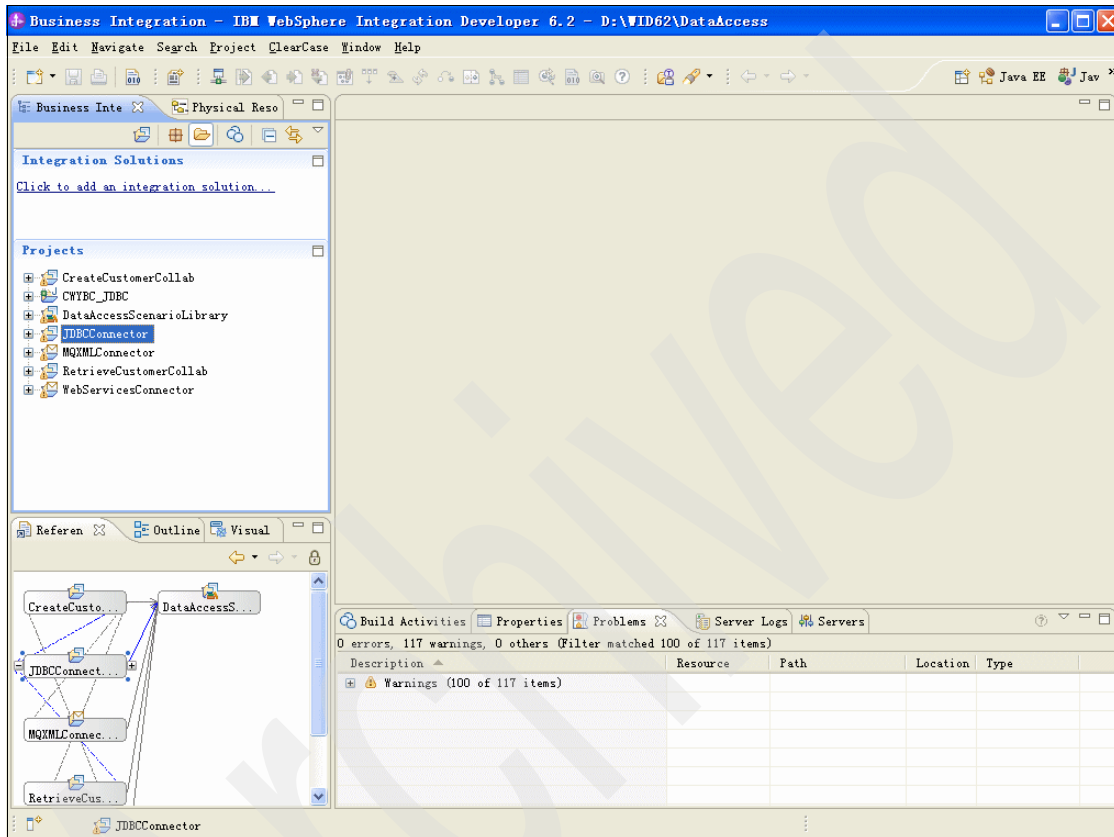


Figure 15-10 Module created

The business objects and maps are migrated to the DataAccessScenarioLibrary. The connector modules contain the mediations that convert Generic-To-Native and Native-To-Generic business objects.

To migrate JDBCCConnector to WebSphere Adapter for JDBC by using the adapter Migration Wizard:

1. In the WebSphere Integration Developer workspace, go to the Java EE perspective by selecting the **Window** → **Open perspective** → **Other** → **Java EE**. Right-click the connector project **CWYBC_JDBC**, and select **Update** → **Update Connector Project**.

Figure 15-11 describes the functional areas of the wizard.

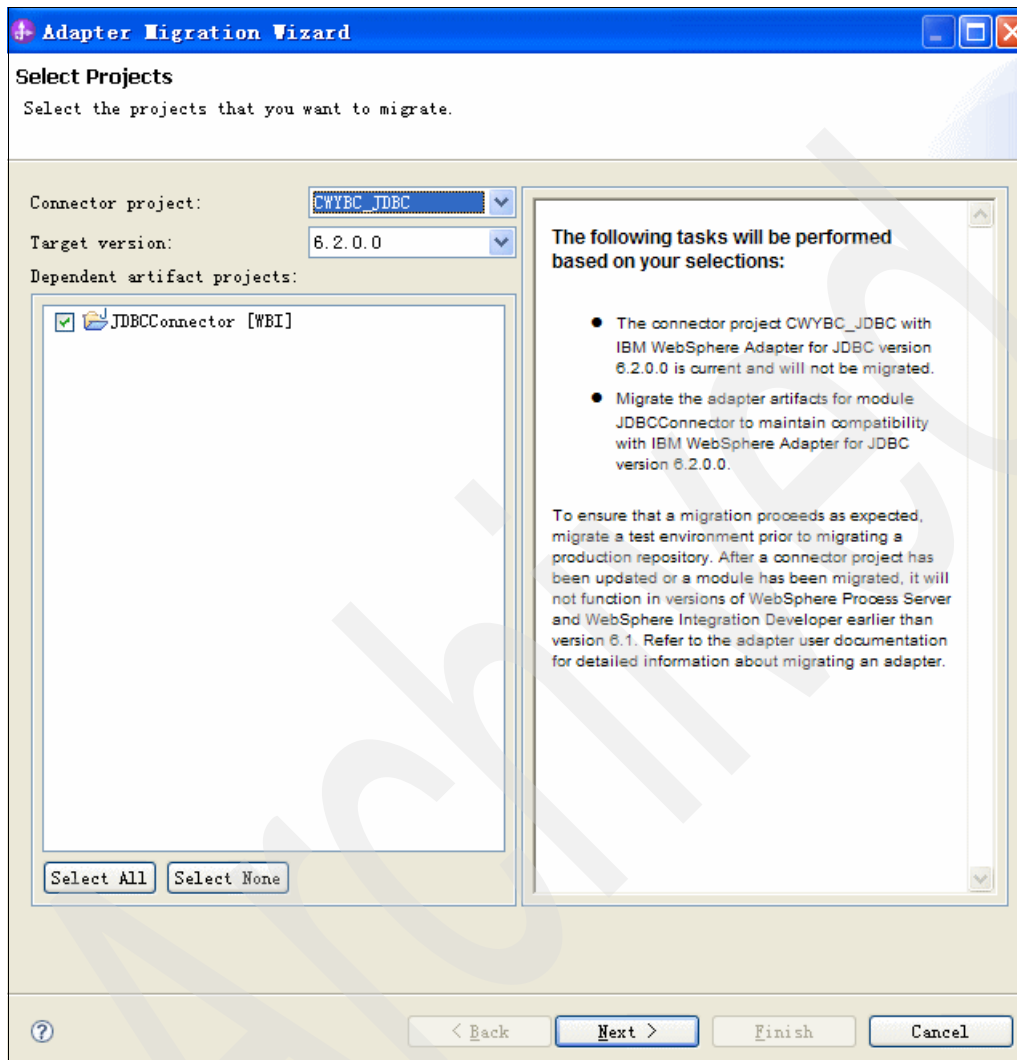


Figure 15-11 Select Projects

When you launch the Migration Wizard from the connector project context menu while in the Java EE perspective, by default, all of the dependent artifact projects are selected. If you deselect a dependent artifact project, that project is not migrated.

2. As shown in Figure 15-12 on page 307, on the Review changes window, you can review the migration changes that will occur in each of the artifacts that you are migrating by clicking the + (plus) sign.

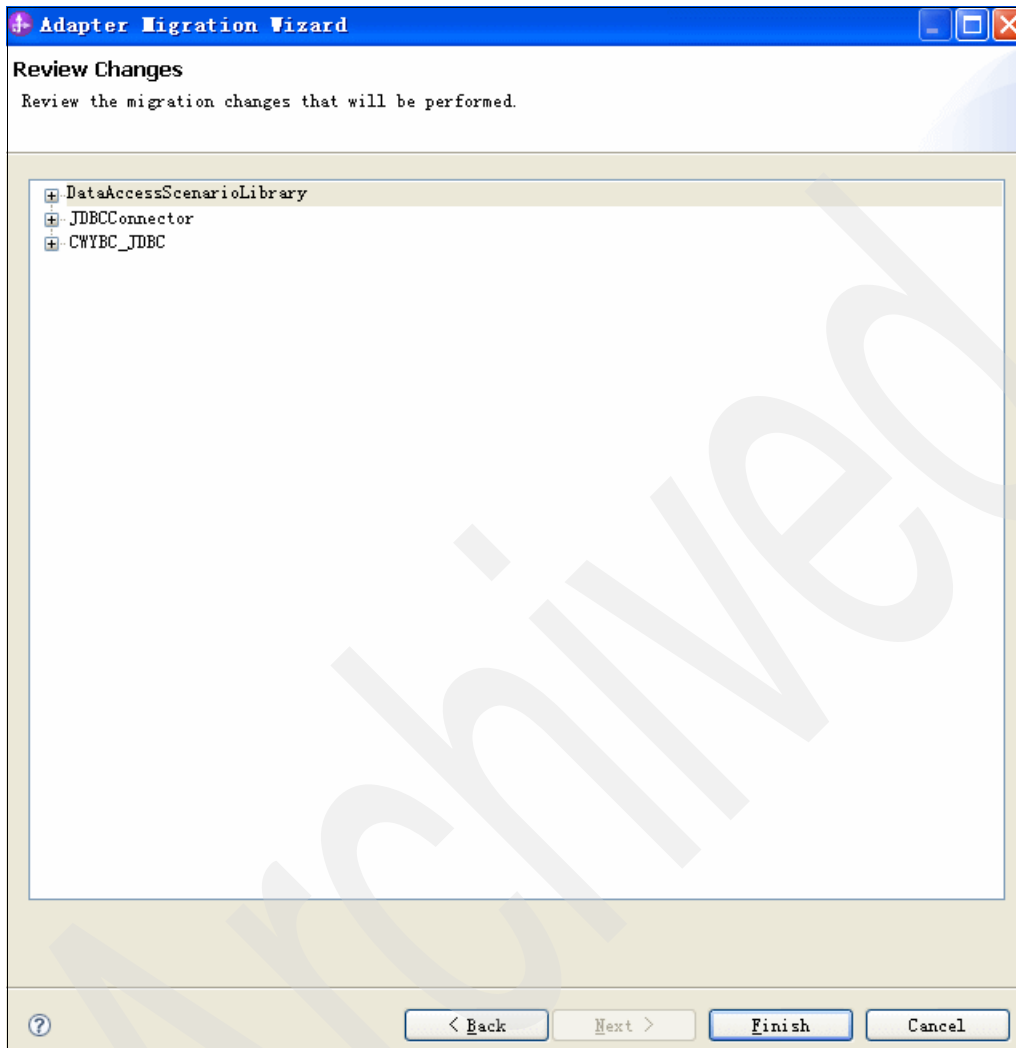


Figure 15-12 Review changes

3. Click **Finish** to perform the migration.

Before performing the migration process, the wizard backs up all of the projects that are affected by the migration. The projects are backed up to a temporary folder within the workspace. If the migration fails for any reason, or if you decide to cancel the migration before it completes, the wizard deletes the modified projects and replaces them with the projects that are stored in the temporary folder.

The CreateCustomerCollab and RetrieveCustomerCollab modules in WebSphere Integration Developer contain the logic that existed previously in the collaboration templates in the WebSphere InterChange Server environment. Open the CreateCustomerCollab Assembly Diagram and examine the process component, CreateCustomerCollab_FromCreate, to see how the Migration Wizard translated the collaboration template logic as shown in Figure 15-13.

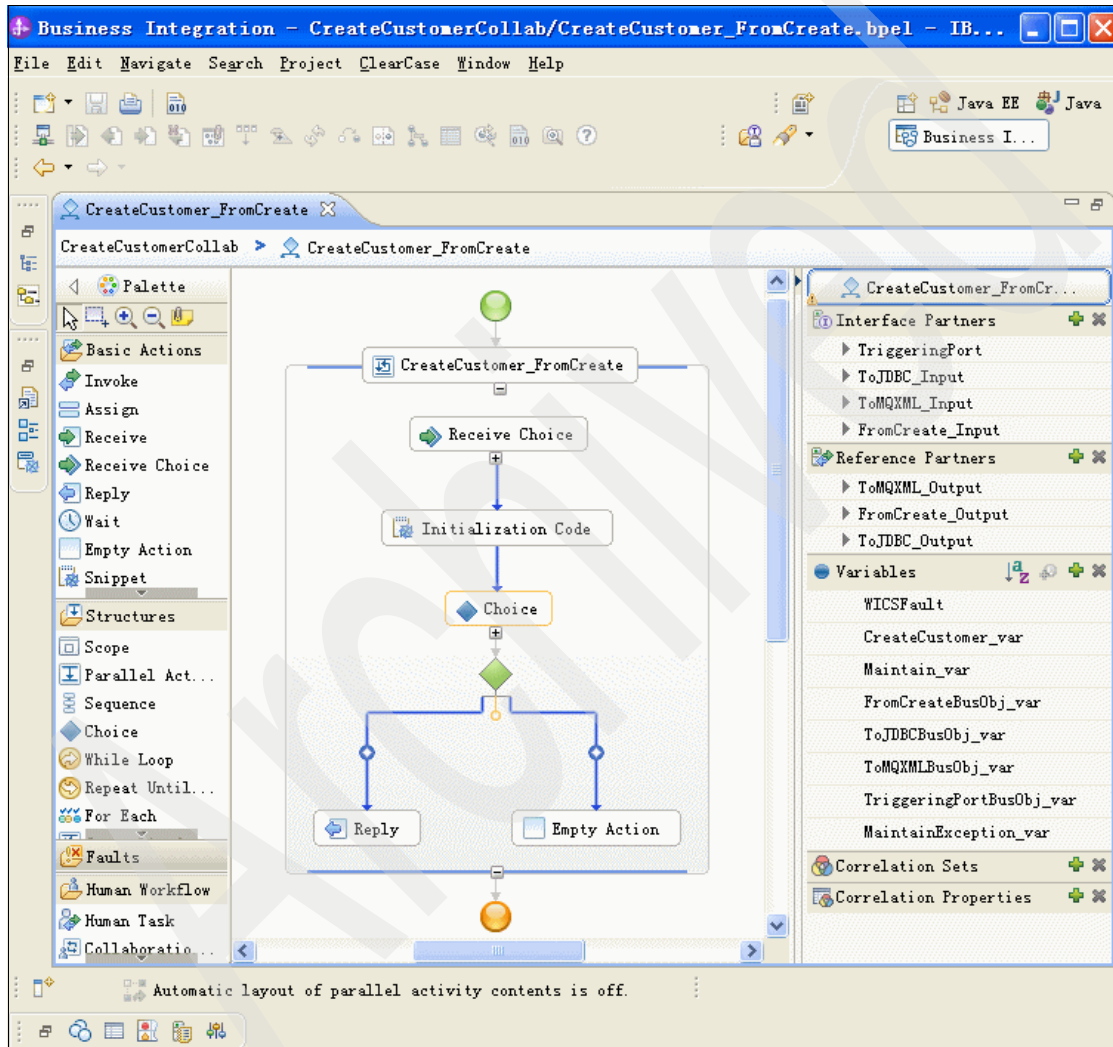


Figure 15-13 Create Customer Process Component

After migration, the JDBCConnector module in WebSphere Integration Developer contains the WebSphere Adapter for JDBC as shown in Figure 15-14.

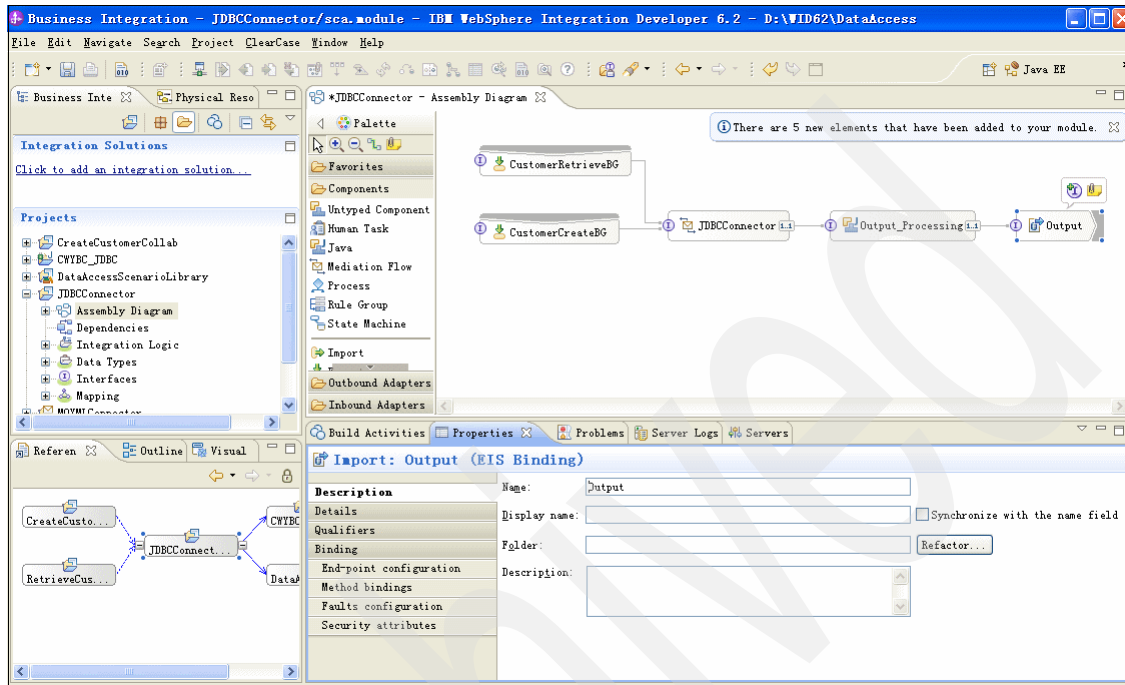


Figure 15-14 JCA JDBC Adapter

After migration, the MQXMLConnector module in WebSphere Integration Developer contains the WebSphere Process Server MQ binding as shown in Figure 15-15 on page 310.

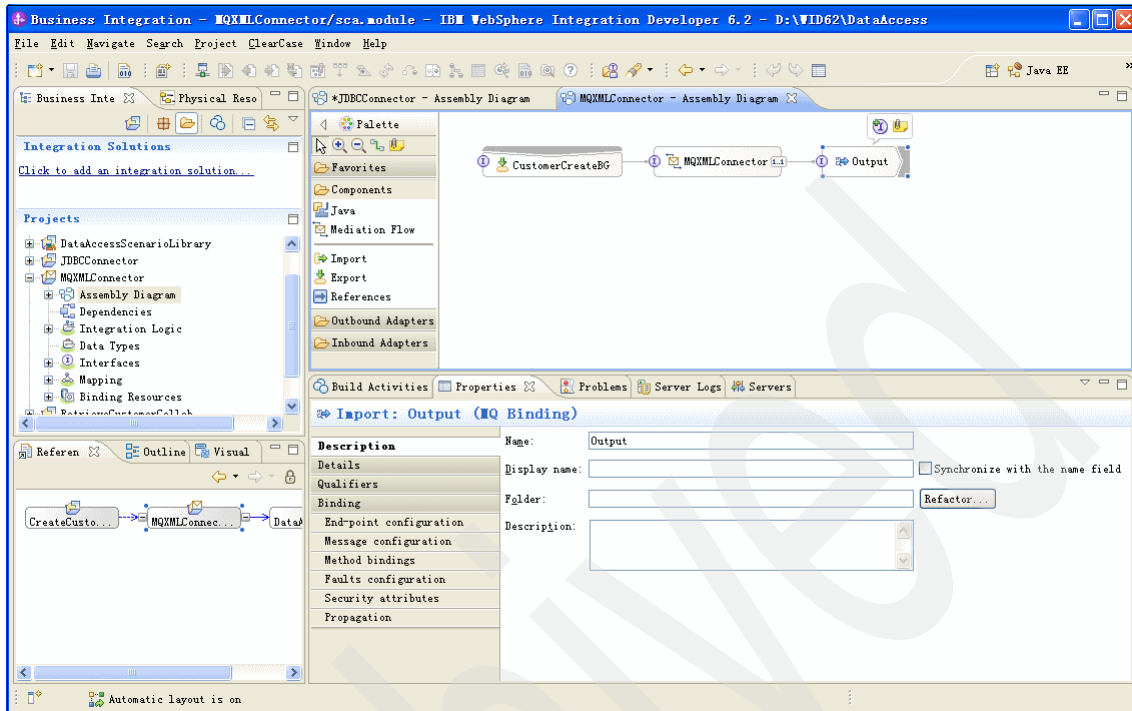


Figure 15-15 MQ binding

After migration, the WebServicesConnector module in WebSphere Integration Developer contains the WebSphere Process Server HTTP binding as shown in Figure 15-16 on page 311.

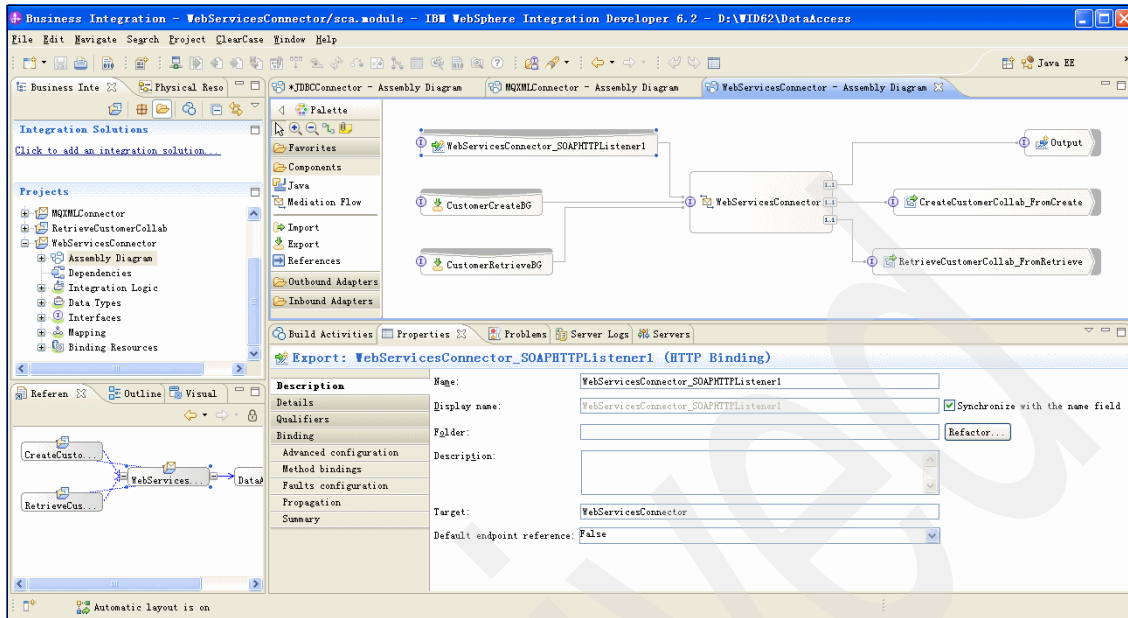


Figure 15-16 HTTP binding

After migrating by using the Migration Wizard and Adapter Migration Wizard, you still have several manual steps to perform:

1. The MQ binding supports dynamic endpoint routing. Therefore, the Request queue is not used, and the migration has left this value blank. You need to provide a valid queue (but we recommend an unused queue in case of problems) before the module can be deployed and started:
 - a. In the WebSphere Integration Developer workspace, open the **Assembly Diagram** of MQXMLConnector module. Double-click the **Output** MQ binding to open the binding.
 - b. In the Properties view, select the **End-point configuration** tab. In the Send destination queue, enter WICS.DUMMY.Q.

Note: Make sure that you create the WICS.DUMMY.Q queue in the MQ queue manager. Here, the queue manager is CUST.QUEUE.MANAGER.

2. The Adapter Migration Wizard changes adapter business object (BO) definitions after they migrate. Certain attributes are lost. This problem will be fixed in WebSphere Process Server V6.2 Fix Pack.

For now, to work around this problem, execute these steps:

- a. Modify the CreateCustomer_FromCreate BPEL in the CreateCustomerCollab module:
 - i. In the WebSphere Integration Developer workspace, open the **Assembly Diagram** of CreateCustomerCollab module. Double-click the **CreateCustomerCollab_FromCreate** BPEL to open the Business Process Editor.
 - ii. In the Properties view, select the **Java Imports** tab, and find the import that is shown in Example 15-1. Change it to the new BO import that is shown in Example 15-2.

Example 15-1 Old BO import

```
import com.ibm.websphere.bo.BOFactory;
```

Example 15-2 New BO import

```
import  
com.ibm.websphere.bo.*;
```

- iii. Select the **Call JDBC** java snippet activity. In the Properties view, select the **Details** tab.
- iv. Find the code that is shown in Example 15-3 and replace it with the code that is shown in Example 15-4.

Example 15-3 Old copy code in Call JDBC java snippet activity

```
var_3.copy(var_4);  
}  
if (triggeringBusObj == null) { TriggeringPortBusObj_var = null;  
} else { TriggeringPortBusObj_var =  
triggeringBusObj.getBusinessGraph(); }  
if (ToJDBCBusObj == null) { ToJDBCBusObj_var = null; } else {  
ToJDBCBusObj_var = ToJDBCBusObj.getBusinessGraph(); }  
}
```

Example 15-4 New copy code in Call JDBC java snippet activity

```
BOCopy boCopy = (BOCopy)  
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOCopy");  
ToJDBCBusObj_var = boCopy.copy(TriggeringPortBusObj_var);  
}
```

- v. Select the **Copy Result to triggeringBO** java snippet activity. In the Properties view, select the **Details** tab.
- vi. Find the code that is shown in Example 15-5, and replace it with the code that is shown in Example 15-6.

Example 15-5 Old copy code in Copy Result to triggeringBO java snippet activity

```
var_5.copy(var_7);  
}  
if (triggeringBusObj == null) { TriggeringPortBusObj_var = null;  
} else { TriggeringPortBusObj_var =  
triggeringBusObj.getBusinessGraph(); }  
if (ToJDBCBusObj == null) { ToJDBCBusObj_var = null; } else {  
ToJDBCBusObj_var = ToJDBCBusObj.getBusinessGraph(); }  
}
```

Example 15-6 New copy code in Copy Result to triggeringBO java snippet activity

```
BOCopy boCopy = (BOCopy)  
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOCopy");  
TriggeringPortBusObj_var = boCopy.copy(ToJDBCBusObj_var);  
}
```

- vii. Select the **Call MQ** java snippet activity. In the Properties view, select the **Details** tab.
- viii. Find the code that is shown in Example 15-7. Replace it with the code that is shown in Example 15-8.

Example 15-7 Old copy code in Call MQ java snippet activity

```
var_6.copy(var_5);  
}  
if (triggeringBusObj == null) { TriggeringPortBusObj_var = null;  
} else { TriggeringPortBusObj_var =  
triggeringBusObj.getBusinessGraph(); }  
if (ToMQXMLBusObj == null) { ToMQXMLBusObj_var = null; } else {  
ToMQXMLBusObj_var = ToMQXMLBusObj.getBusinessGraph(); }  
}
```

Example 15-8 New copy code in Call MQ java snippet activity

```
BOCopy boCopy = (BOCopy)  
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOCopy");  
ToMQXMLBusObj_var = boCopy.copy(TriggeringPortBusObj_var);  
}
```

- b. Modify the RetrieveCustomer_FromRetrieve BPEL in the RetrieveCustomerCollab module:
 - i. In the WebSphere Integration Developer workspace, open the **Assembly Diagram** of RetrieveCustomerCollab module. Double-click the **RetrieveCustomerCollab_FromRetrieve** BPEL to open the Business Process Editor.
 - ii. In the Properties view, select the **Java Imports** tab. Find the import that is shown in Example 15-9 and change it to the new BO import that is shown in Example 15-10.

Example 15-9 Old BO import

```
import com.ibm.websphere.bo.BOFactory;
```

Example 15-10 New BO import

```
import com.ibm.websphere.bo.*;
```

- iii. Select the **UNNAMED_ACTIVITY_6** java snippet activity. In the Properties view, select the **Details** tab.
- iv. Find the code that is shown in Example 15-11, and replace it with the code that is shown in Example 15-12.

Example 15-11 Old copy code in UNNAMED_ACTIVITY_6 java snippet activity

```
var_17.copy(var_2);
}
if (triggeringBusObj == null) { TriggeringPortBusObj_var = null;
} else { TriggeringPortBusObj_var =
triggeringBusObj.getBusinessGraph(); }
if (ToJDBCBusObj == null) { ToJDBCBusObj_var = null; } else {
ToJDBCBusObj_var = ToJDBCBusObj.getBusinessGraph(); }
```

Example 15-12 New copy code in UNNAMED_ACTIVITY_6 java snippet activity

```
BOCopy boCopy = (BOCopy)
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOCopy");
ToJDBCBusObj_var = boCopy.copy(TriggeringPortBusObj_var);
}
```

- v. Select the **UNNAMED_ACTIVITY_8** java snippet activity. In the Properties view, select the **Details** tab.
- vi. Find the code that is shown in Example 15-13 on page 315, and replace it with the code that is shown in Example 15-14 on page 315.

Example 15-13 Old copy code in UNNAMED_ACTIVITY_8 java snippet activity

```
var_13.copy(var_15);  
}  
if (triggeringBusObj == null) { TriggeringPortBusObj_var = null;  
} else { TriggeringPortBusObj_var =  
triggeringBusObj.getBusinessGraph(); }  
if (ToJDBCBusObj == null) { ToJDBCBusObj_var = null; } else {  
ToJDBCBusObj_var = ToJDBCBusObj.getBusinessGraph(); }  
}
```

Example 15-14 New copy code in UNNAMED_ACTIVITY_8 java snippet activity

```
BOCopy boCopy = (BOCopy)  
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOCopy");  
TriggeringPortBusObj_var = boCopy.copy(ToJDBCBusObj_var);  
}
```

3. After migration, the response ServiceMessageObject map in the connector module mediation flow cannot work in an inbound synchronized call. There will be a TechNote for this problem in the WebSphere Process Server V6.2 Fix Pack. To work around this problem for now:
 - a. Modify the CustomerCreate_TO_WS_CustomerCreate_TLO_Sync_Inbound_Response_SMO_Map Map in WebServicesConnector module:
 - i. In the WebSphere Integration Developer workspace, open the Java perspective by selecting **Window** → **Open Perspective** → **Other** → **Java** → **OK**.
 - ii. Open the WebServicesConnector component by right-clicking CustomerCreate_TO_WS_CustomerCreate_TLO_Sync_Inbound_Response_SMO_Map.map. Select **Open With** → **Text Editor**.
 - iii. Find the line that is shown in Example 15-15, and change it to the line that is shown in Example 15-16 on page 316.

Example 15-15 Old response BG

```
xmlns:ServiceMessageObject="smo://smo/name%3Dwsdl-primary/message%3D%257Bhttp%253A%252F%252Fwww.ibm.com%252Fwebsphere%252Fcrosswor%252F2002%252FCFGSchemas%252FWebServicesConnector%252Finterface%252Fconnector%252Fsync_async%257DWS_CustomerCreate_TLOBG/xpath%3D%252F/namespace%3DB0Map/smo.xsd"
```

Example 15-16 New response BG

```
xmlns:ServiceMessageObject="smo://smo/name%3Dwsdl-primary/message%3D%257Bhttp%253A%252F%252Fwww.ibm.com%252Fwebsphere%252Fcrossworl%252F2002%252FCFGSchemas%252FWebServicesConnector%252Finterface%252Fconnector%252Fsync_async%257DWS_CustomerCreate_TLOBG_Response/xpath%3D%252F/namespace%3DB0Map/smo.xsd"
```

- iv. Find the line that is shown in Example 15-17, and change it to the line that is shown in Example 15-18.

Example 15-17 Old response BG verb

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerCreate_TLOBG/verb" />
```

Example 15-18 New Response BG verb

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerCreate_TLOBGResponse/Response/verb" />
```

- v. Find the line that is shown in Example 15-19, and change it to the line that is shown in Example 15-20.

Example 15-19 Old response BO

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerCreate_TLOBG/WS_CustomerCreate_TLO"  
variableName="WS_CustomerCreate_TLO0output0" />
```

Example 15-20 New response BO

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerCreate_TLOBGResponse/Response/WS_Custome%257Bhttp%253A%252F%252Fwww.ibm.com%252Fwebsphere%252Fcrossworl%252F2002%252FCFGSchemas%252FWebServicesConnector%252Finterface%252Fconnector%252Fsync_async%257DWS_CustomerCreate_TLOBG_Response/xpath%3D%252F/namespace%3DB0Map/smo.xsd"rCreate_TLO" variableName="WS_CustomerCreate_TLO0output0" />
```

- b. Modify the CustomerRetrieve_TO_WS_CustomerRetrieve_TLO_Sync_Inbound_Response_SMO_Map Map in the WebServicesConnector module:
 - i. In the WebSphere Integration Developer workspace, open the Java perspective by selecting **Window** → **Open Perspective** → **Other** → **Java** → **OK**.

- ii. Open the WebServicesConnector component, right-click CustomerRetrieve_TO_WS_CustomerRetrieve_TLO_Sync_Inbound_Response_SMO_Map.map, and select **Open With** → **Text Editor**.
- iii. Find the line that is shown in Example 15-21, and change it to the line that is shown in Example 15-22.

Example 15-21 Old response BG

```
xmlns:ServiceMessageObject="smo://smo/name%3Dwsdl-primary/message%3D%257Bhttp%253A%252F%252Fwww.ibm.com%252Fwebsphere%252Fcrossworl%252F2002%252FCFGSchemas%252FWebServicesConnector%252Finterfac%252Fconnector%252Fsync_async%257DWS_CustomerRetrieve_TLOBG/xpath%3D%252F/namespace%3DBOMap/smo.xsd"
```

Example 15-22 New response BG

```
xmlns:ServiceMessageObject="smo://smo/name%3Dwsdl-primary/message%3D%257Bhttp%253A%252F%252Fwww.ibm.com%252Fwebsphere%252Fcrossworl%252F2002%252FCFGSchemas%252FWebServicesConnector%252Finterfac%252Fconnector%252Fsync_async%257DWS_CustomerRetrieve_TLOBG_Response/xpath%3D%252F/namespace%3DBOMap/smo.xsd"
```

- iv. Find the line that is shown in Example 15-23, and change it to the line that is shown in Example 15-24.

Example 15-23 Old response BG verb

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerRetrieve_TLOBG/verb" />
```

Example 15-24 New Response BG verb

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerRetrieve_TLOBGResponse/Response/verb"  
/>
```

- v. Find the line that is shown in Example 15-25, and change it to the line that is shown in Example 15-26 on page 318.

Example 15-25 Old response BO

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerRetrieve_TLOBG/WS_CustomerRetrieve_TLO"
```

```
variableName="WS_CustomerRetrieve_TL00Output0" />
```

Example 15-26 New response BO

```
<map:output  
businessObjectVariableRef="ServiceMessageObject_output"  
property="body/WS_CustomerRetrieve_TLOBGResponse/Response/WS_CustomerRetrieve_TLO" variableName="WS_CustomerRetrieve_TL00Output0"  
>
```

4. When migrating WebServices connector, we migrate it to an HTTP binding with WebSphere InterChange Server data handler. However, this data handler cannot work well with WebSphere Process Server. We create a new WebSphere Process Server data handler as a work-around:
 - a. In the WebSphere Integration Developer workspace, open the Java perspective by selecting **Window** → **Open Perspective** → **Other** → **Java** → **OK**.
 - b. Right-click the **WebServicesConnector** module, select **New** → **Package**, and enter the package name: `com.ibm.redbook.dh`. Click **Finish**.
 - c. Copy `WebServiceExportCreateCustomerDataHandler.java` and `WebServiceExportRetrieveCustomerDataHandler.java` to `com.ibm.redbook.dh` package.
 - d. Save the changes by selecting **File** → **Save All**.
 - e. Build the workspace by selecting **Project** → **Clean** → **OK**.
 - f. Open the Business Integration perspective by selecting **Window** → **Open Perspective** → **Business Integration**.
 - g. Open the Assembly Diagram of `WebServicesConnector` module. Select the **WebServicesConnector_SOAPHTTPListener1** export.
 - h. In the Properties view, select the **Method bindings** tab. Select the **WS_CustomerCreate_TLOBG_CreateCustomerCollab** bound method. Click the **Data Serialization** tab, click the **Select** for Output data format, and click **Select your custom data format transformation from the workspace**. Click **Select**, select **WebServiceExportCreateCustomerDataHandler**, and click **OK**. Then, click **Finish**.

- i. In the Properties view, select the **Method bindings** tab, select the **WS_CustomerRetrieve_TLOBG_RetrieveCustomerCollab** bound method. Click the **Data Serialization** tab, click **Select** for Output data format, and click **Select your custom data format transformation from the workspace**. Click **Select**, select **WebserviceExportRetrieveCustomerDataHandler**, and click **OK**. Then, click **Finish**.
- j. Save the changes by selecting **File** → **Save All**.
- k. Build the workspace by selecting **Project** → **Clean** → **OK**.

15.2.3 Deployment

After migrating the data access scenario from WebSphere InterChange Server to WebSphere Integration Developer, the integration system is ready to be deployed in WebSphere Process Server.

To deploy the components:

1. From WebSphere Integration Developer, from the Servers tab, click the embedded instance of **WebSphere Process Server**, and select **Start** as shown in Figure 15-17 on page 320.

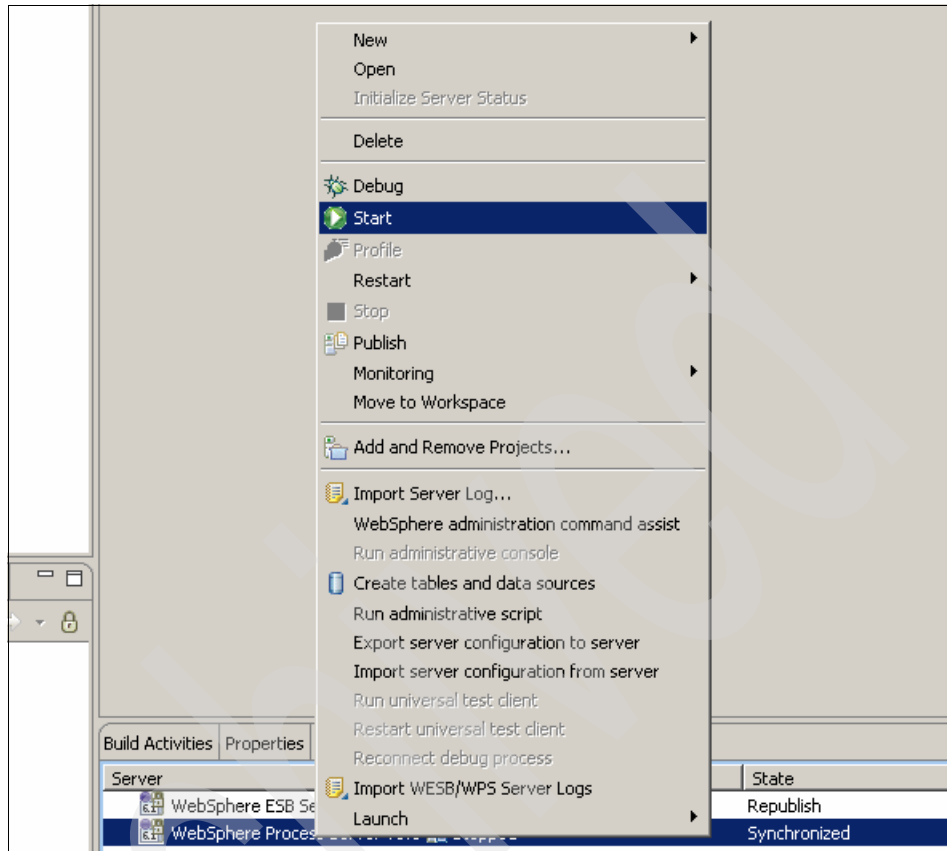


Figure 15-17 Starting WebSphere Process Server

2. Disable the default security settings in WebSphere Process Server to simplify the deployment and testing of the data access scenario:
 - a. Click the embedded instance of **WebSphere Process Server**, and select **Run administrative console** to launch the WebSphere Process Server administrative console (Figure 15-18 on page 321).

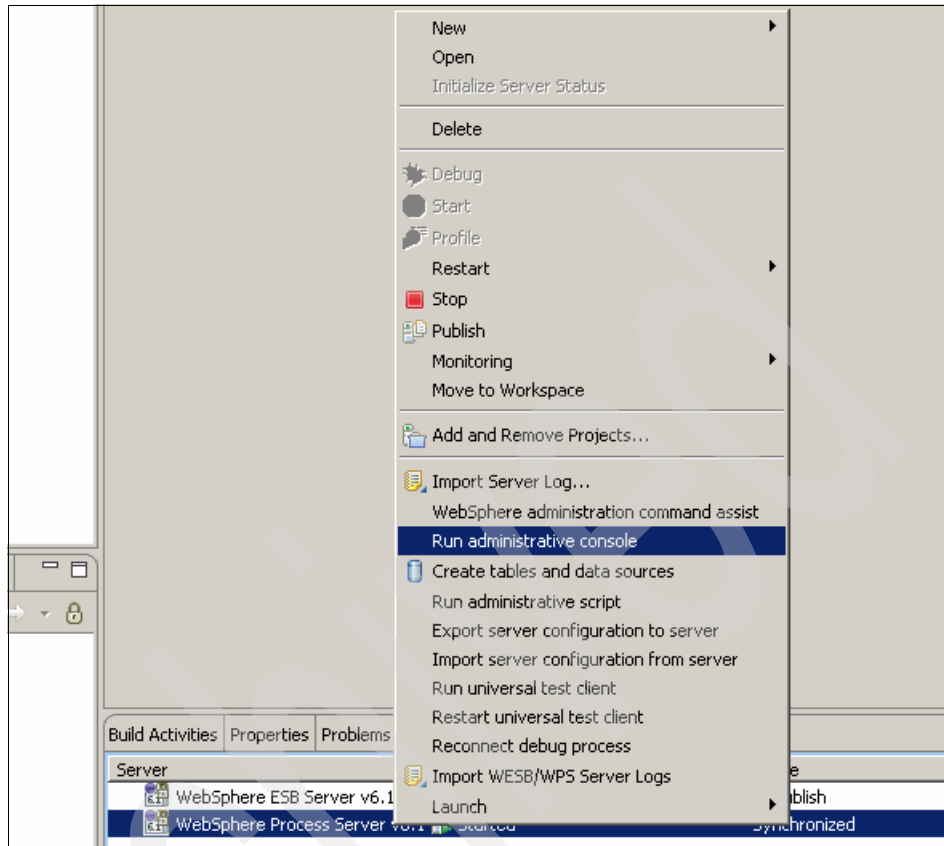


Figure 15-18 Running the administrative console

- b. Log in by using the default administrative user ID and password.

- c. In the Administrative Console:
 - i. Under **Security**, choose **Secure administration, applications, and infrastructure**.
 - ii. In the right pane, clear the **Enable administrative security**, **Enable application security**, and **Use Java 2 security to restrict application access to local resources** check boxes (Figure 15-19).
 - iii. Click **Apply** and save the changes.

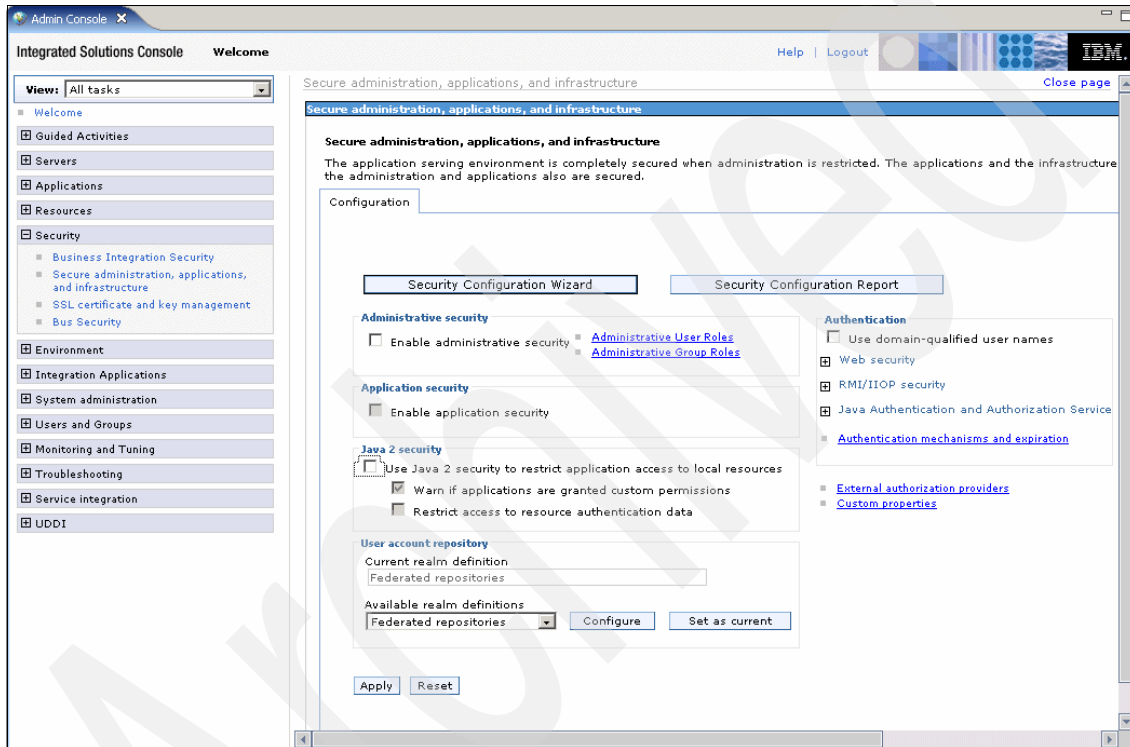


Figure 15-19 Disabling administrative security

- d. Under **Security** in the left navigation panel, choose **Bus Security**. In the right pane, click each Bus name and disable security for each bus, one by one. Figure 15-20 shows an example of the results. Click **Apply** and save the changes.

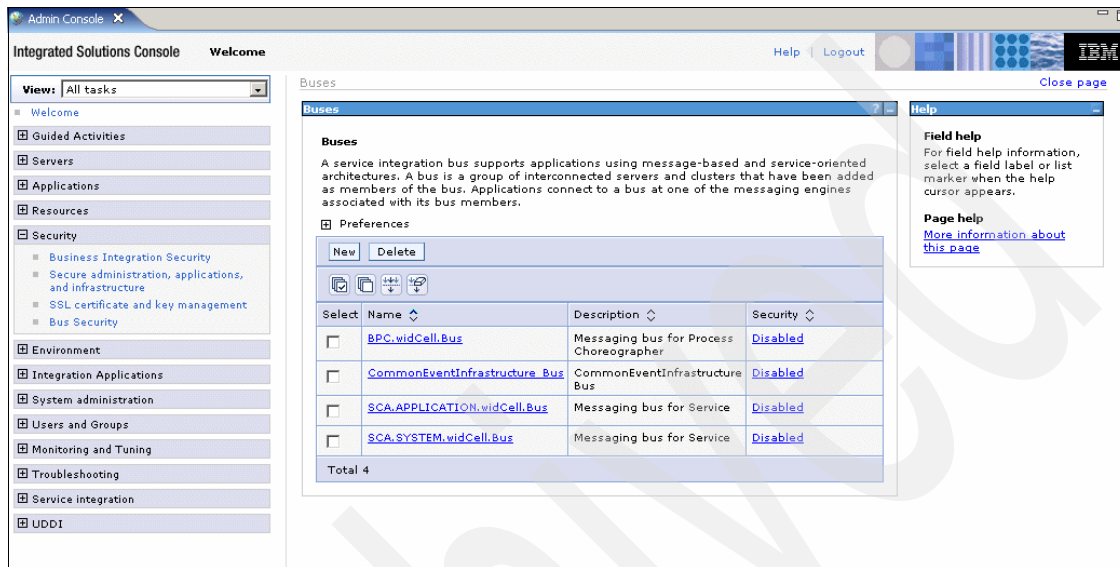


Figure 15-20 Disabling bus security

- e. Under Buses, click **SCA.APPLICATION.widCell.Bus SI Bus**.
- f. Under General Properties (Figure 15-21), for Permitted transports, select **Allow the use of all defined transport channel chains**. Click **Apply** and save the changes.

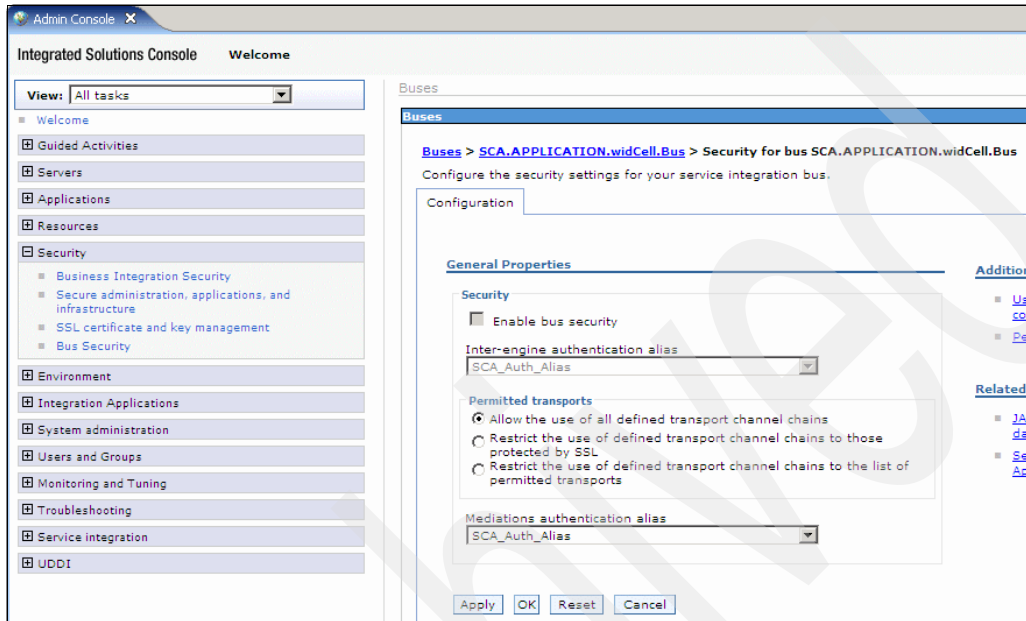


Figure 15-21 Allowing permitted transports for bus security

- g. Select **Application Servers** → **server1** → **Business Integration** → **Business Process Choreographer** → **Business Flow Manager**. Clear the **Enable Common Event Infrastructure logging** check box. Click **Apply** and save the changes.
- h. Log out from the WebSphere Process Server administrative console and close the console window.

- i. Open the profile configuration for WebSphere Process Server. In WebSphere Integration Developer, from the **Servers** tab, select the appropriate **WebSphere Process Server** instance, and choose **Open** (Figure 15-22).

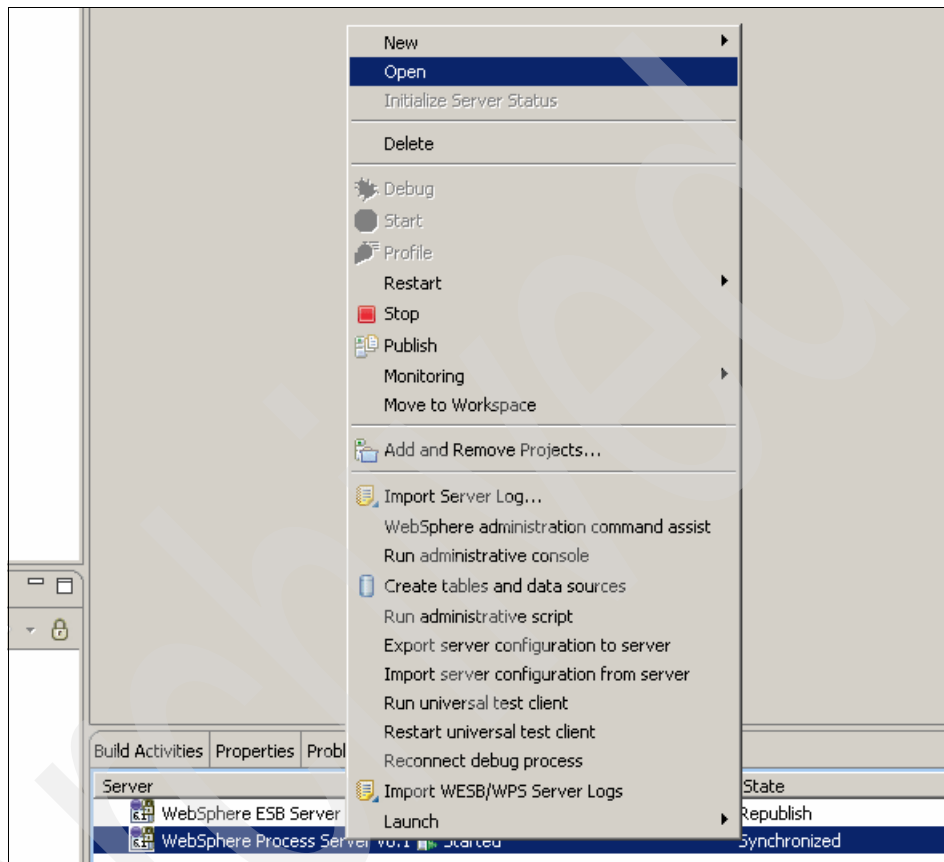


Figure 15-22 Opening the profile configuration

- j. In the Security section (Figure 15-23), clear the **Security is enabled on this server** check box to disable the security settings in the profile configuration. Close the profile configuration window and save the settings.

The screenshot shows the 'Overview' page of the WebSphere Process Server v6.2 configuration console. The 'Server' section is expanded, showing settings for the 'wps' profile. The 'Security' section is also expanded, showing the 'Security is enabled on this server' checkbox, which is currently checked. The 'Current active authentication settings' section shows the 'User ID' as 'admin' and the 'Password' as '*****'. The 'Automatically trust server certificate during SSL handshake' checkbox is also checked. The 'Test Connection' button is visible below the authentication settings.

General Information
Specify the host name and other common settings.

Server name: WebSphere Process Server v6.2 at localhost
Host name: localhost
Runtime Environment: WebSphere Process Server v6.2

Server
Enter settings for the server.

WebSphere profile name: wps
WebSphere server name: server1
Update server status interval (in milliseconds): 5000

Server connection types and administrative ports

☐ Automatically determine connection settings
☒ Manually provide connection settings

Connection Type	Port	Default port	Description
<input type="checkbox"/> RMI	2811	2809	Designed to improve
<input checked="" type="checkbox"/> SOAP	8882	8880	Designed to be more

[Test Connection](#)

☒ Enable universal test client
☒ Optimize server for testing and developing
☐ Terminate server on workbench shutdown

Publishing
Timeouts
Publishing settings for WebSphere Application Server
Modify the publishing settings.

☐ Run server with resources within the workspace
☒ Run server with resources on Server
☐ Minimize application files copied to the server

Security
Enable and setup security.

☒ Security is enabled on this server

Current active authentication settings:

User ID: admin
Password: *****
☒ Automatically trust server certificate during SSL handshake
[Test Connection](#)

Figure 15-23 Disabling profile security

- k. Restart the embedded instance of WebSphere Process Server. From the **Servers** tab (Figure 15-24), select the instance, right-click, and select **Restart** → **Start**.

Error message: If you see a startup error message in the console window, stop and start the server instead of using Restart.

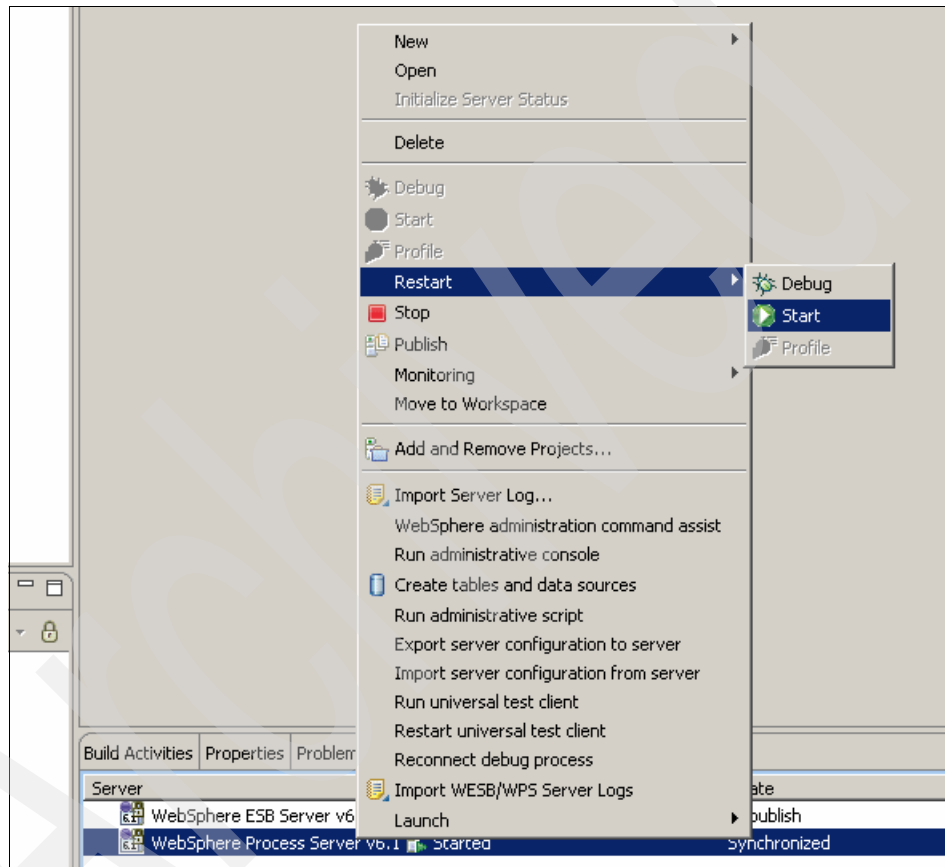


Figure 15-24 Restarting the WebSphere Process Server

3. Deploy the projects in the workspace. In WebSphere Integration Developer (Figure 15-25), from the **Servers** tab, select the appropriate **WebSphere Process Server** instance, and select **Add and Remove Projects**.

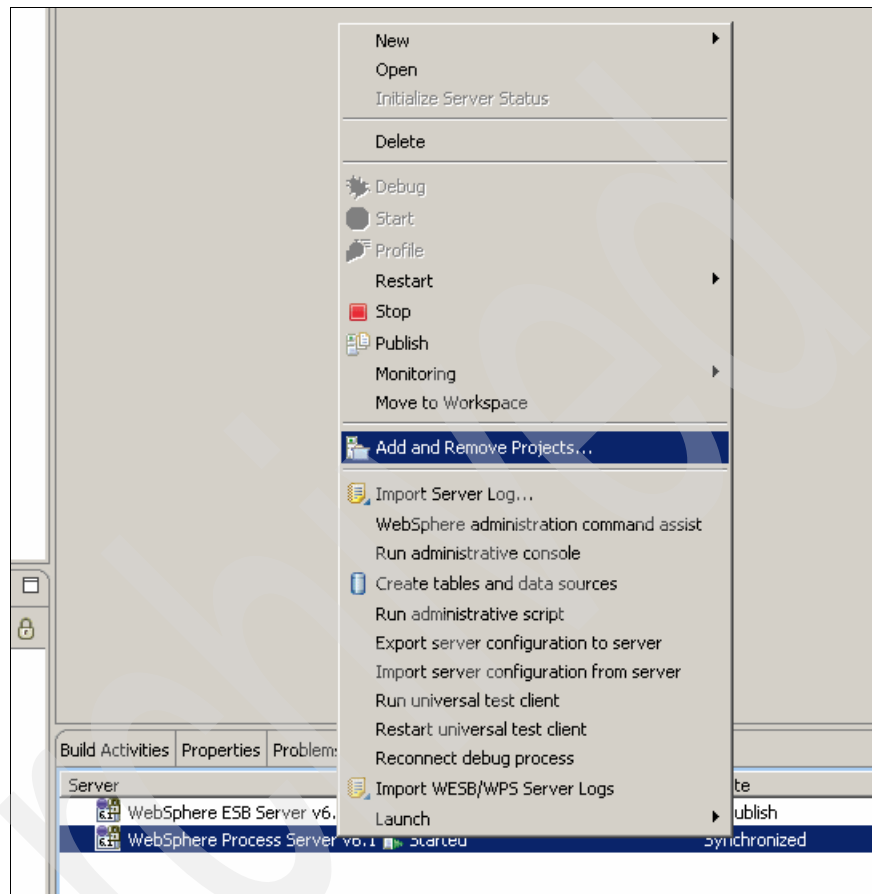


Figure 15-25 Adding projects to WebSphere Process Server

4. In the Add and Remove Projects window (Figure 15-26), click **Add All >>** to add all the projects in the workspace to the server, and then click **Finish** to initiate the deployment.

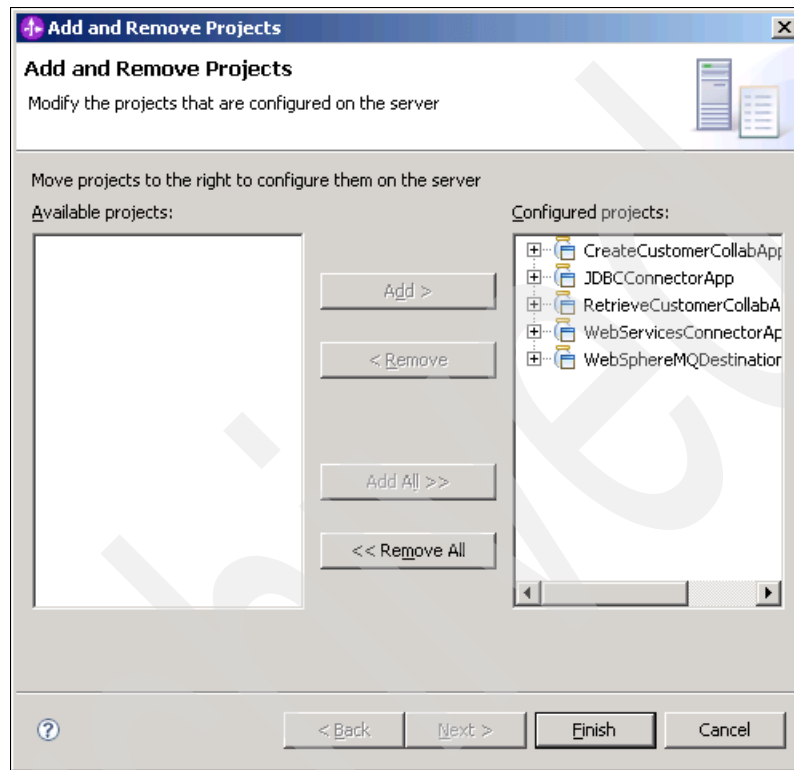


Figure 15-26 Deploying to WebSphere Process Server

5. Verify that all modules are deployed and have started correctly in WebSphere Process Server. Expand the embedded WebSphere Process Server instance to show all the deployed modules. After the deployment has finished, the modules show a status of *Started*.

When applications are not displayed: Sometimes the applications are not displayed as started on the Servers tab although they are started. If this situation happens, verify the status of the applications in the administrative console. Select **Applications** → **Enterprise Applications**, and check the application status. If any of the newly deployed applications are not started, start them from the Enterprise Applications window.

6. After installing the JDBCConnector application to WebSphere Process Server, configure the custom properties of the application that the database requires for the resource providers and resource factories:
 - a. Launch the WebSphere Process Server administrative console, enter the User ID, and click **Log In**.
 - b. Click **Application** → **Enterprise Applications**, and select the **JDBCConnectorApp** application.
 - c. Click **Manage Module** → **IBM WebSphere Adapter for JDBC** → **Resource Adapter**.
 - d. In the Class path box, enter the JDBC driver class path as shown in Example 15-27.

Example 15-27 JDBC driver classpath

C:/WPS62/universalDriver_wbi/lib/db2jcc.jar

Note: C:/WPS62 is the home directory of WebSphere Process Server.

- e. Click **J2C connection factories** → **JDBCConnector.Output_CF** → **Custom properties**.
- f. Click **XADatabaseName**, enter the value, which is ICSREPOS in our case. Click **OK** → **Save** to save the change.
- g. Click **jdbcDriverClass**, change the value to com.ibm.db2.jcc.DB2XADatasource. Click **OK** → **Save** to save the change.
- h. Click **XADatasourceName**, enter the value, which is com.ibm.db2.jcc.DB2XADatasource in our case. Click **OK** → **Save** to save the change.

15.2.4 Setting up the testing module

In the following steps, we import the WSDL files that are associated with the data access scenario. The files are imported into a library that serves the testing process and the subsequent function of replacing the WebSphere Business Integration Adapter for Web Services with the HTTP binding.

To import the WSDL files:

1. Create a new library in the workspace:
 - a. Select **File** → **New** → **Library**.
 - b. In the New Library window (Figure 15-27), for Library Name, type `WebServicesUtilityLibrary`, and click **Finish**.

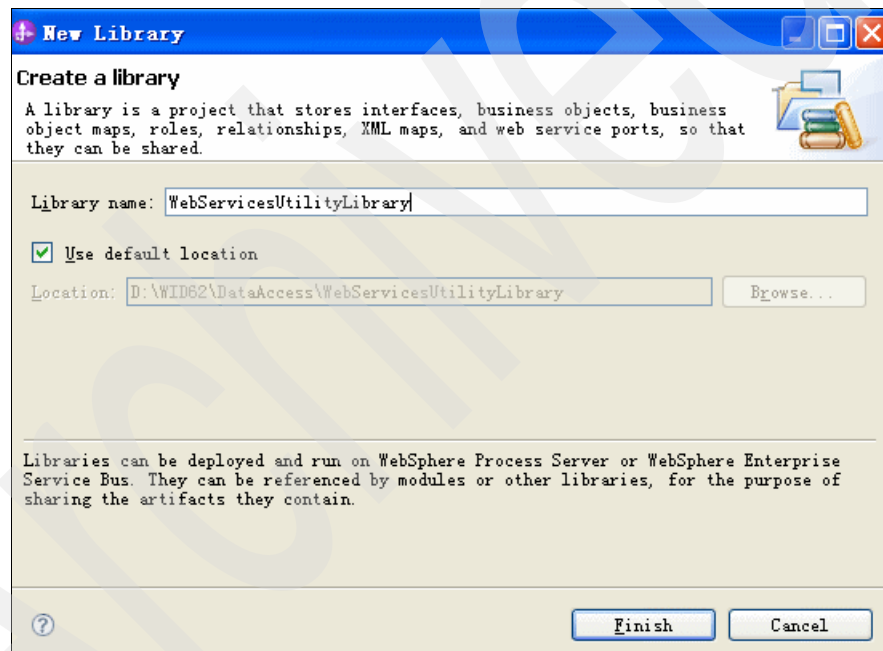


Figure 15-27 New Web services utility library

2. Import the WSDL files that were originally generated when the data access collaborations were exposed as Web services in WebSphere InterChange Server. These files can be located on the distribution media that accompanied this book (refer to Appendix D, “Additional material” on page 651). To initiate the WSDL file import:
 - a. Select **File** → **Import**.
 - b. In the Import – Select window (Figure 15-28), click **Business Integration**, select **WSDL/Interface**, and click **Next**.

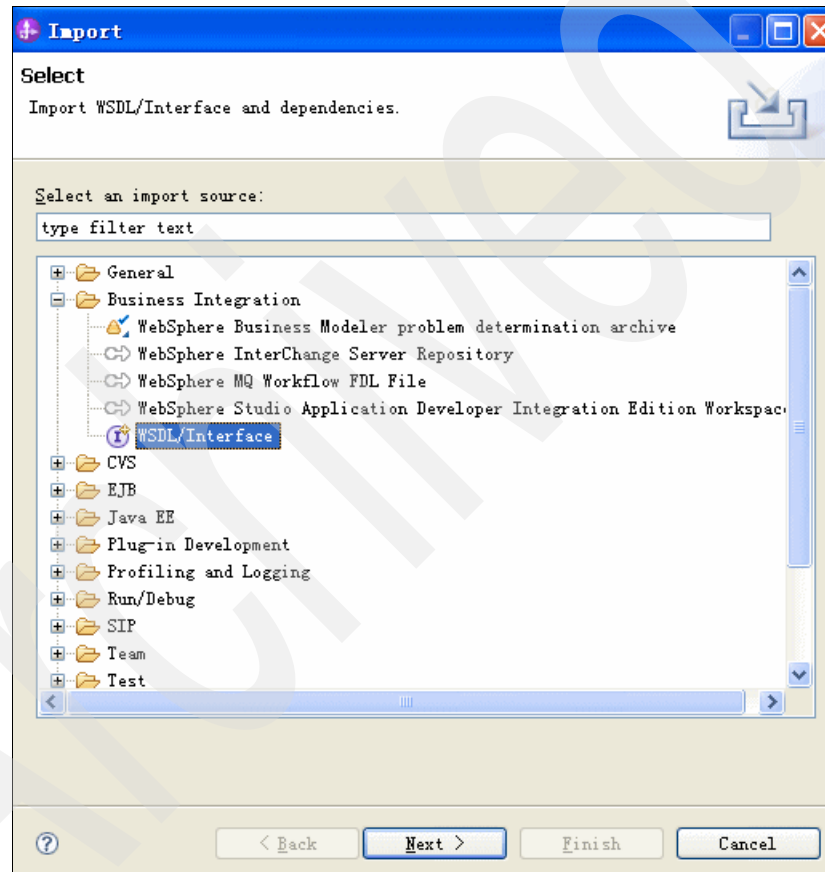


Figure 15-28 Importing the WSDL interface files

Files for download: You can download all of the files that are mentioned in this section as explained in Appendix D, “Additional material” on page 651.

3. In the Import – WSDL/Interface Import window (Figure 15-29 on page 334):
 - a. For Import from, select **c:\setuptemp\Chapter 15 Samples** (created when the included files were extracted as explained in Chapter 14, “Preparation for the technical solutions” on page 279).
 - b. Select the **WSDLFiles** folder, which automatically selects all subfolders, which are CreateCustomerCollab and RetrieveCustomerCollab. These folders contain the WSDL interface and implementation files that need to be imported.
 - c. For Into module, select **WebServicesUtilityLibrary**.
 - d. Click **Finish** to execute the import.

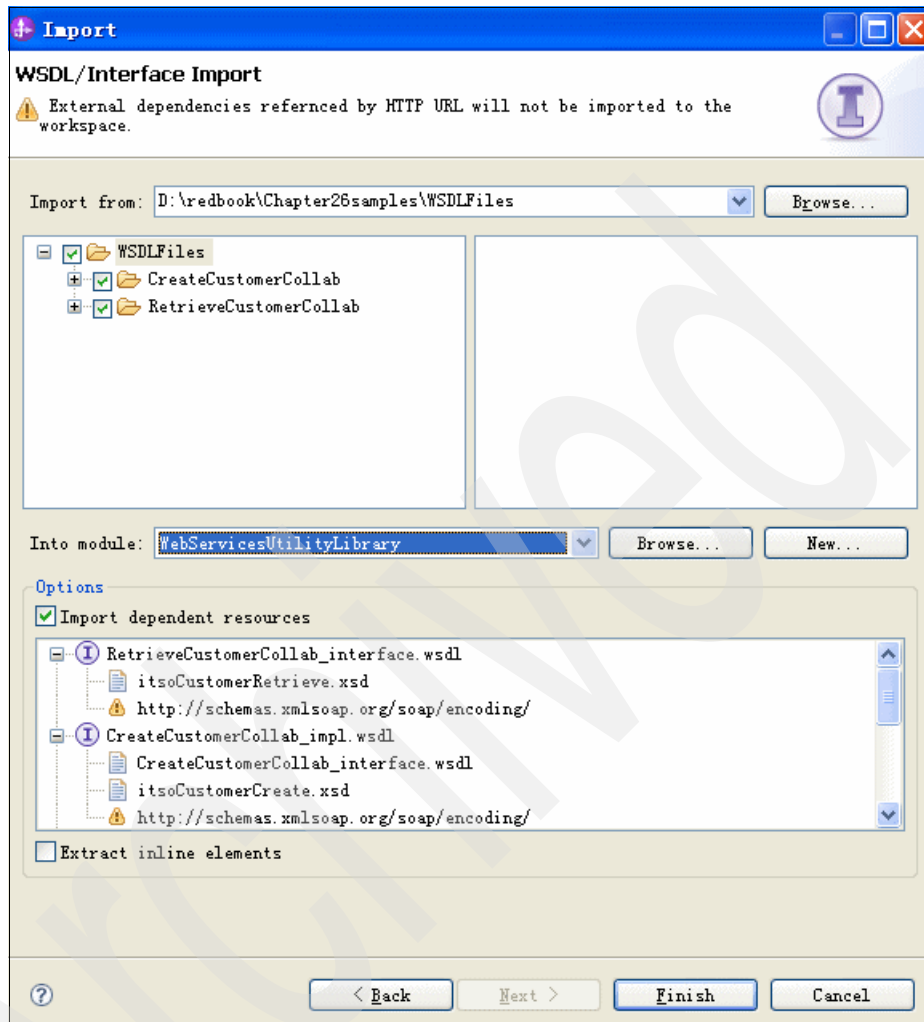


Figure 15-29 Importing the WSDL interface selection

- Verify the WSDL file import by examining the artifacts that were generated during the WSDL file import (Figure 15-30 on page 335). The imported artifacts include business objects, interfaces, and Web service ports.

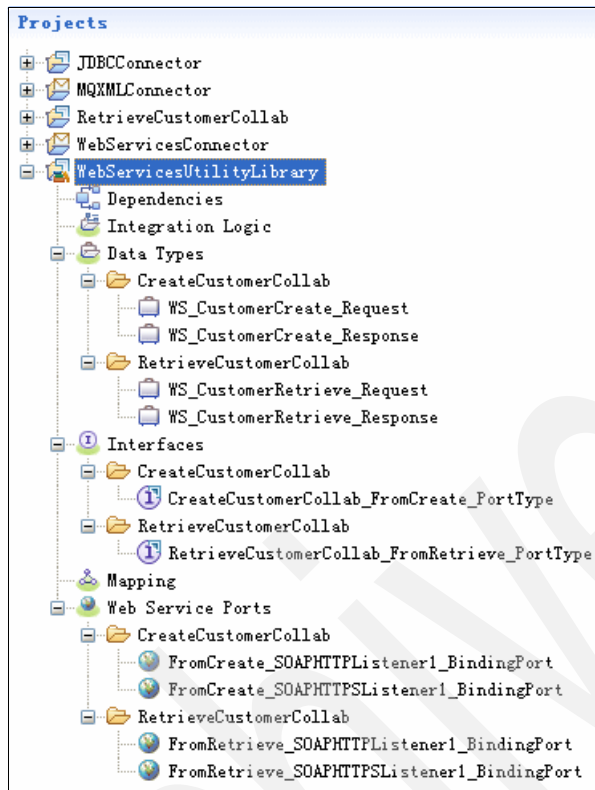


Figure 15-30 Importing the WSDL generated artifacts

15.3 Testing the end-to-end solution

In this section, we assume that you have deployed all the required applications and are running them on WebSphere Process Server.

In the following procedure, we verify the function of the data access scenario. The goal of the testing is to verify that the migration succeeded in generating a functionally compatible integration system in WebSphere Process Server. The testing verifies functional compatibility:

1. To test the migration of the data access scenario, be sure that the external database system and MQ queues are connected and running.

Setup instructions: See Chapter 14, “Preparation for the technical solutions” on page 279, for detailed instructions about setting up the external database and MQ queues for the migration scenarios.

In addition, be sure that the WebSphere Integration Developer workspace has built without error, the modules have published to the embedded WebSphere Process Server without error, and the modules have all started successfully.

2. Because we have already migrated the WebServices connector to the HTTP binding, we need to modify the client application before connecting to the HTTP binding. To perform the change:
 - a. In WebSphere Integration Developer, expand **WebServicesUtilityLibrary**, and open the **FromCreate_SOAPHTTPListener1_BindingPort** as shown in Figure 15-30 on page 335. Replace the line that is shown in Example 15-28 with the line that is shown in Example 15-29.

Example 15-28 Old WebService address

```
<soap:address location="http://localhost:8080/" />
```

Example 15-29 New WebService address

```
<soap:address  
location="http://localhost:9080/WebServicesConnectorWeb" />
```

- b. In WebSphere Integration Developer, expand **WebServicesUtilityLibrary**, and open the **FromRetrieve_SOAPHTTPListener1_BindingPort** as shown in Figure 15-30 on page 335. Replace the line that is shown in Example 15-30 with the line that is shown in Example 15-31.

Example 15-30 Old WebService address

```
<soap:address location="http://localhost:8080/" />
```

Example 15-31 New WebService address

```
<soap:address  
location="http://localhost:9080/WebServicesConnectorWeb" />
```

3. In WebSphere Integration Developer, expand **WebServicesUtilityLibrary** and expand **Web service Ports** → **CreateCustomerCollab** and **RetrieveCustomerCollab**. Under the CreateCustomerCollab folder,

right-click **FromCreate_SOAPHTTPListener1_BindingPort**, and select **Web Services** → **Test with Web Services Explorer**.

4. Maximize the Web Services Explorer frame and expand **CreateCustomerCollab** → **FromCreate_SOAPHTTPListener1_Binding**. Select the **createCustomer** operation to open the WSDL operation test form. In the Actions pane, complete the data fields as shown in Figure 15-31. Then, click **Go** to execute a test.

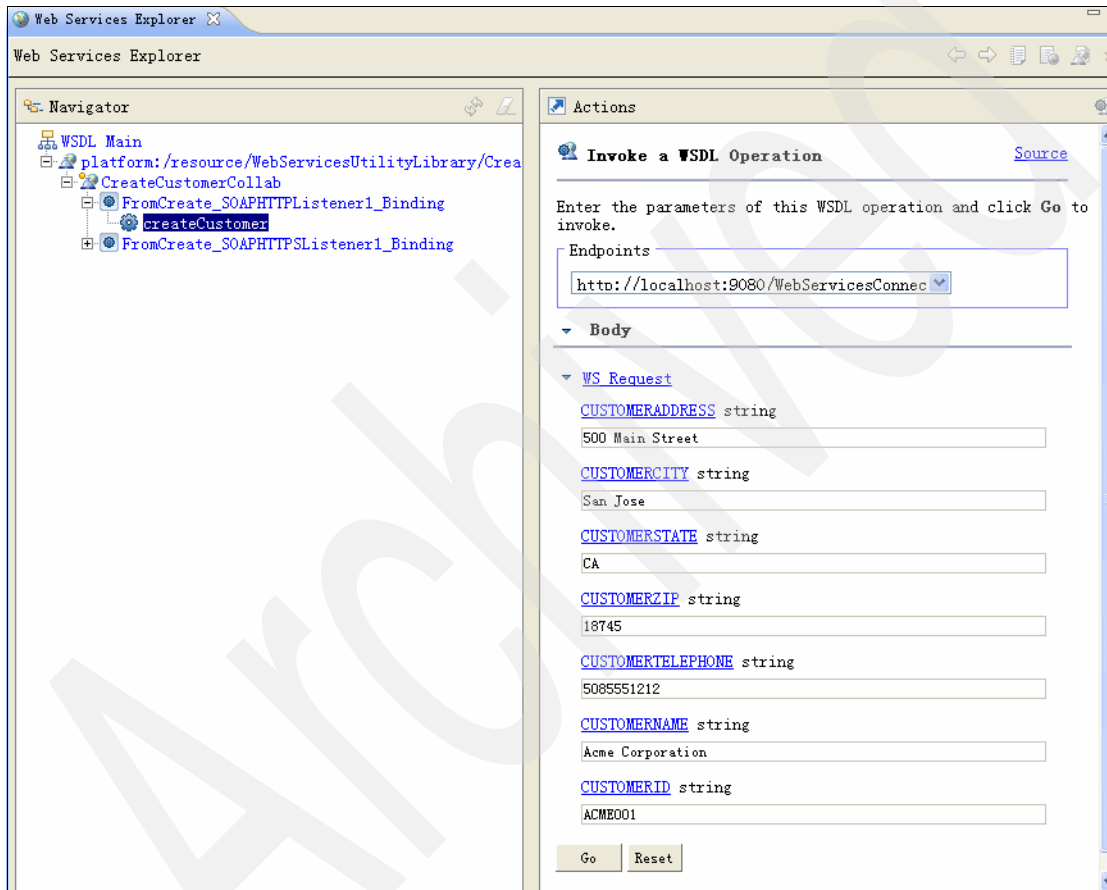


Figure 15-31 Testing createCustomer

Figure 15-32 shows a successful result.

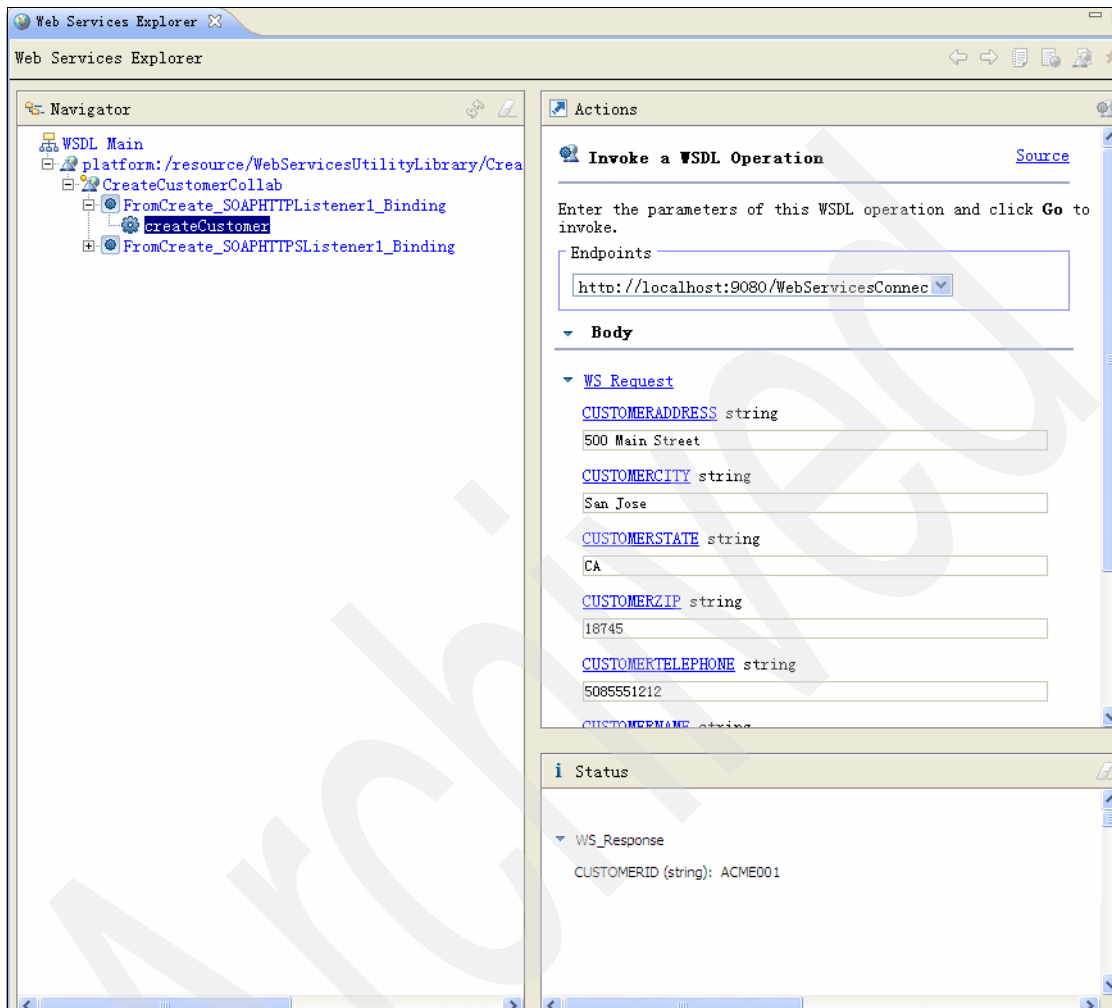


Figure 15-32 Result of testing CreateCustomer

Note: A valid CustomerID value is required to create a customer in the data access scenario. In addition, the value must be unique. Therefore, subsequent tests must use different CustomerID values.

5. In the WebSphere Integration Developer navigation frame, under the RetrieveCustomerCollab folder, right-click **FromRetrieve_SOAPHTTPListener1_BindingPort**, and select **Web Services** → **Test with Web Services Explorer**.
6. Maximize the Web Services Explorer frame, and expand **RetrieveCustomerCollab** and **FromCreate_SOAPHTTPListener1_Binding**. Select the **retrieveCustomer** operation to open the WSDL operation test form. Complete the data fields as shown in Figure 15-33. Then, click **Go** to execute a test.

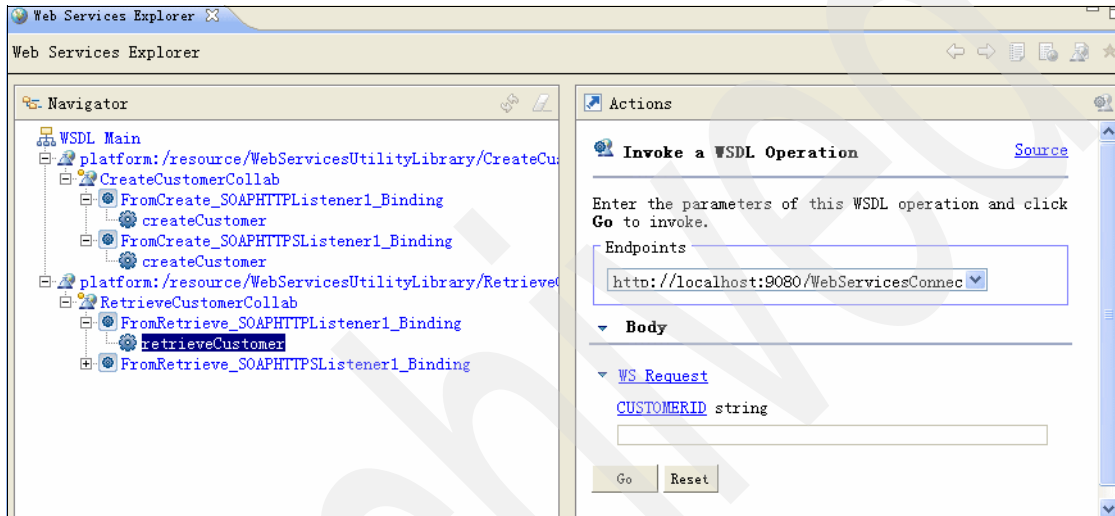


Figure 15-33 Testing RetrieveCustomer

Figure 15-34 shows a successful result.

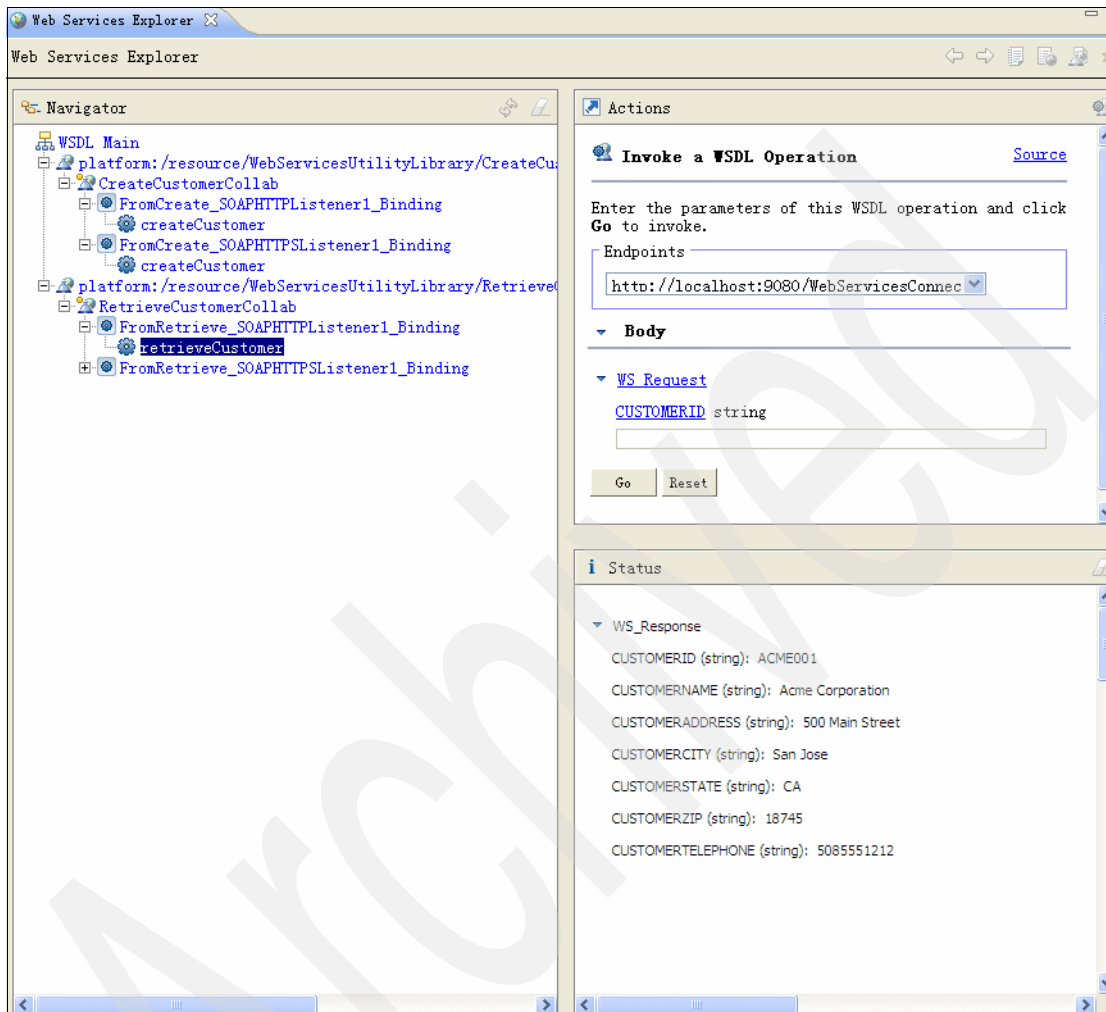


Figure 15-34 Result of testing RetrieveCustomer

Note: You might get a runtime error when you run the RetrieveCustomer. Check 19.1.3, “WebSphere Process Server log and trace” on page 561 for detailed information.

15.4 Conclusion

In this chapter, we showed a step-by-step migration of a data access solution. The migrated WebSphere InterChange Server solution involved various adapters for back-end connectivity. The migration started by using the WebSphere Integration Developer Migration Wizard to get the project working in a WebSphere Process Server environment. After this general migration, the JDBC adapter was migrated to a WebSphere Adapter for the back-end connectivity.

In addition, we performed an end-to-end test to check the functionality of the migrated components after each step.

Data synchronization scenario with technology adapters

The data synchronization migration scenario outlined in this chapter illustrates an end-to-end scenario in which data entities are synchronized between back-end systems through a typical set of WebSphere InterChange Server components. Together with WebSphere InterChange Server, these components form an integration system.

This chapter focuses on the steps that are necessary to migrate an entire data synchronization integration system that was originally designed for WebSphere InterChange Server to WebSphere Process Server. By performing the steps illustrated in this chapter, an end-to-end data synchronization scenario can be completely migrated to WebSphere Process Server, optimized, and tested.

The data synchronization scenario uses three WebSphere Business Integration Adapters: the WebSphere Business Integration Adapter for WebSphere MQ, the WebSphere Business Integration Adapter for JDBC, and the WebSphere Business Integration Adapter for JText. It also uses the WebSphere Business Integration Data Handler for XML and the WebSphere Business Integration Delimited Text Data Handler. In addition, the scenario uses business objects,

maps, relationships, and collaborations. Specifically, the collaboration in this migration scenario uses collaboration properties and multiple process scenarios.

The scenario in this chapter simulates an integration system that is designed to provide synchronization between three back-end systems:

- ▶ A Customer Relationship Management (CRM) system internally accessed through Java Database Connectivity (JDBC)
- ▶ A CustomerMaster mainframe system delivering new and updated customer information through WebSphere MQ
- ▶ An Enterprise Resource Planning (ERP) system that receives changes through the file system

Data synchronization is a common WebSphere InterChange Server pattern where two systems exchange data by using business objects, supporting the four basic verbs or actions of create, retrieve, update, and delete. In WebSphere InterChange Server, an integration specialist uses a collaboration template that synchronizes data between System A and System B. To synchronize between the CRM and ERP systems, the integration specialist creates a collaboration instance, where System A is configured as the CustomerMaster and System B is configured as the CRM system. To add a third system, an integration specialist adds a collaboration instance, where System A is configured as the CustomerMaster and System B is configured as the ERP System.

Although it seems as though two separate integration flows are represented as two collaboration instances, they behave like one integration where a single event triggers data synchronization to two systems. Bidirectionality can easily be configured by repeating the collaboration instance creation step, creating an instance that goes from System B to System A. This step is beyond the scope of this scenario, but it is mentioned for completeness.

The data synchronization scenario is completely independent of the other examples and scenarios in this book. However, many of the concepts and steps that are necessary to migrate individual artifacts are the same and are used here to illustrate the migration of a complete end-to-end scenario.

As a reminder, see Part 2, “Migration implementation concepts” on page 71, for an overall discussion about the possible options for migration implementation, and Part 3, “Migration tooling” on page 177, provides detailed information about the standard migration tools.

We include the following sections:

- ▶ 16.1, “Target environment” on page 346
- ▶ 16.2, “Implementation” on page 346
- ▶ 16.3, “Testing the end-to-end solution” on page 419

- ▶ 16.4, “Conclusion” on page 429

Archived

16.1 Target environment

In this chapter, we illustrate the migration of an end-to-end scenario from WebSphere InterChange Server to WebSphere Process Server. The WebSphere InterChange Server development was done with the product versions that are listed in Table 16-1. It was migrated step-by-step as shown in this chapter to run on the WebSphere Process Server Version 6.2.

16.1.1 Software environment

Table 16-1 lists the software products and versions that we used to demonstrate the example in this chapter.

Table 16-1 Software versions used

Software	Version
WebSphere Integration Developer	6.2
WebSphere Process Server	6.2
WebSphere MQ	6.0.2 Fix Pack 4
WebSphere InterChange Server	4.3.0 Fix Pack 5
DB2	8.1.16 (or 8.2.9)
WebSphere Business Integration Data Handler for XML	2.7.3

16.2 Implementation

In this section, we describe the process of migrating the data synchronization scenario to WebSphere Integration Developer.

16.2.1 Premigration overview

The data synchronization scenario is a functional implementation of the data synchronization pattern that is typically implemented in WebSphere InterChange Server. This scenario is restricted to the following two collaborations that are both based on the same collaboration template:

- ▶ MQDL_To_JDBC_CustomerSync
- ▶ MQDL_To_JTextXML_CustomerSync

The MQDL_To_JDBC_CustomerSync collaboration synchronizes customer data between a source application that publishes changes (create, update, or delete) to a WebSphere MQ queue and a destination application that uses a DB2 database as its data store. This collaboration has a single inbound port (From) and a single outbound port (To). The From port is bound to an instance of the WebSphere Business Integration Adapter for WebSphere MQ. The To port is bound to an instance of the WebSphere Business Integration Adapter for JDBC as illustrated in Figure 16-1.

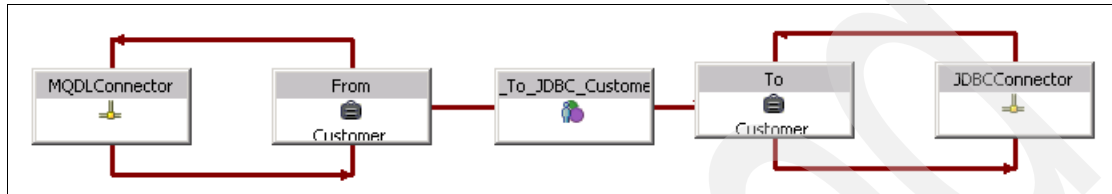


Figure 16-1 MQDL_To_JDBC_CustomerSync collaboration object

The MQDL_To_JTextXML_CustomerSync collaboration synchronizes customer data between the same source application as the MQDL_To_JDBC_CustomerSync collaboration and a destination application that consumes text files as its input. The details of the destination application are beyond the scope of this example. The example stops at the point where it delivers the text files to a preconfigured folder.

This collaboration has a single inbound port (From) and a single outbound port (To). The From port is bound to the same instance of the WebSphere Business Integration Adapter for WebSphere MQ as the MQDL_To_JDBC_CustomerSync collaboration. The To port is bound to an instance of the WebSphere Business Integration Adapter for JText as illustrated in Figure 16-2.

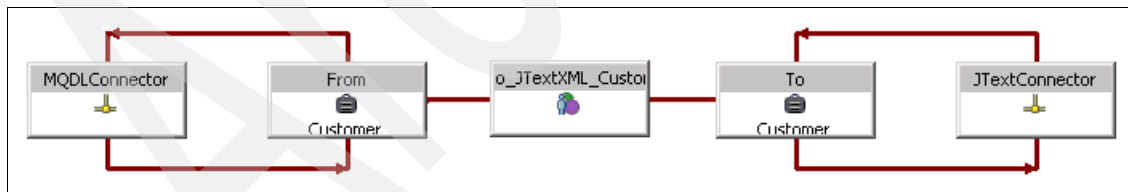


Figure 16-2 MQDL_To_JTextXML_CustomerSync collaboration object

This scenario demonstrates the commonly used publish/subscribe integration pattern. A single published event is delivered in parallel to two subscribing collaborations, which in turn, deliver it to two separate back-end systems.

16.2.2 Preparation

The goal of this chapter is to migrate an existing WebSphere InterChange Server example and test it in the WebSphere Process Server. As a prerequisite, we show how to test the DataSyncScenario example in WebSphere InterChange Server. Then, we explain the details of the migration of the example to WebSphere Process Server.

We begin by explaining the initial steps that you must complete in order to migrate the DataSyncScenario example to WebSphere Process Server.

Files for download: You can download all of the files that are mentioned in this section as explained in Appendix D, “Additional material” on page 651.

Deploy the DataSyncScenario example on WebSphere InterChange Server V4.3

To deploy the DataSyncScenario example in the WebSphere InterChange Server:

1. Create a new Interchange component library, DataSyncICL, in the WebSphere InterChange Server System Manager (Figure 16-3 on page 349). Import the ICSDataSyncScenario.jar file into it. The Java archive (JAR) file is in c:\setuptemp\Chapter16samples. Right-click the **DataSyncICL** project and select **Import from repository file**.
2. Make sure that the WebSphere MQ configuration information in the MQDLCconnector definition is consistent with the WebSphere MQ that you use.
3. Make sure that the database configuration information in the JDBCConnector and Relationship definition is consistent with the database that you use.

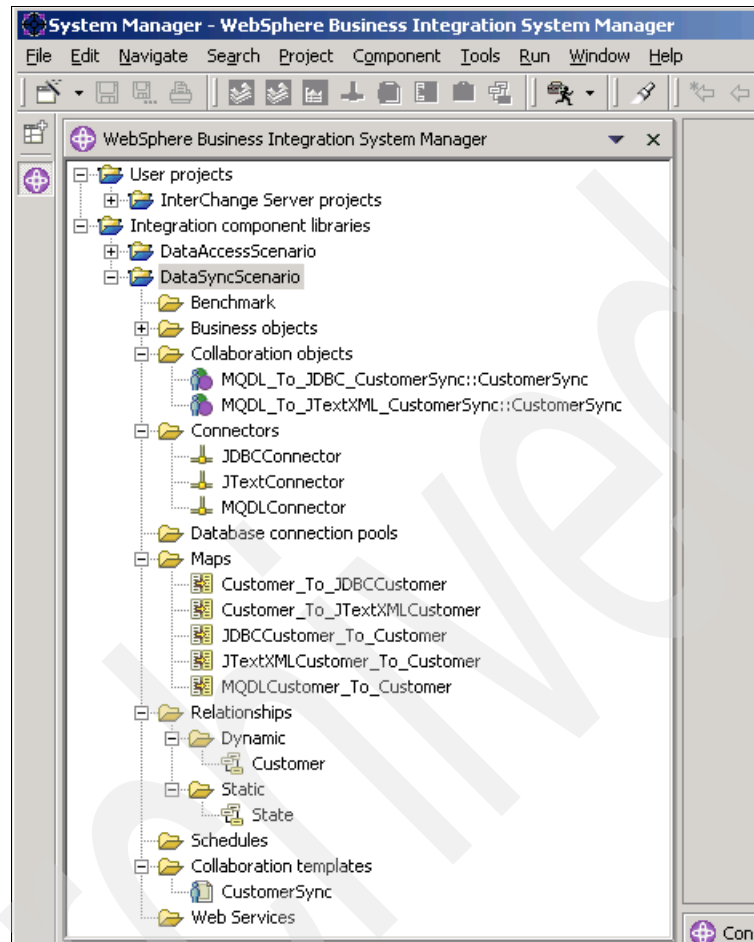


Figure 16-3 DataSyncScenario project as viewed in System Manager

4. Create a user project, called *DataSyncScenario*, in the System Manager:
 - a. Add all the components from the newly created DataSyncICL project by selecting the **DataSyncICL** check box.
 - b. Connect to the WebSphere InterChange Server instance.
 - c. Validate the project.
 - d. Right-click **DataSyncScenario** and select **Deploy user project**.
 - e. In the Deploy wizard window, expand the **DataSyncScenario** project and select **Business Objects** and **Relationships**. Select **Create schema**.

5. Confirm that the Relationships Customer and State are deployed successfully onto the WebSphere InterChange Server instance. Expand the **InterChange Server instances** → **Relationships** → **Dynamic** → **Customer**. Under Relationships, also expand **Static** → **State**. The relationships have a green arrow next to them, which confirms that the relationships deployed successfully (Figure 16-4).

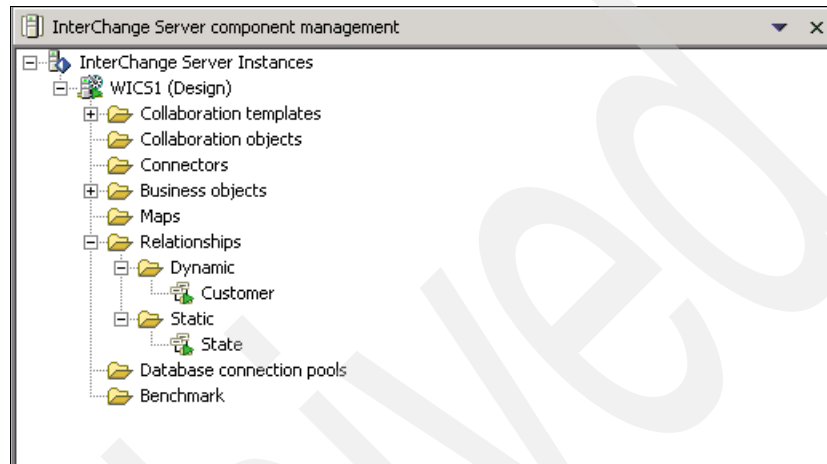


Figure 16-4 Relationships successfully deployed to WebSphere InterChange Server

6. Open DB2 control center, right-click the **ICSREPOS** database, and select **Query**.
7. Insert the static Relationship instances by executing the SQL scripts for the State Static Relationship:
 - a. Click the Script's **Open** button.
 - b. Browse to select the **C:\setuptemp\Chapter16samples\STATE_STATE_CD_T_InsertScript.sql** file.
 - c. Click the green arrow button to execute the script.
 - d. Repeat the previous three steps to execute the **STATE_STATENM_T_InsertScript.sql** script.
 - e. If you see a DB2 message to replace the current input with the contents, click **Yes**, and execute.

16.2.3 Migration

In this section, we explain how to migrate the data synchronization scenario to WebSphere Integration Developer. The `ICSDataSyncScenario.jar` file, which you extract to the `c:\setuptemp` directory in Chapter 14, “Preparation for the technical solutions” on page 279, is the WebSphere InterChange Server Repository file.

Exporting a WebSphere InterChange Server Repository file: From the WebSphere Business Integration System Manager, select an appropriate integration component library, right-click, and select **Export as** → **Repository File**.

To migrate the data synchronization scenario to WebSphere Integration Developer:

1. Launch WebSphere Integration Developer with a new workspace called *DataSyncScenarioWorkspace*:
 - a. In WebSphere Integration Developer, switch to the **Business Integration** perspective.
 - b. Select **File** → **New** → **Library** to create a new library.
 - c. In the New Library window (Figure 16-5 on page 352), for Library Name, type `DataSyncScenarioLibrary` to create a new library in the workspace. Click **Finish**.

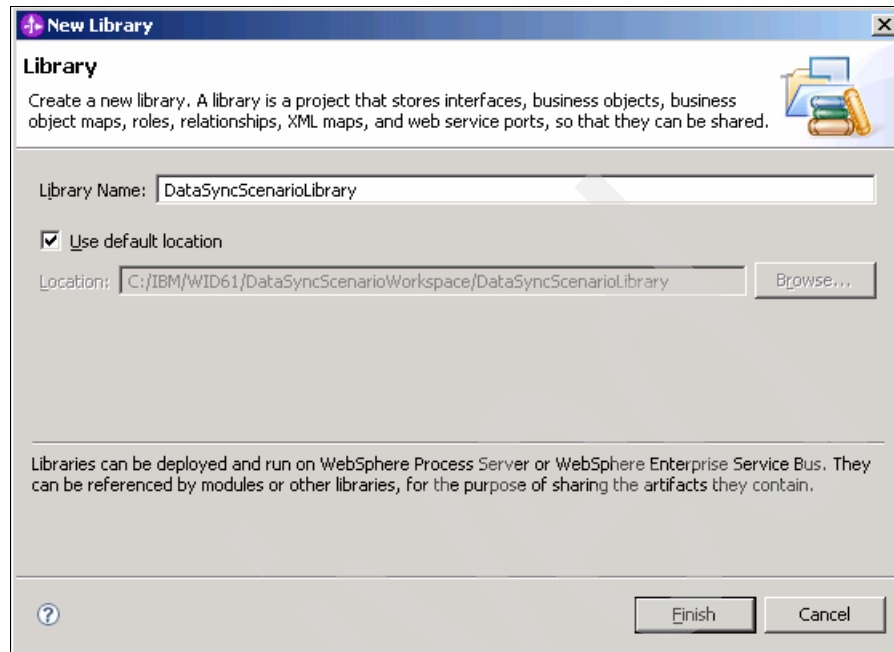


Figure 16-5 New data synchronization scenario library

2. Import the WebSphere InterChange ServerRepository file named ICSDDataSyncScenario.jar by using the WebSphere InterChange Server Migration Wizard in WebSphere Integration Developer:
 - a. Select **File** → **Import**.
 - b. In the Import: Select window (Figure 16-6) under the Business Integration folder, select **WebSphere InterChange Server JAR File**. Click **Next**.

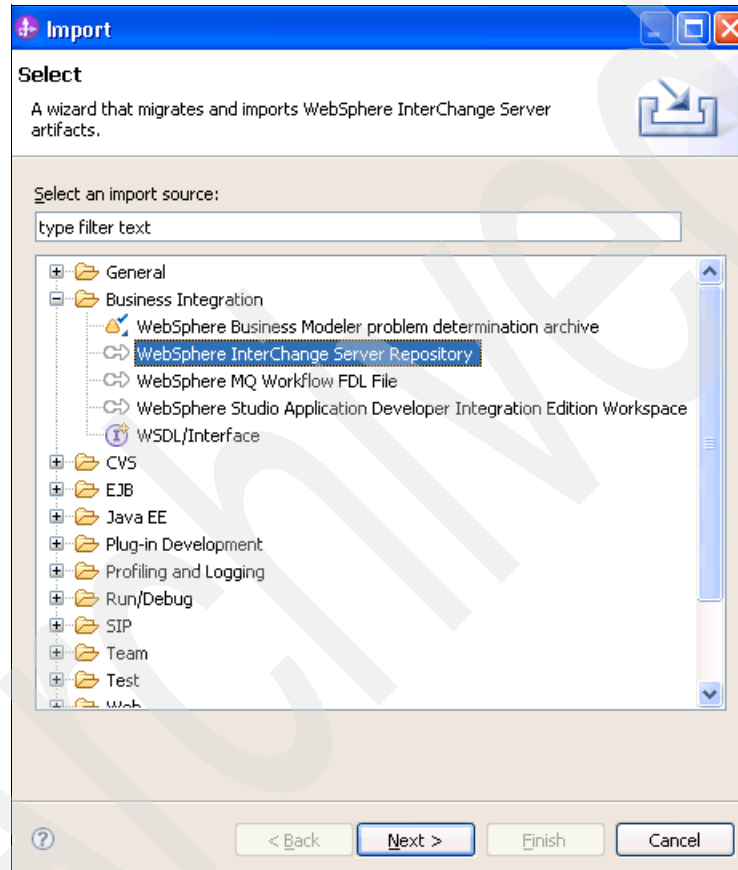


Figure 16-6 Import wizard: WebSphere InterChange Server JAR File

- c. In the Import Wizard Repository Details window (Figure 16-7 on page 354), for repository path selection, select the **ICSDDataSyncScenario.jar** file, and for Library, select **DataSyncScenarioLibrary**. Click **Next**.

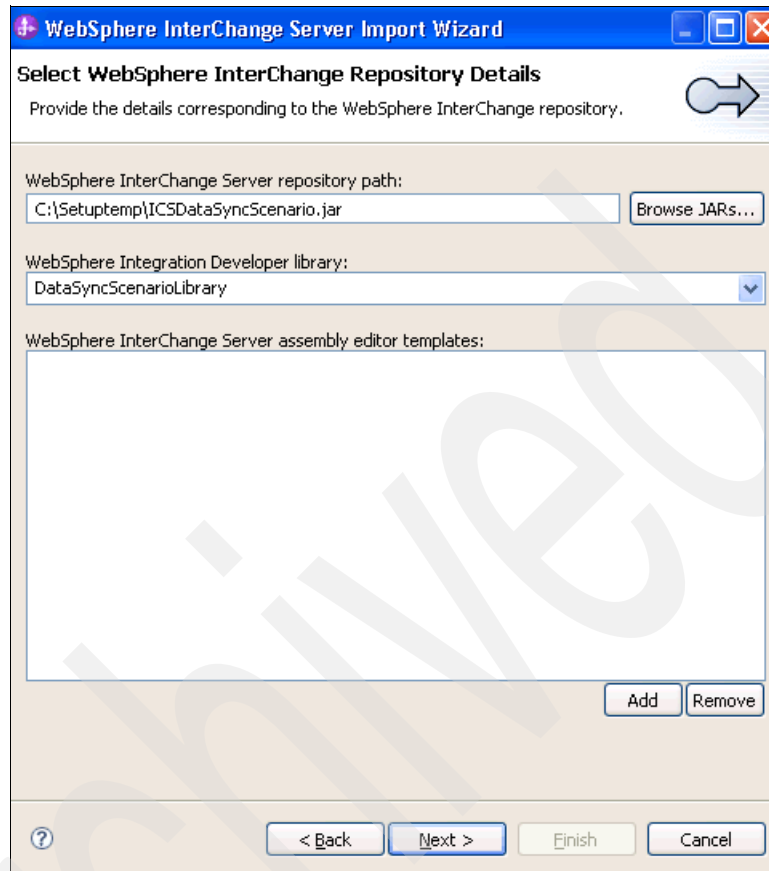


Figure 16-7 Import Wizard: Select WebSphere Interchange Repository Details

- d. In the Import Wizard: Configure Connector Migration window, select **JTextConnector** in the left panel. In the Binding option, choose **JMS to JText (FlatFile) WBI Adapter** in the drop-down list (Figure 16-8 on page 355).

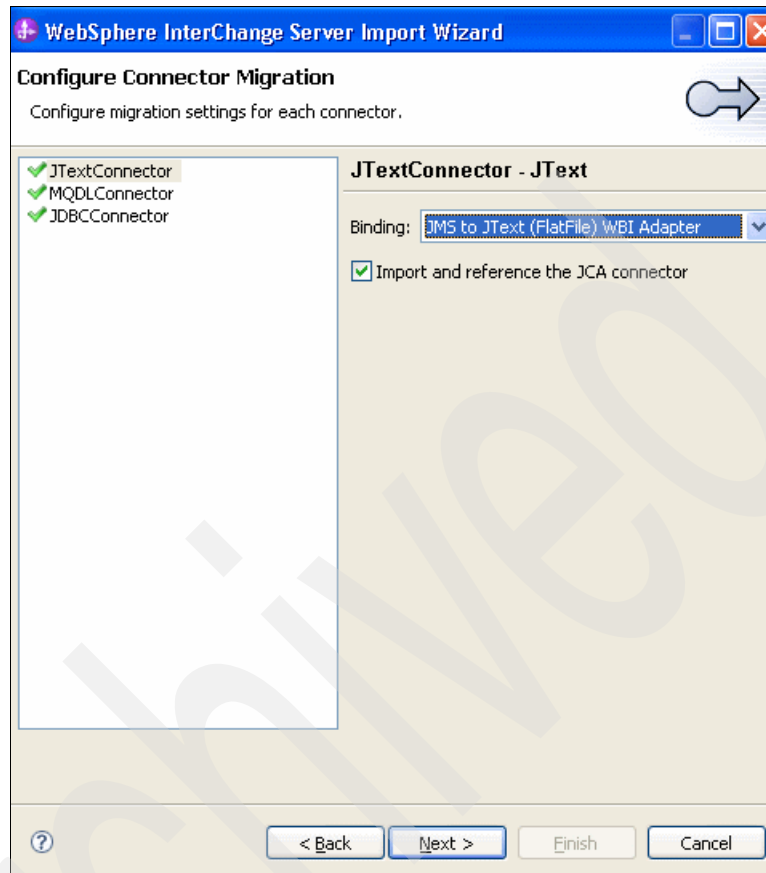


Figure 16-8 Import Wizard: Configure Connector Wizard JTextConnector

- e. In the same window, select **MQDLConnector** in the left panel. In the Binding option, choose **MQ Binding** from the drop-down list (Figure 16-9 on page 356).

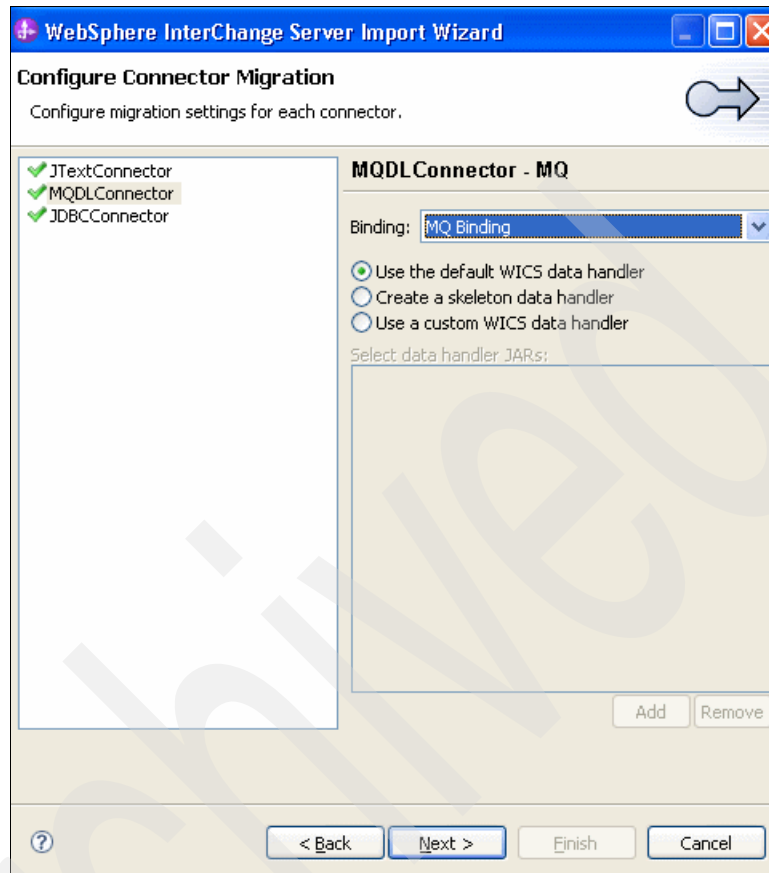


Figure 16-9 Import Wizard: Configure Connector Wizard MQDLConnector

- f. Also, in the same window, select **JDBCConnector** in the left panel. In the Binding option, choose **JMS to JDBC WBI Adapter** from the drop-down list (Figure 16-10 on page 357), and click **Next**.

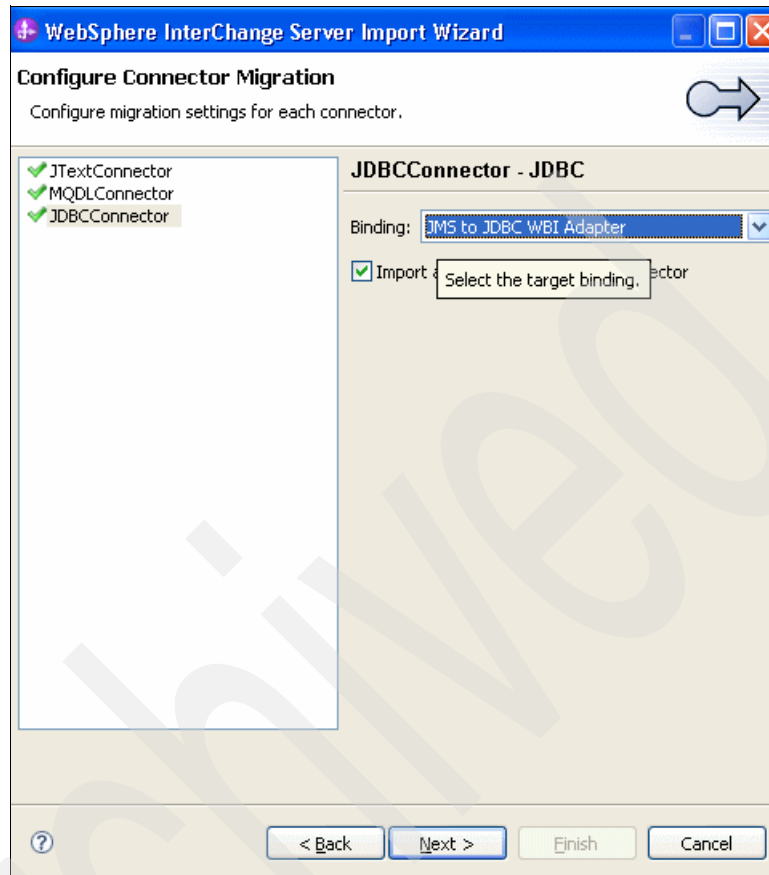


Figure 16-10 Import Wizard: Configure Connector Migration JDBCConnector

- g. In the Import Wizard: Conversion Options windows, select the options as shown in Figure 16-11 on page 358, and click **Next**.

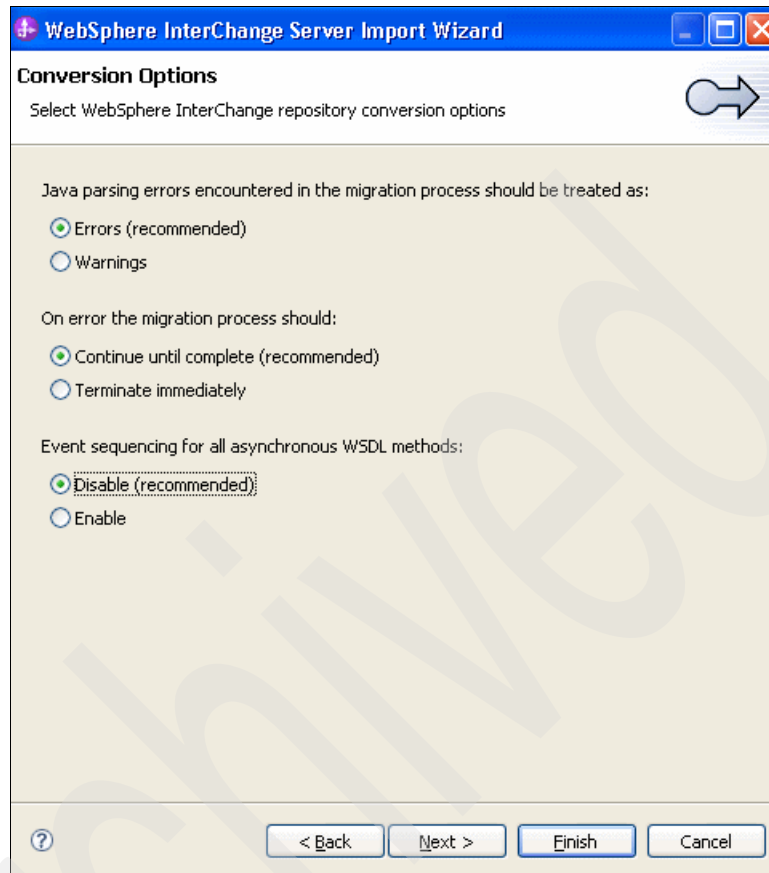


Figure 16-11 Import Wizard: Conversion Options

- h. In the Import Wizard: Migration Summary Window, confirm that all information is correct (Figure 16-12 on page 359), and click **Finish**.

WebSphere InterChange Server Import Wizard

Migration Summary

Summary of the WebSphere InterChange repository migration.

Repository Path:	C:\Setuptemp\WCSDDataSyncScenario.jar
Library:	DataSyncScenarioLibrary [new]

Java parsing errors encountered in the migration process should be treated as:	Errors (recommended)
On error the migration process should:	Continue until complete (recommended)
Event sequencing for all asynchronous WSDL methods:	Disable (recommended)

JTextConnector

Connector Type:	JText
Target Binding:	JMS to JText (FlatFile) WBI Adapter
Import JCA Connector:	true

MQDLConnector

Connector Type:	MQ
Target Binding:	MQ Binding
Data Handler Type:	Default Data Handler

JDBCConnector

Connector Type:	JDBC
Target Binding:	JMS to JDBC WBI Adapter
Import JCA Connector:	true

Figure 16-12 Import Wizard Migration Summary

- i. Wait until the migration process completes. The Migration Results window is shown (Figure 16-13 on page 360). Click **Close**.

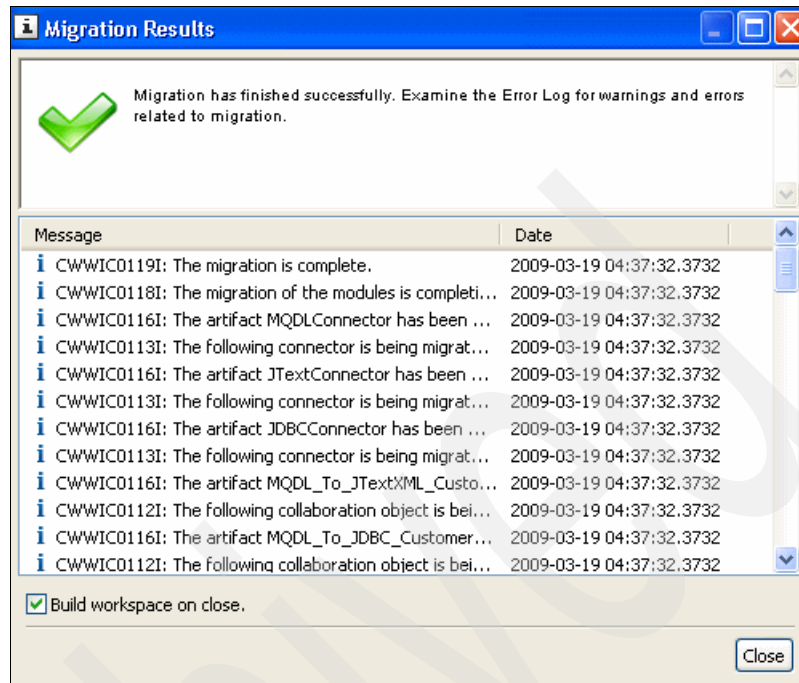


Figure 16-13 Import Wizard Migration Results

- j. The building of projects in WebSphere Integration Developer can take a while. After the build completes, the workspace contains seven new modules in addition to the DataSyncScenarioLibrary that was previously created (Figure 16-14 on page 361). Three of the new modules correspond to three WebSphere Business Integration Adapters that are used in this scenario: JDBCConnector, MQDLConnector, and JTextConnector. Two new modules correspond to the collaborations from WebSphere InterChange Server MQDL_To_JDBC_CustomerSync and MQDL_To_JTextXML_CustomerSync. The other two modules, CWBC_JDBC and CWFF_FlatFile, are resource files for WebSphere Adapter for JDBC Adapter and WebSphere Adapter for Flat Files. We will update WebSphere Business Integration Adapters to WebSphere Adapters in the following steps.

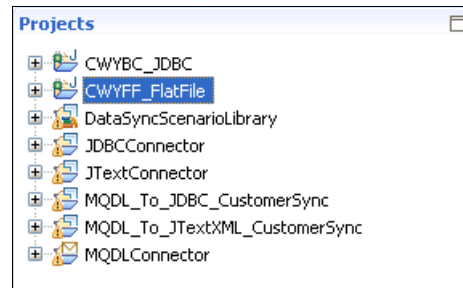


Figure 16-14 Module created

3. Upgrade WebSphere Business Integration Adapter for JText to WebSphere Adapter for Flat Files:
 - a. In the Business Integration perspective, right-click the module **JTextConnector**. Select **Update** → **Migrate Adapter Artifacts** (Figure 16-15).

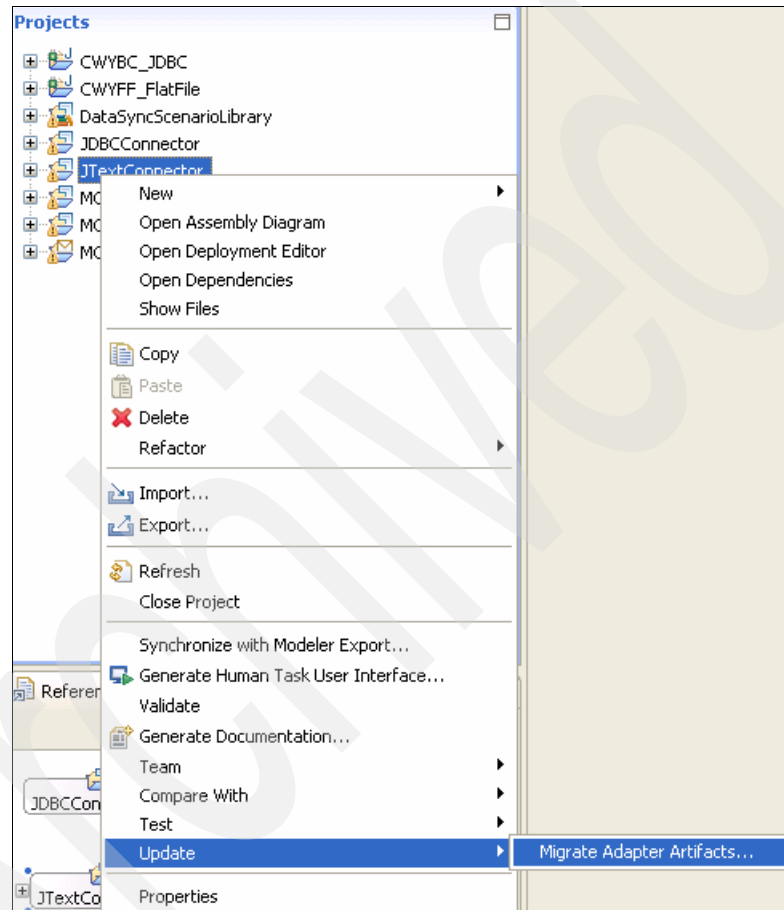


Figure 16-15 Update WebSphere Business Integration Adapter for JText

- b. In the Migration Wizard: Select Projects window (Figure 16-16), accept the default value, and click **Next**.

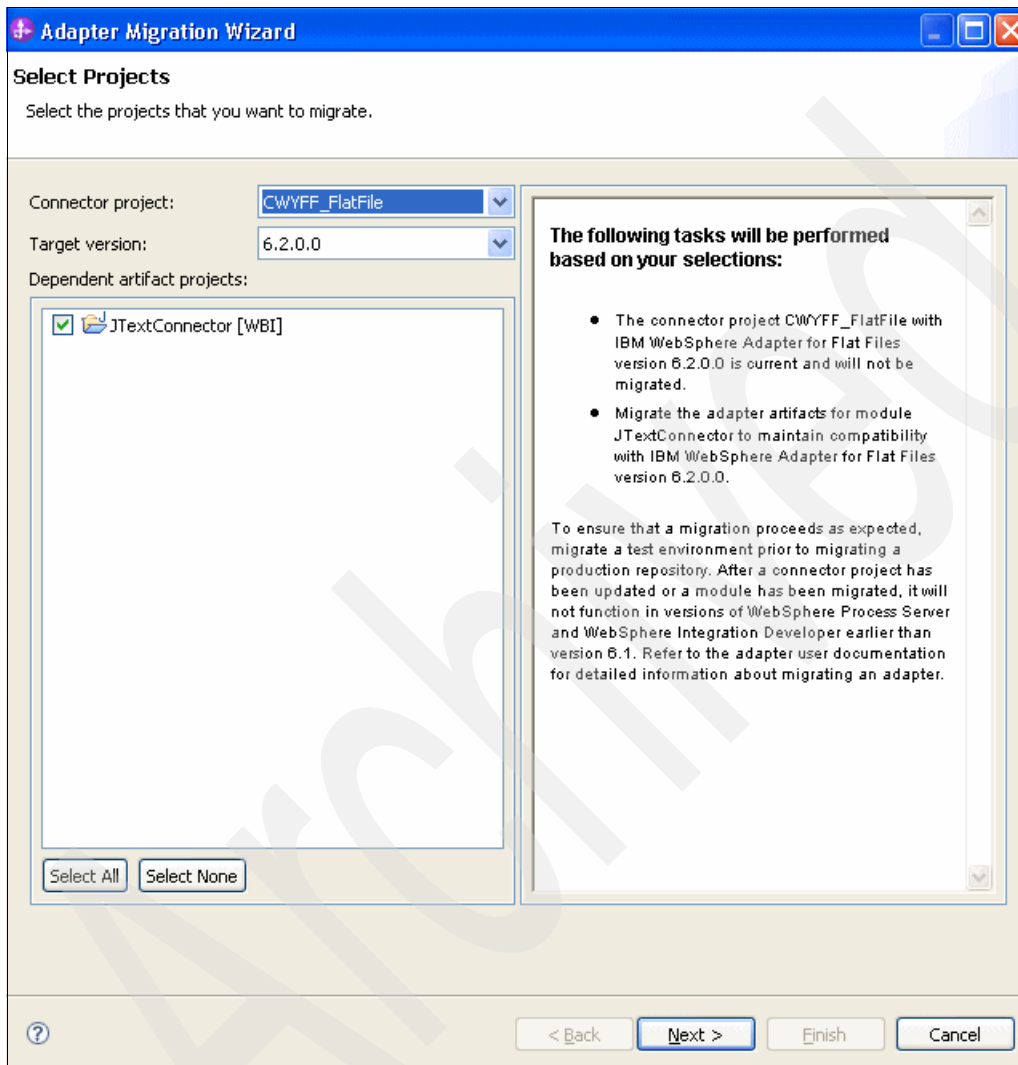


Figure 16-16 Upgrade WebSphere Business Integration Adapter for JText: Select Projects

- c. In the Migration Wizard Warning window (Figure 16-17), click **OK** to continue.

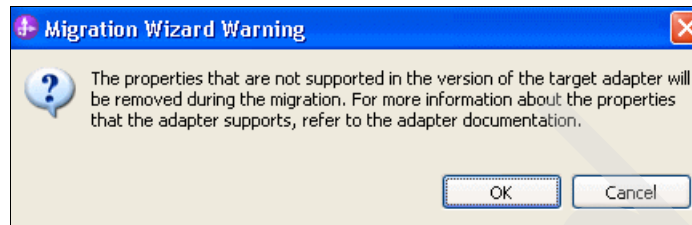


Figure 16-17 Upgrade WebSphere Business Integration Adapter for JText: Warning

- d. The Migration Wizard takes a second to collect information. When you see the Adapter Migration Wizard: Review Changes window (Figure 16-18), click **Finish**.

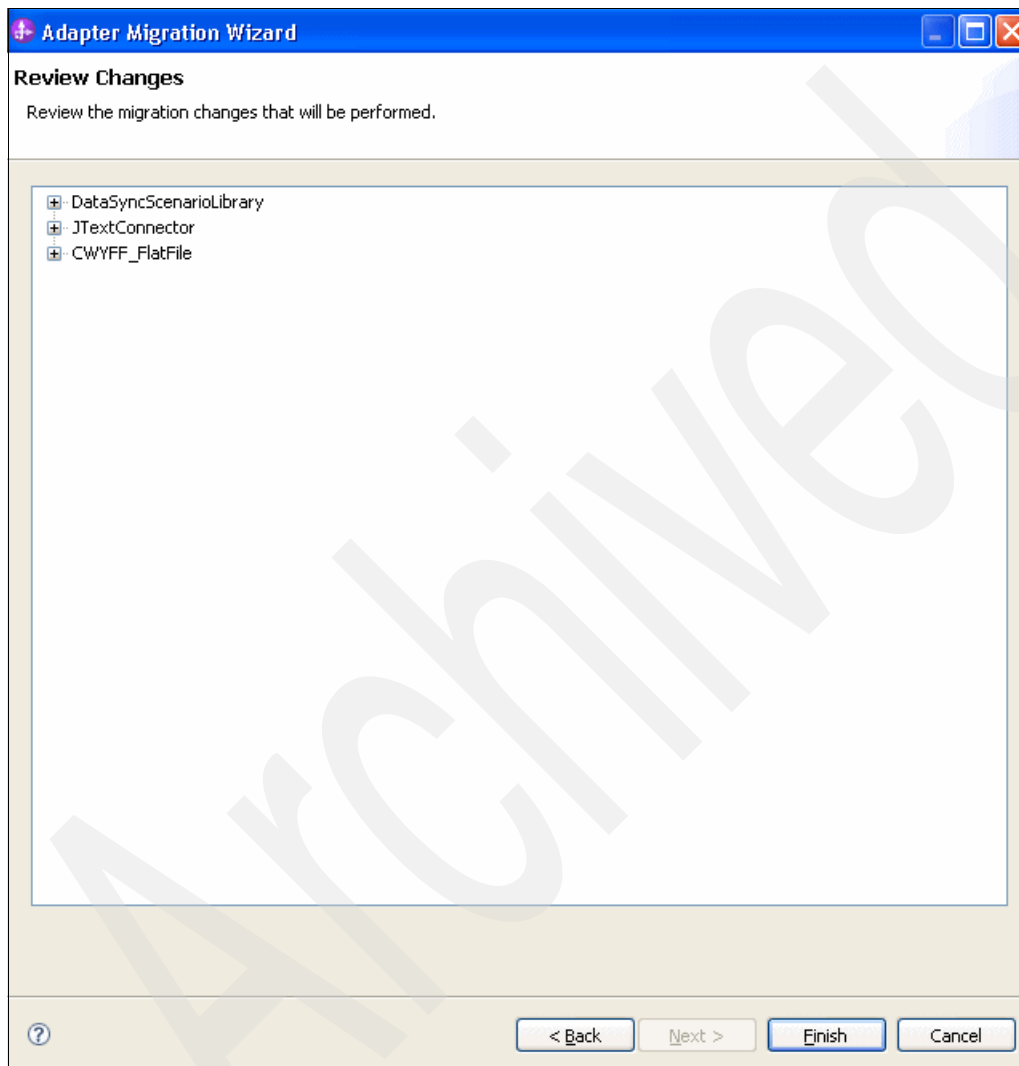


Figure 16-18 Upgrade WebSphere Business Integration Adapter for JText: Review Changes

- e. Then, the migration executes, and the windows close when finished.

4. Upgrade WebSphere Business Integration Adapter for JDBC to WebSphere Adapter for JDBC:
 - a. In the Business Integration perspective, right-click the module **JDBCConnector**. Select **Update** → **Migrate Adapter Artifacts** (Figure 16-19).

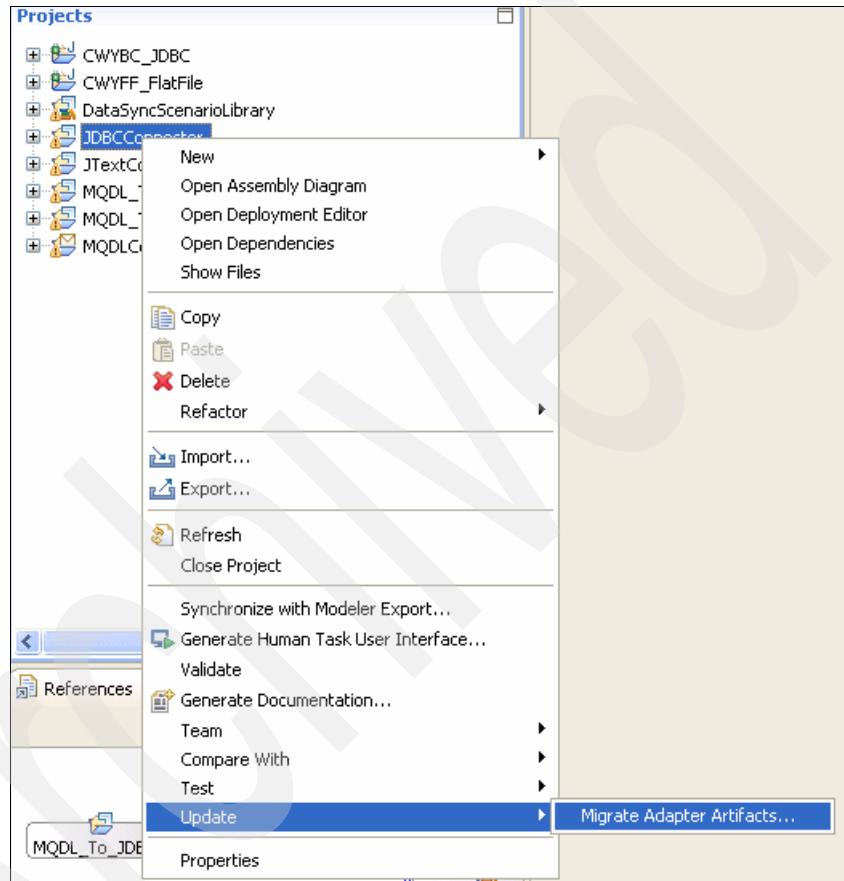


Figure 16-19 Upgrade WebSphere Business Integration Adapter for JDBC

- b. In the Migration Wizard, select the Project window (Figure 16-20), accept the default value, and click **Next**.

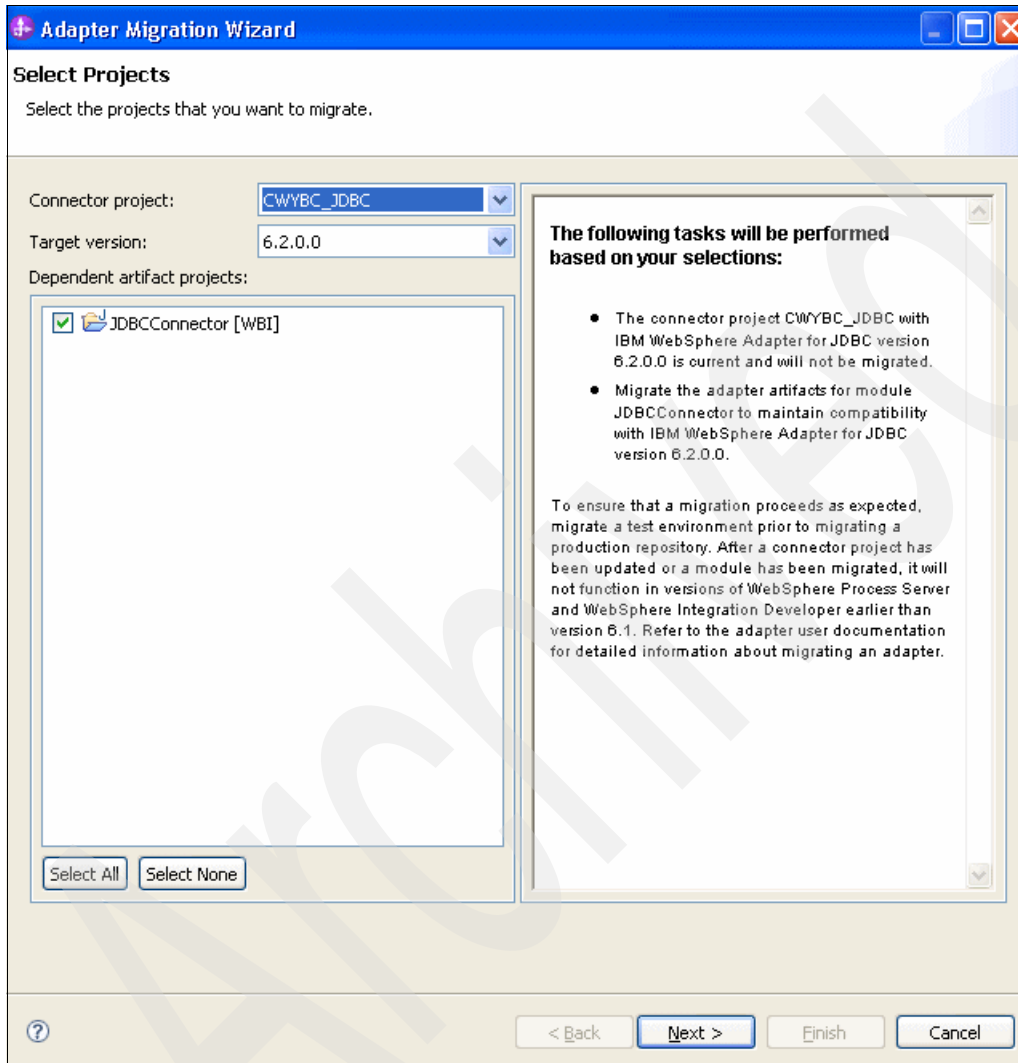


Figure 16-20 Upgrade WebSphere Business Integration Adapter for JDBC: Select Projects

- c. In the Migration Wizard Warning window (Figure 16-21), click **OK** to continue.

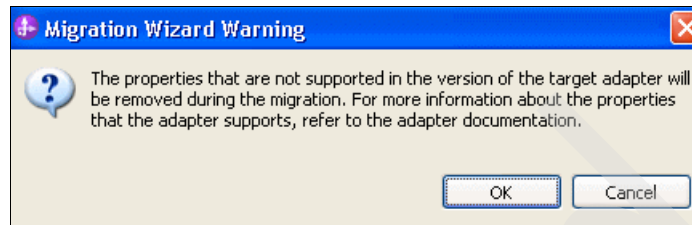


Figure 16-21 Upgrade WebSphere Business Integration Adapter for JDBC: Warning

- d. The Migration Wizard takes a second to collect information. When you see the Adapter Migration Wizard: Review Changes window (Figure 16-22), click **Finish**.

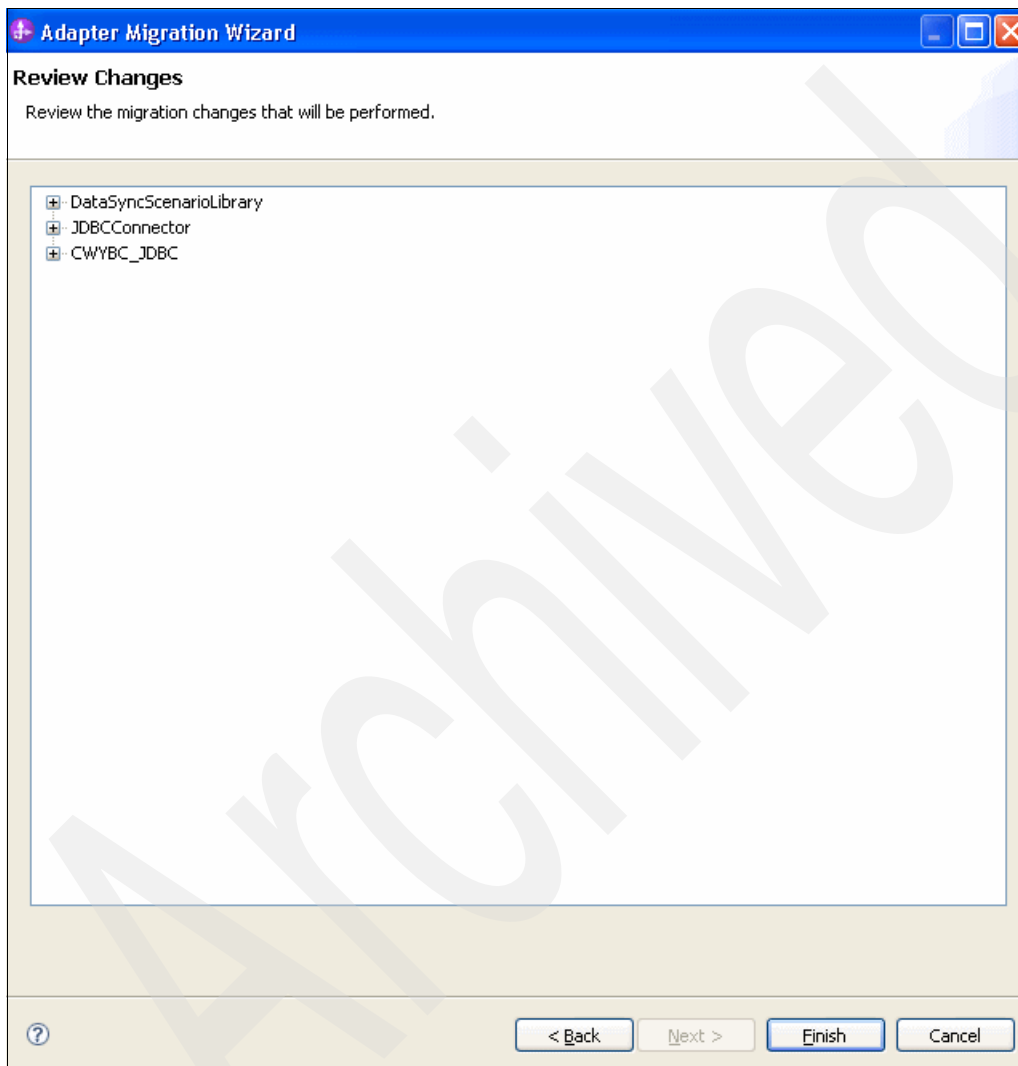


Figure 16-22 Upgrade WebSphere Business Integration Adapter for JDBC: Review Changes

- e. Then, the migration executes, and the windows close when finished.

5. Because the business object (BO) definitions between WebSphere InterChange Server and WebSphere Process Server differ slightly, the copy BO API does not work. You need to rewrite the code to use the WebSphere Process Server API instead. Modify the Java snippet in BPEL:
 - a. In the Business Integration view, expand the **MQDL_To_JDBC_CustomerSync** module, expand **Integration Logic**, expand **Processes**, and double-click **CustomerSync_From** to open the process.
 - b. Click the process name on the top of the right side to modify the process properties (Figure 16-23).

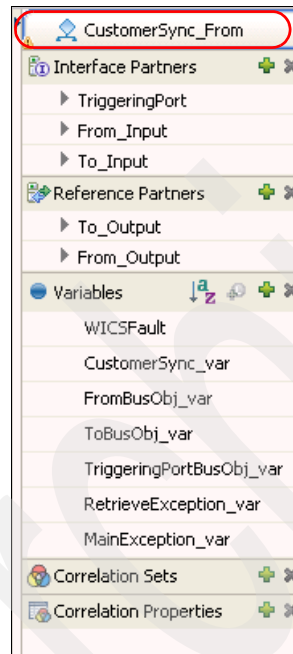


Figure 16-23 Modify process: Open process properties

- c. In the Process - CustomerSync_From properties view, select the **Java Imports** tab, and modify the following line (Figure 16-24 on page 371):


```
import com.ibm.websphere.bo.BOFactory;
```

to:

```
import com.ibm.websphere.bo.*;
```

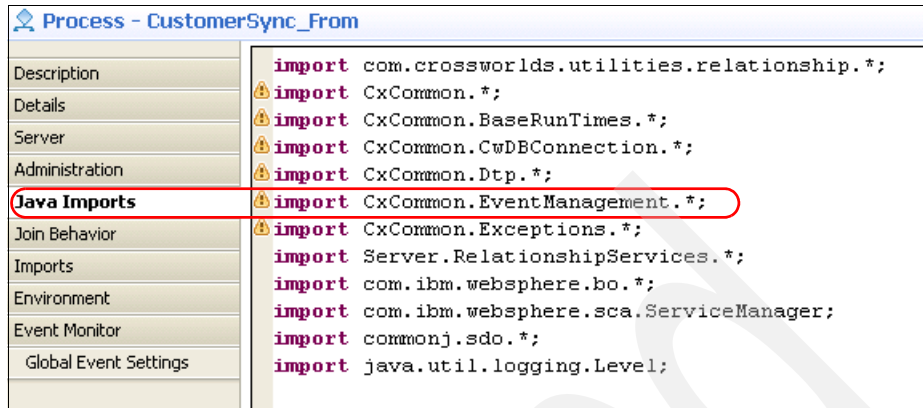


Figure 16-24 Modify process: Modify Java Imports

- d. Right-click the canvas, and select **Expand All Activities**.
- e. Right-click the canvas again, and select **Align Parallel Activities Contents Automatically**.
- f. Find the node Initialize Parameters_1 (see Figure 16-25 on page 372).

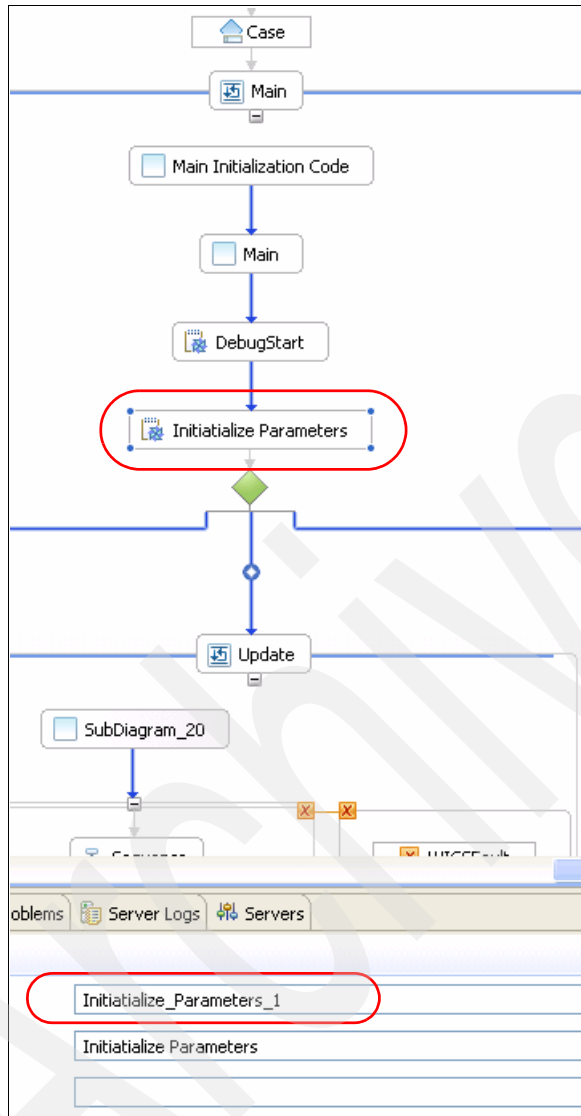


Figure 16-25 Modify process: Select node

- g. In the Snippet _ Initialize Parameters property view, select the **Detail** tab.
- h. Modify the Java snippet to look like Example 16-1 on page 373.

Example 16-1 Modifying the CustomerSync_From_Modify node Initialize_parameters Java snippet

```
//var_2.copy(var_4);
//var_3.copy(var_14);
}
BOCopy boCopy =
(BOCopy)ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOC
opy");
        FromBusObj_var= boCopy.copy(TriggeringPortBusObj_var);
        ToBusObj_var = boCopy.copy(TriggeringPortBusObj_var);

//if (triggeringBusObj == null) { TriggeringPortBusObj_var = null; }
else { TriggeringPortBusObj_var = triggeringBusObj.getBusinessGraph();
}
//if (FromBusObj == null) { FromBusObj_var = null; } else {
FromBusObj_var = FromBusObj.getBusinessGraph(); }
//if (ToBusObj == null) { ToBusObj_var = null; } else { ToBusObj_var =
ToBusObj.getBusinessGraph(); }
```

- i. Save and close the **CustomerSync_From** process.
 - j. Repeat the same steps for the process CustomerSync_From in module MQDL_To_JTextXML_CustomerSync.
6. Modify the .mon file for each process:
- a. Switch to the Java perspective. In the WebSphere Integration Developer menu bar, click **window** → **Open Perspective** → **Other**.
 - b. In the Open Perspective window (Figure 16-26 on page 374), select **Java**, and click **OK**.

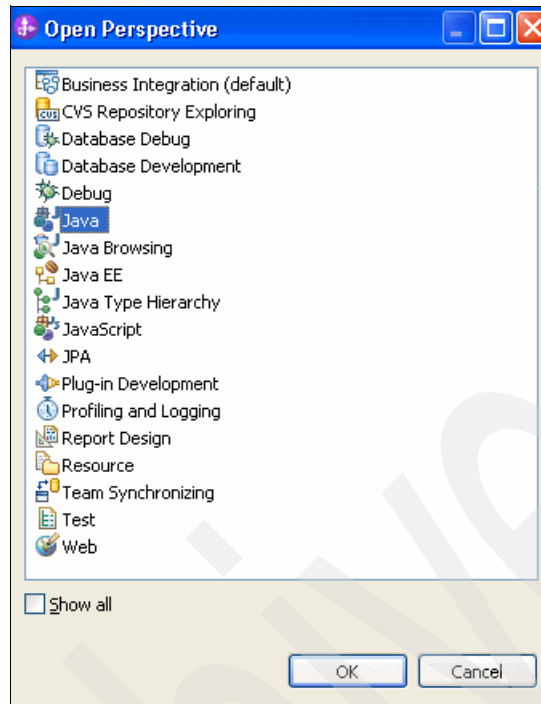


Figure 16-26 Open Perspective: Java

- c. Expand **MQDL_To_JDBC_CustomerSync**. Double-click **CustomerSync_From_bpel.mon** (Figure 16-27).

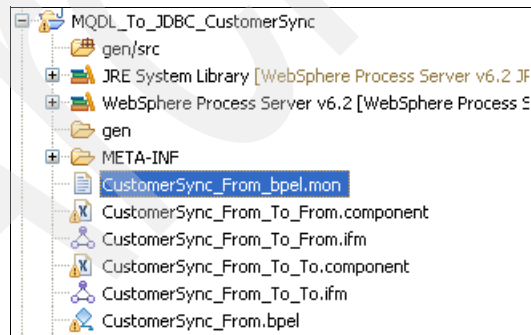


Figure 16-27 Select .mon file of process

d. Modify the following line from:

```
<Entry key="IdentifyingAttribute" value="name"/>
```

to:

```
<Entry key="IdentifyingAttribute" value="wpc:ids"/>
```

See Figure 16-28.



Figure 16-28 Modify .mon file for process

e. Repeat the previous steps for the MQDL_To_JTextXML_CustomerSync project.

Note: The file with suffix .mon does not exist after migration. It will be generated after the modification of the process (BPEL) when building the project. Make sure that this step is performed after step 7.

7. After the migration, the MQDLConnector module is not wired to the MQDL_To_JTextXML_CustomerSync module. To wire the MQDLConnector to the MQDL_To_JTextXML_CustomerSync module:

a. Remove the incorrect node in the mediation flow:

- i. Switch to the Java perspective.
- ii. Locate the file MQDLConnector.medflow under the folder MQDLConnector. Right-click **MQDLConnector.medflow**. Select **Open with** → **xml editor**.
- iii. Find the node as shown in Figure 16-29 on page 376, right-click the nodes, and select **Remove**.

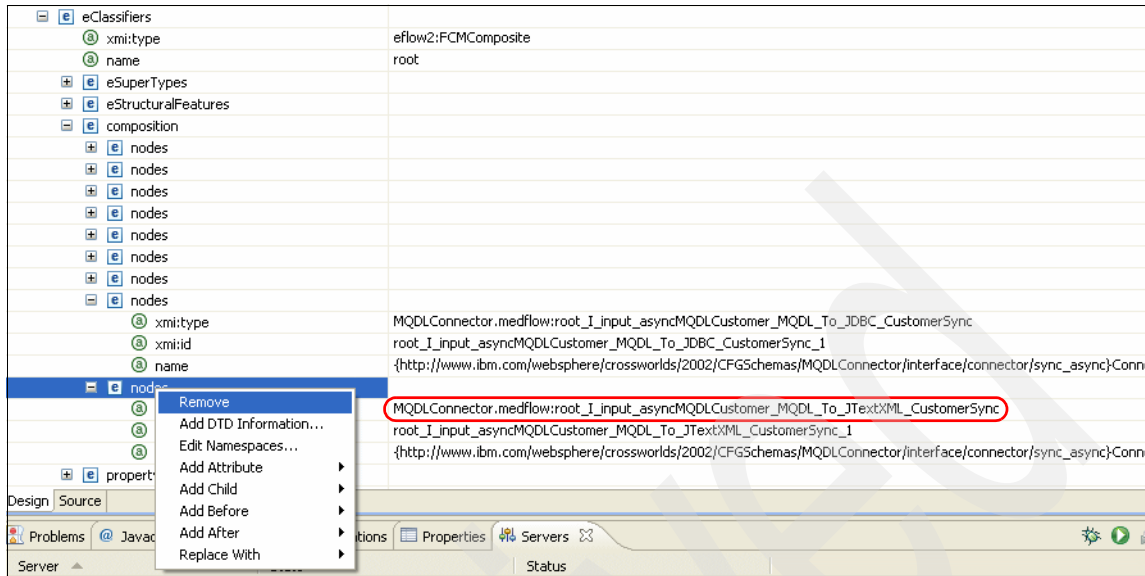


Figure 16-29 Connect MQDL_To_JTextXML_CustomerSync: Remove node 1 in .medflow file

- iv. Find the other node that is shown in Figure 16-30, right-click the node, and select **Remove**.

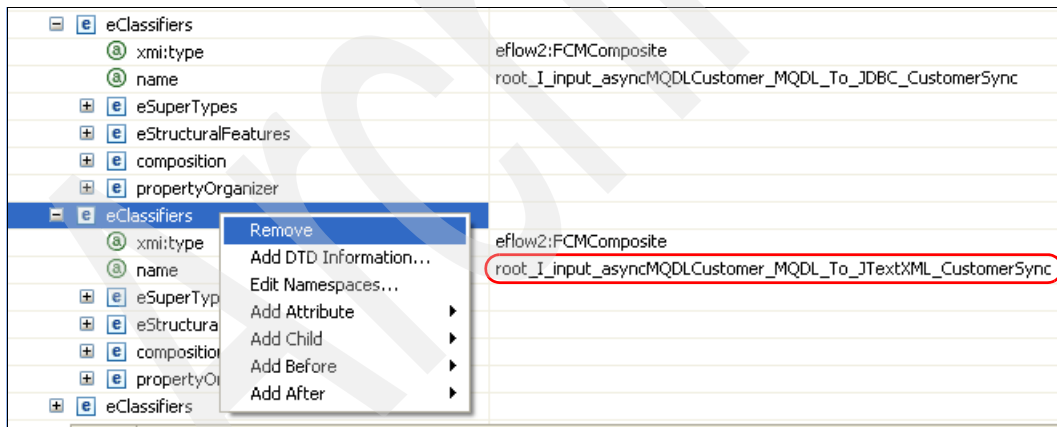


Figure 16-30 Connect MQDL_To_JTextXML_CustomerSync: Remove node 2 in .medflow file

- v. Save and close the file.

- b. Remove the connection in the Mediation Flow Component (MFC):
 - i. Perform the same task in the Java perspective, find the MQConnector.mfc file under MQDLConnector. Right-click **MQConnector.mfc file**, and select **Open with** → **xml editor**.
 - ii. Find the node that is shown in Figure 16-31. There are two duplicate nodes, right-click one of them, and select **Remove**.

extendedOperationBinding	
source	MQDLCustomerBG
target	Async_CustomerBG
sourcePortType	connector:Connector_PortType
targetReference	CustomerBG_PortTypePartner
extendedOperationBinding	
source	MQDLCustomerBG
target	Async_CustomerBG
sourcePortType	connector:Connector_PortType
targetReference	CustomerBG_PortTypePartner
extendedOperationBinding	
extendedOperationBinding	
operationFlow	

Figure 16-31 Connect MQDL_To_JTextXML_CustomerSync: Remove the node in the .mfc file

- iii. Save and close the file.
- c. Add a reference in the mediation flow MQDLConnector component in the assembly diagram:
 - i. Switch back to the Business Integration perspective.
 - ii. Open the assembly diagram of the MQDLConnector module.
 - iii. Right-click Component **MQDLConnector(mediation flow)**, which is shown in Figure 16-32 on page 378. Select **Add** → **Reference**.

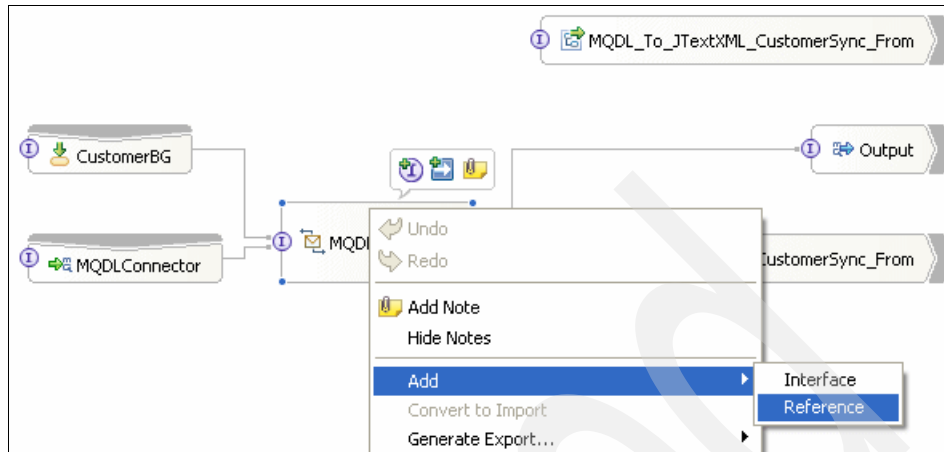


Figure 16-32 Connect MQDL_To_JTextXML_CustomerSync: Add reference for mediation flow component

- i. In the Add Reference window, select **CustomerBG_PortType** as shown in Figure 16-33 on page 379, and click **OK**.

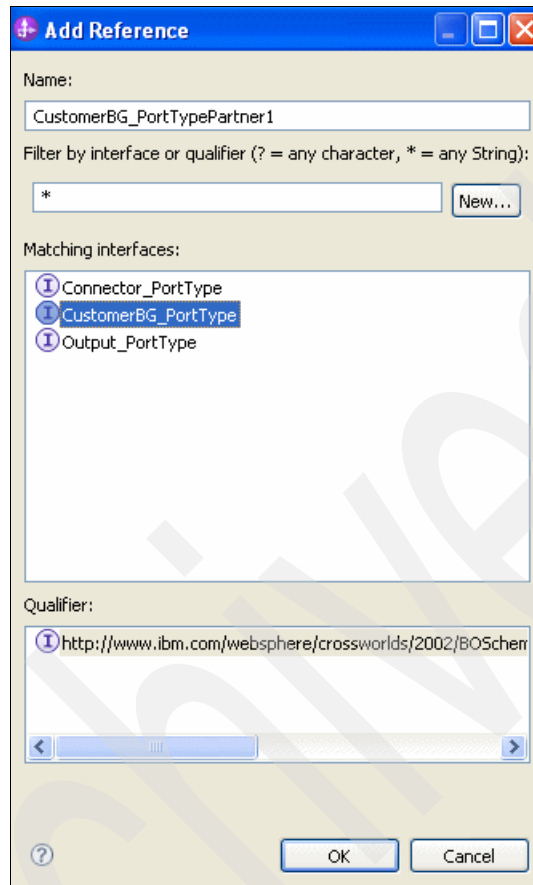


Figure 16-33 Connect MQDL_To_JTextXML_CustomerSync: Select reference

- ii. Drag a line from the new reference of Component MQDLConnector to Import MQDL_To_JTextXML_CustomerSync_From to link them together (Figure 16-34 on page 380).

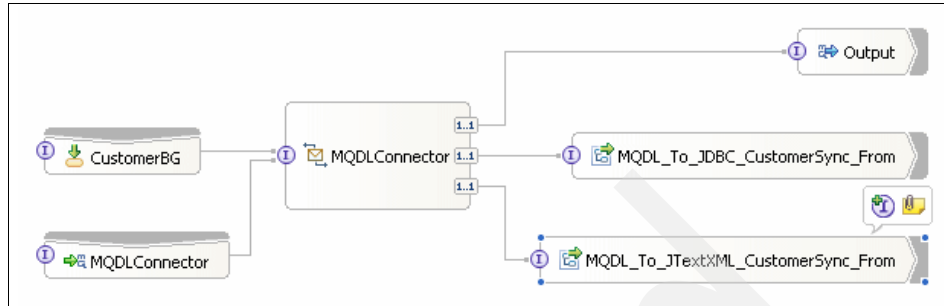


Figure 16-34 Connect MQDL_To_JTextXML_CustomerSync: Connect reference in component

- d. Add the reference in the mediation flow:
 - i. In the assembly diagram of the MQDLConnector, double-click Component **MQDLConnector(Mediation Flow)** to open it in the Mediation Flow Editor.
 - ii. Click **Add Reference**.
 - iii. In the Add Reference window, select **CustomerBG_PortType** as shown in Figure 16-35 on page 381, and click **OK**. A new reference named CustomerBG_PortTypePartner1 is added in the graph.
 - iv. Draw a line from the operation MQDLCustomerBG in the interface Connector_PortType to the operation Async_CustomerBG in the reference CustomerBG_PortTypePartner1.
 - v. Click the link that was added in the last step. In the Palette, draw a line to link BO Map MQDLCustomer_and_Customer_SMO_Map and Call out Async_CustomerBG : CustomerBG_PortTypePartner1. Save and close the Mediation Flow Editor (Figure 16-35 on page 381).

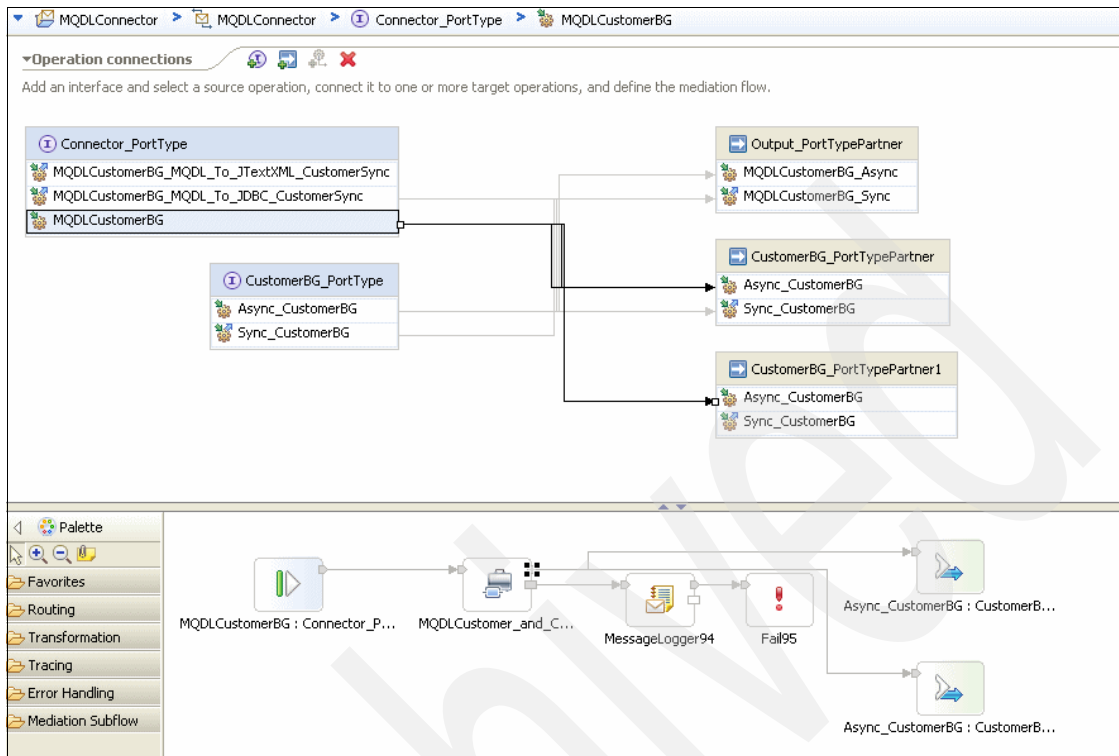


Figure 16-35 Connect MQDL_To_JTextXML_CustomerSync: Connect reference in mediation flow

- vi. Save.
8. Set the destination queue name for the MQ binding in the module MQDLConnector:
 - a. In the assembly diagram of module MQDLConnector, select **Export: MQDLConnector (MQ Binding)**.
 - b. In the properties view, select **Binding** → **end-point configuration**.
 - c. In the field Send destination queue, enter DummyQueue1 (Figure 16-36 on page 382).

Export: MQDLConnector (MQ Binding)

Messaging Resource Configuration

Select configuration view option:

- ☒ Specify properties for configuring WebSphere MQ resources
- ☐ Specify JNDI name for pre-configured WebSphere MQ resources

End-point configuration

Message configuration

Method bindings

Faults configuration

Security attributes

Propagation

Request queue manager: * CUST.QUEUE.MANAGER

Receive destination queue:* WICS.INPUT.Q

Respond queue manager: CUST.QUEUE.MANAGER

Send destination queue: * **DummyQueue1**

Figure 16-36 MQDLConnector: Set Destination Queue for MQ Binding Export

- d. Select Import: OutPut (MQ Binding).
- e. In the properties view, select **Binding** → **end-point configuration**.
- f. In the field Send destination queue, enter DummyQueue2 (Figure 16-37).

Import: Output (MQ Binding)

Messaging Resource Configuration

Select configuration view option:

- ☒ Specify properties for configuring WebSphere MQ resources
- ☐ Specify JNDI name for pre-configured WebSphere MQ resources

End-point configuration

Message configuration

Method bindings

Faults configuration

Security attributes

Propagation

Request queue manager: * CUST.QUEUE.MANAGER

Send destination queue: * **DummyQueue2**

Receive destination queue:* WICS.REPLYTO.Q

Figure 16-37 MQDLConnector: Set Destination Queue for MQ Binding Import

- g. Save and close the assembly diagram of module MQDLConnector.
9. After the migration, the output BO in operation createJTextXMLCustomer of interface OutPut_PortType is changed, so you need to add code to apply the changes:
 - a. Open the Java file Output_Processing.java in the JTextconnector.
 - b. Find the method BusinessObjectBG_Sync(DataObject BusinessObjectBG). Replace the return code with the following code that is shown in Example 16-2 on page 383. Save and close.

```
String id =
output.getDataObject("createJTextXMLCustomerOutput").getDataObject("CreateResponse").
getString("filename");
BOFactory bof =
(BOFactory)ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory");
DataObject rJTextXMLCustomerBG =
bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/JTextXMLCustomer/J
TextXMLCustomerBG", "JTextXMLCustomerBG");
DataObject JTextXMLCustomerBO =
rJTextXMLCustomerBG.createDataObject("JTextXMLCustomer");
JTextXMLCustomerBO.setString("ID", id);
return rJTextXMLCustomerBG;
```

10. The operation name that was generated by Output_Processing.java is not correct, so you need to modify it manually:
- Open the assembly diagram of Module JTextConnector (Figure 16-38). Double-click component **Output_Processing** to open the Java file Output_Processing.java.

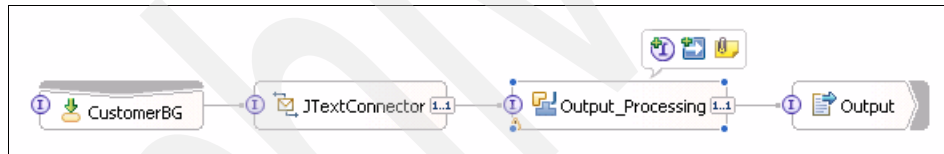


Figure 16-38 JTextConnector Assembly Diagram

- In the Output_Processing.java file, add the code in method getOperation() as shown in Example 16-3 before you return to Figure 16-39 on page 384.

```
if (operation.endsWith("Wrapper")) {
    operation = operation.substring(0, operation.length() - "Wrapper".length());
}
```

```

private String getOperation(String verb, DataObject dataObject)
    throws Exception {
    String operation = null;
    String outRootTypeName = null;
    outRootTypeName = WPSServiceHelper.getRootBusinessObjectInstance(
        dataObject).getType().getName();
    if (verb == null) {
        throw new MediateException(
            "The operation to invoke on the target interface could not be determ
    }
    if (outRootTypeName == null) {
        throw new MediateException(
            "The operation to invoke on the target interface could not be determ
    }
    verb = verb.toLowerCase();
    if (verb.equalsIgnoreCase("create")) {
        operation = verb + outRootTypeName;
    }
    if (operation.endsWith("Wrapper")) {
        operation = operation.substring(0, operation.length() - "Wrapper".length())
    }
    return operation;
}

```

Figure 16-39 Modify Output_Processing.java file

11. Specify the Sequence file path for WebSphere Business Integration Adapter for JText (Figure 16-40 on page 385).

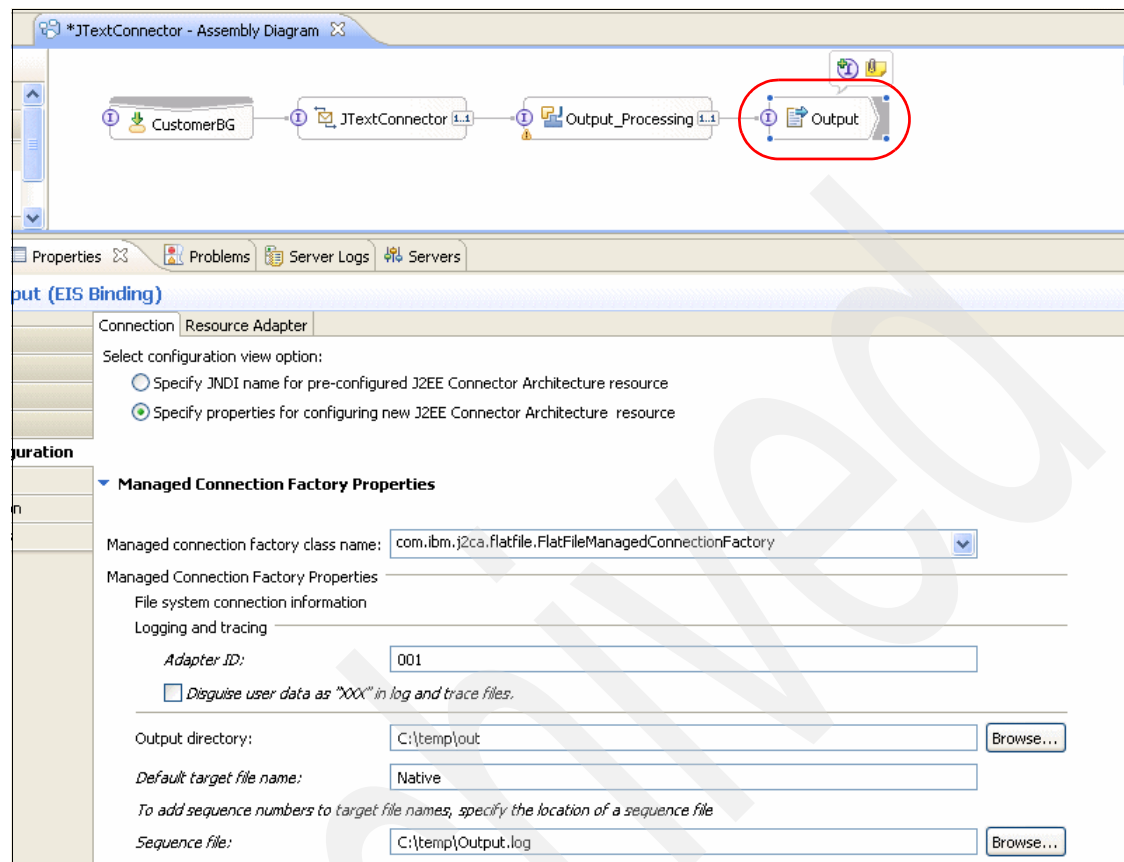


Figure 16-40 Specify Sequence file path for WebSphere Adapter for Flat Files

12. The JDBC driver of WebSphere Adapter for JDBC differs from the WebSphere Business Integration Adapter for JDBC. To Modify the JDBC driver for WebSphere Adapter for JDBC:
- Switch to the Java perspective. In the WebSphere Integration Developer menu bar, click **window** → **Open Perspective** → **Other**.
 - In the Open Perspective window (Figure 16-41), select **Java**, and click **OK**.

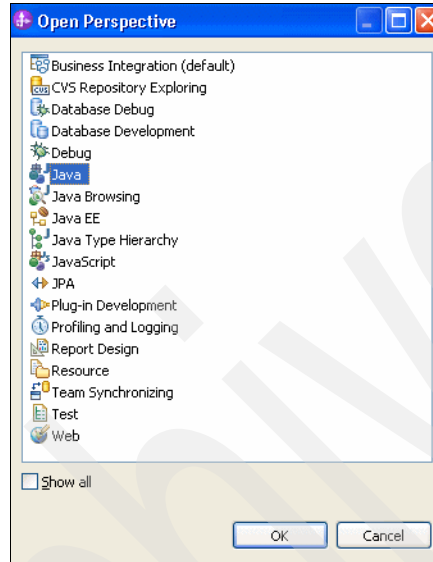


Figure 16-41 Open Perspective: Java

- c. In the Package Explorer tab, expand **JDBCConnector**, and double-click **Output.import** to open the file as shown in Figure 16-42.

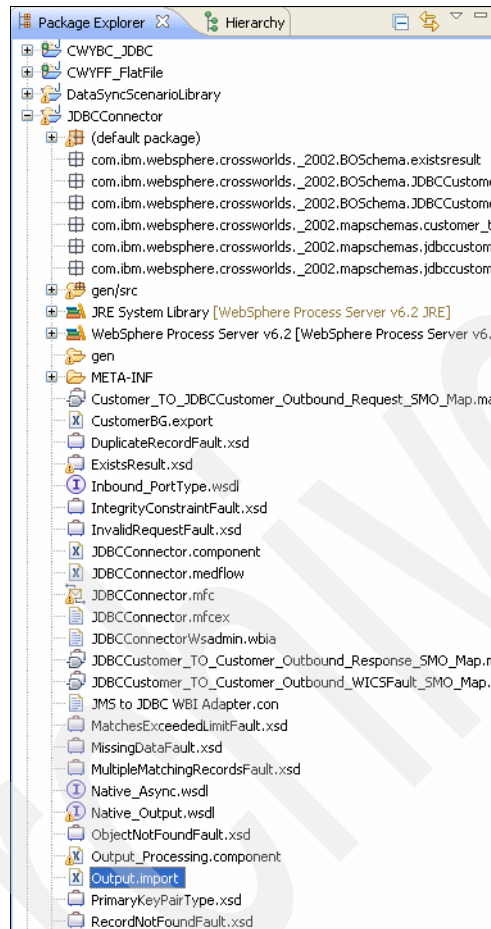


Figure 16-42 Update JDBC Driver Path: Select file

- d. In the output.import file, find this line:
- ```
<jdbcDriverClass>COM.ibm.db2.jdbc.app.DB2Driver</jdbcDriverClass>
```
- Change it to this line:
- ```
<jdbcDriverClass>com.ibm.db2.jcc.DB2Driver</jdbcDriverClass>
```
- e. Save and close the file.

16.2.4 Work-around for dynamic relationship

After migration, dynamic relationships do not work because of differences in the transaction management approaches between WebSphere InterChange Server and WebSphere Process Server. In WebSphere Process Server, transactions are managed by the underlying WebSphere Application Server. WebSphere InterChange Server does not have this capability. Because of this key difference, you must modify the stored procedures that manage the dynamic relationship tables so that there are no explicit commits or rollbacks when used with WebSphere Process Server.

In this specific scenario, we have one dynamic relationship, Customer, which uses three stored procedures. You only need to modify two of them: MQDLCUST_RELATIONSHIP_SP and JDBCCUST_RELATIONSHIP_SP. These stored procedures each have one rollback and one commit statement.

By using DB2 Development Center, modify the stored procedures by either commenting out or removing the rollback and commit statements:

1. Launch the DB2 Development Center. Select **Start** → **Programs** → **IBM DB2** → **Development Tools** → **Development Center**. The stored procedure wizard automatically launches.
2. In the Development Center Launchpad window (Figure 16-43 on page 389), click **Create Project**.

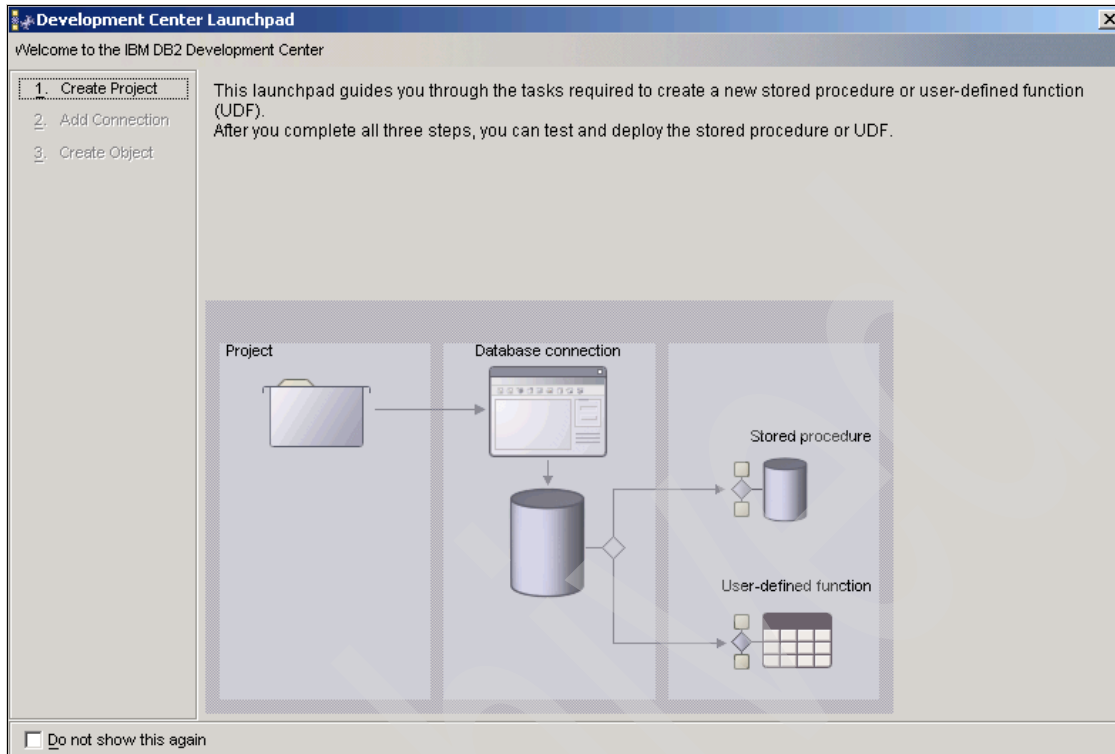


Figure 16-43 DB2 Development Center: Launchpad for the stored procedure

3. In the Open Project window (Figure 16-44), select a project name. Use the default value of Project1 if it does not already exist. Click **OK**.

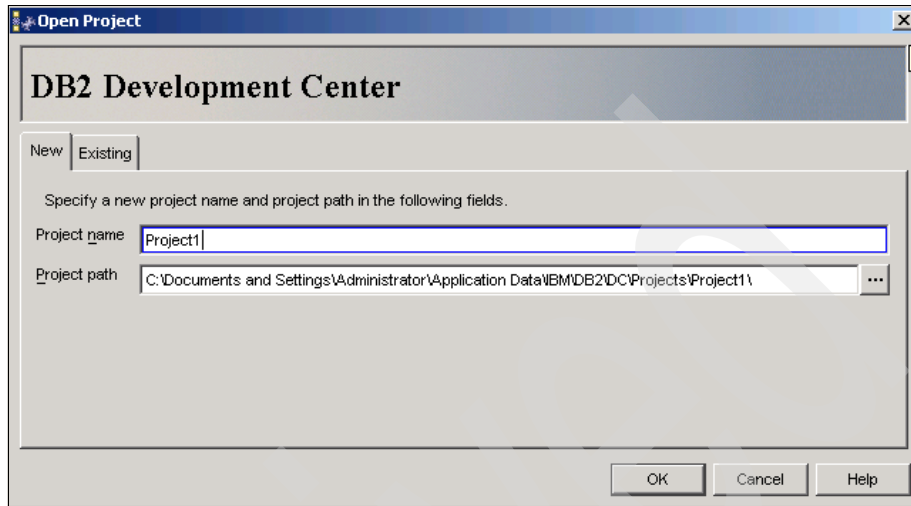


Figure 16-44 DB2 Development Center: Creating a project

4. In the Development Center Launchpad window (Figure 16-45), click **Add Connection**.

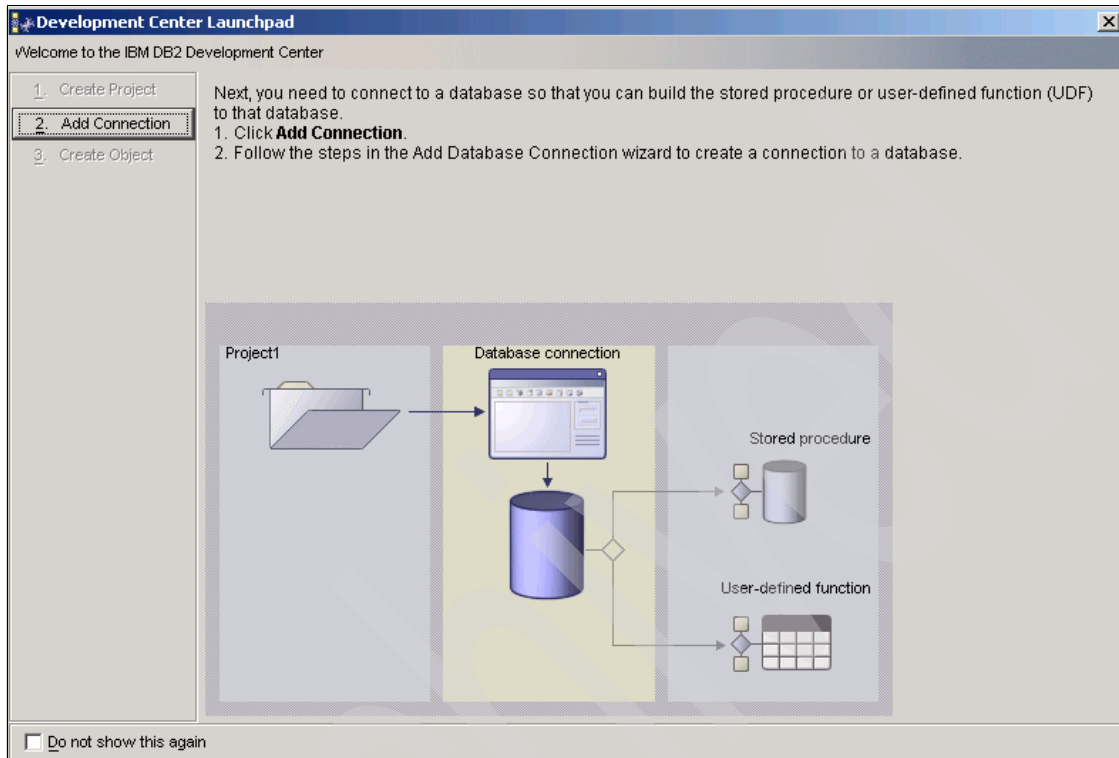


Figure 16-45 DB2 Development Center: Add Connection

5. In the Add Database Connection Wizard window (Figure 16-46), in the Connection Type pane, select **Online**. Click **Next**.

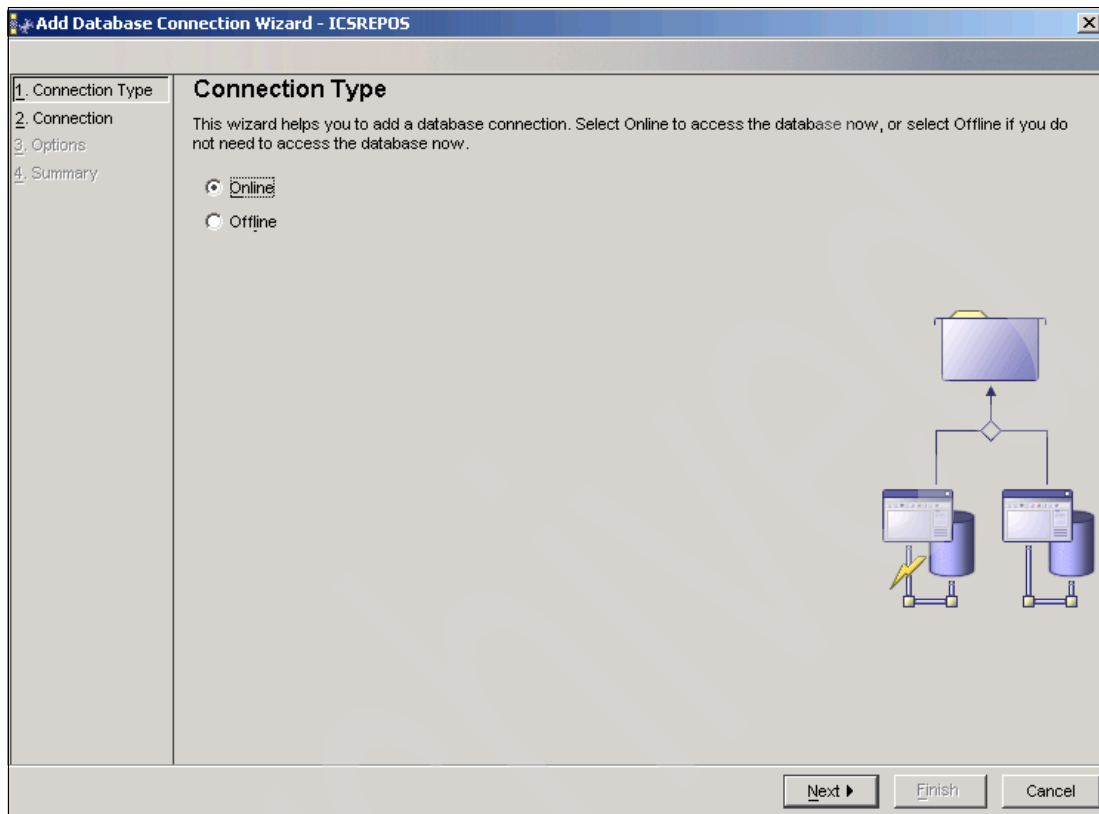


Figure 16-46 DB2 Development Center: Connection Type

6. In the next window (Figure 16-47), in the Connection pane, provide the connection details for the database. Click **Finish**, and close the connection wizard.

Add Database Connection Wizard - ICSREPOS

1. Connection Type
2. **Connection**
3. Options
4. Summary

Connection
Specify a driver and the database alias that you want to use for your connection.

Driver: IBM DB2 alias

Database:
Database: ICSREPOS
Comments:
Host name:
Port number:
Location (URL): jdbc:db2:ICSREPOS
Driver class: COM.ibm.db2.jdbc.app.DB2Driver

User information:
☐ Use your current user ID and password
User ID: Administrator
Password: *****

Test Connection

Navigation: ◀ Back Next ▶ Finish Cancel

Figure 16-47 DB2 Development Center: Connection

7. With the project now visible in the Project View (Figure 16-48), right-click **Stored Procedures**, and select **import**.

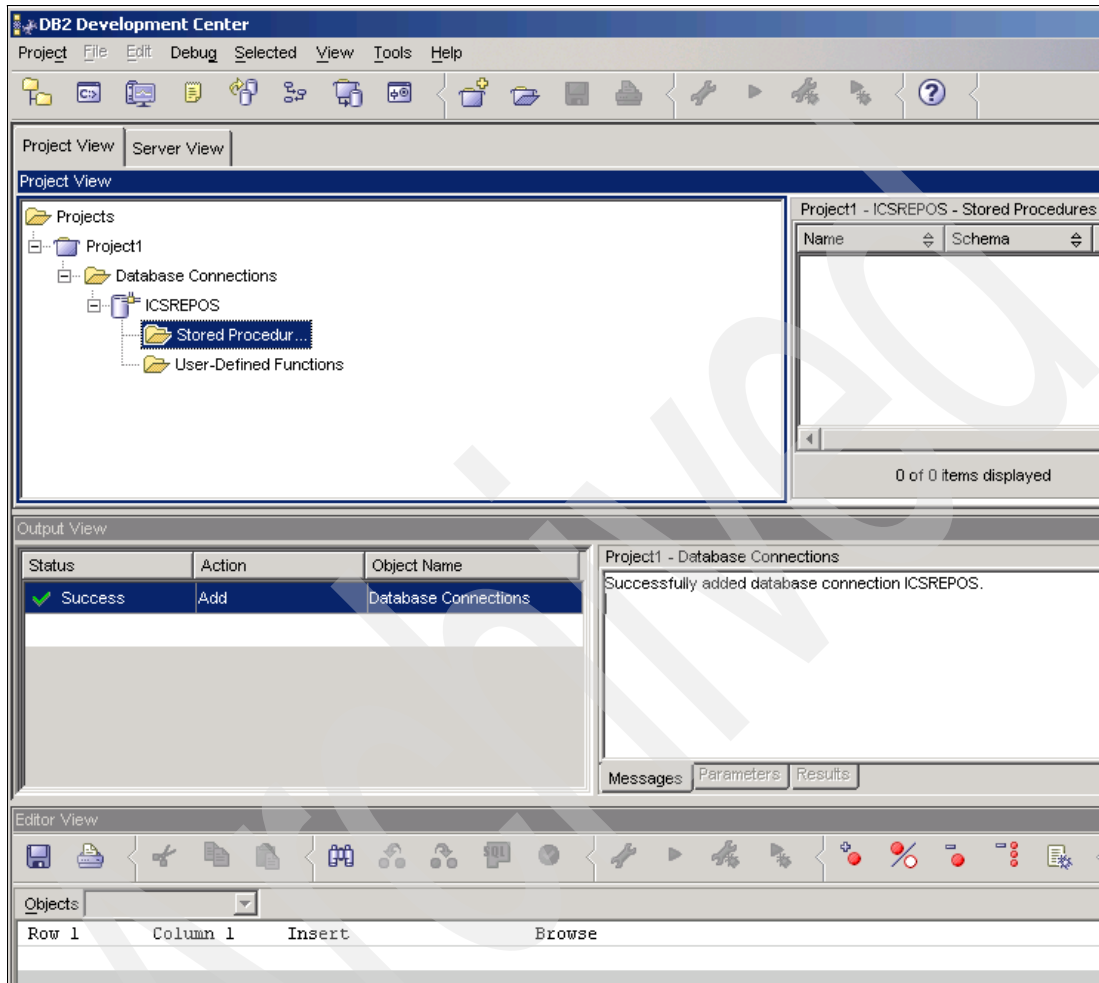


Figure 16-48 DB2 Development Center: Project created

8. In the Import window (Figure 16-49), select **Database** and **Stored Procedure**, and click **OK**.

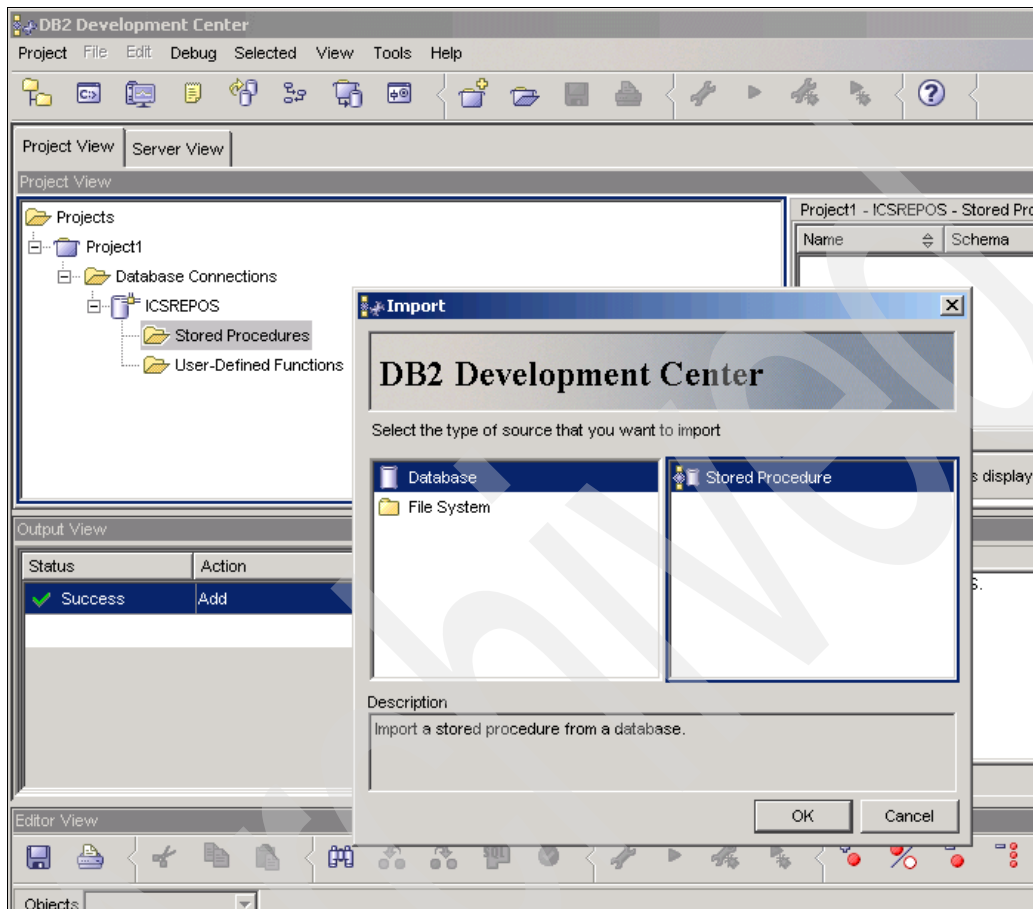


Figure 16-49 DB2 Development Center: Importing stored procedures

9. In the Source Database pane of the Import Wizard window (Figure 16-50), select the source database from the list. Click **Next**.

The screenshot shows the 'Import Wizard' window with the 'Source Database' pane selected. The pane contains a description of the wizard's purpose and several input fields for configuring the source database connection. The 'Driver' is set to 'IBM DB2 alias'. The 'Database' is 'ICSREPOS'. The 'Location(URL)' is 'jdbc:db2:ICSREPOS'. The 'Driver class' is 'COM.ibm.db2.jdbc.app.DB2Driver'. The 'User ID' is 'Administrator' and the 'Password' is masked with asterisks. A 'Next' button is visible at the bottom right.

Import Wizard

1. Source Database

This wizard helps you import objects such as stored procedures and user-defined functions from a database. Specify the alias and login information for the source database. The target database is your current connection.

Driver: **IBM DB2 alias**

Database:

Database: **ICSREPOS**

Comments:

Host name:

Port number:

Location(URL): **jdbc:db2:ICSREPOS**

Driver class: **COM.ibm.db2.jdbc.app.DB2Driver**

User information:

☐ Use your current user ID and password

User ID: **Administrator**

Password: *********

Next **Finish** **Cancel**

Figure 16-50 DB2 Development Center: Specifying the source database

10. In the Filter pane of the Import Wizard window (Figure 16-51), accept the defaults, and click **Next**.

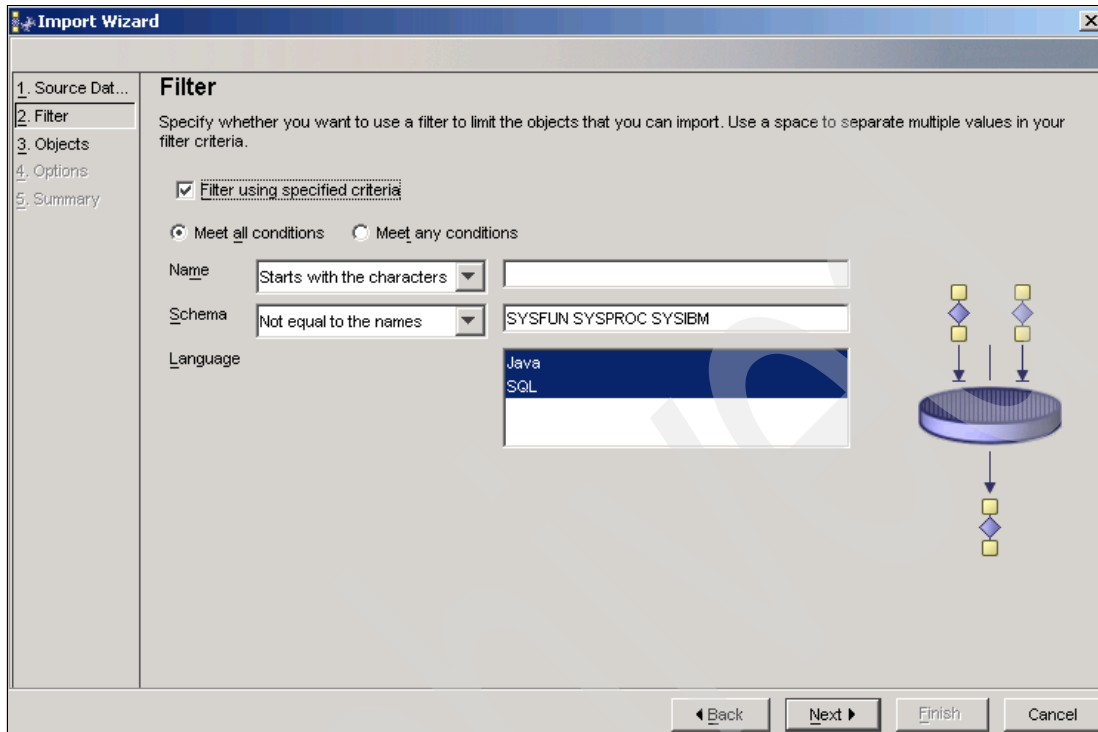


Figure 16-51 DB2 Development Center: Providing the filter details

11. In the Objects pane of the Import Wizard window (Figure 16-52), under the heading Available, select the two dynamic relationship stored procedures ***username.JDBCCUST_RELATIONSHIP_SP*** and ***username.MQDLCUST_RELATIONSHIP_SP*** to modify, and click > to move them to the Selected pane. Then, click **Finish**.

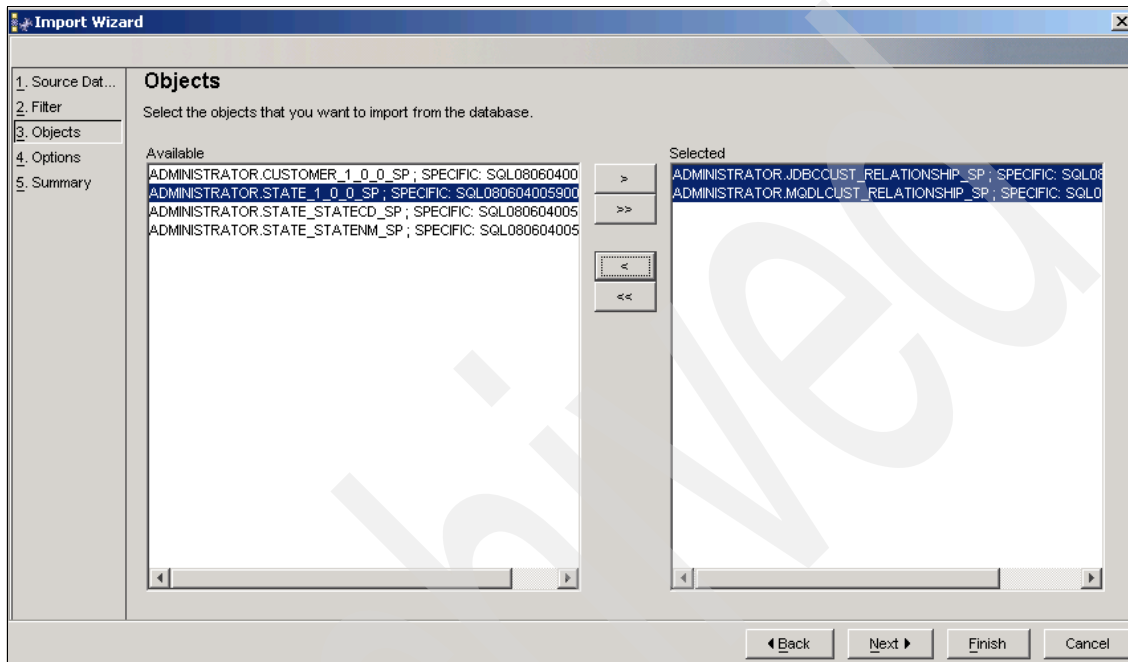


Figure 16-52 DB2 Development Center: Selecting the two stored procedures that you want to modify

12. In the DB2 Development Center window (Figure 16-53 on page 400), double-click a stored procedure to open it in the Editor View. Comment out the Rollback and Commit statements in the stored procedure. There is only one rollback statement in each stored procedure, and there is only one commit statement in each stored procedure. Click the **build** icon (a wrench) in the Editor View. Click **Yes** if you see a message window.

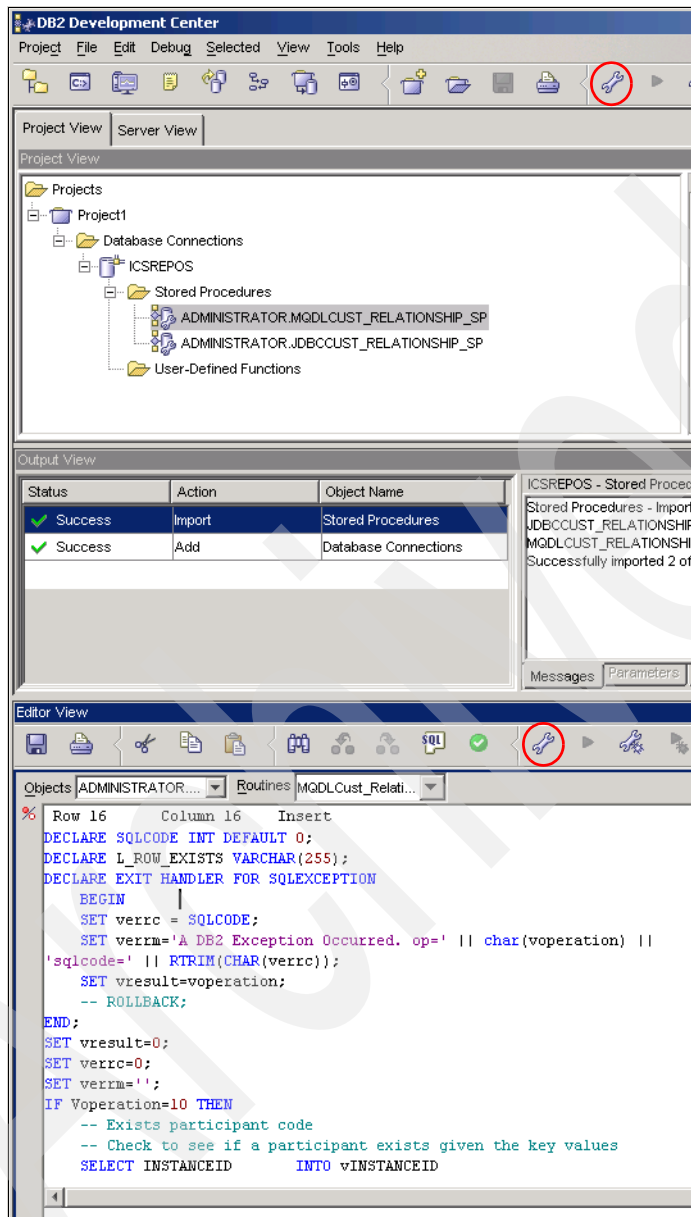


Figure 16-53 DB2 Development Center: Opening a stored procedure that you want to modify in the Editor View

16.2.5 Running a Jython script and modifying JDBC data sources

For WebSphere InterChange Server artifacts that have no corresponding artifacts in WebSphere Process Server, a Jython script is generated during migration. For this scenario, the Jython script that is generated has references to relationships. You must run this script by using the **wsadmin** command to create WebSphere Process Server configuration definitions that correspond to the original WebSphere InterChange Server artifacts.

Saving and applying changes: In the following steps, apply and save every change that you make in the administrative console.

To run this script and modify the JDBC data sources:

1. Switch your perspective in WebSphere Integration Developer (Figure 16-54).

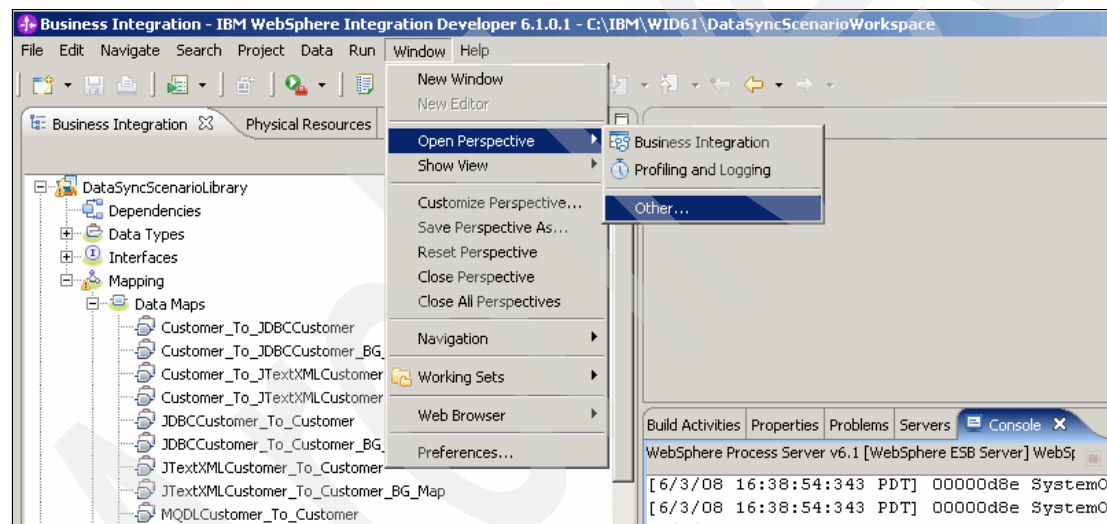


Figure 16-54 Switching perspectives

2. Select **Java**.

The Jython script `InstallAdministrativeObjects.py` is placed under the library project `DataSyncScenarioLibrary` in this scenario (Figure 16-55). The generated Jython script is self-explanatory and contains details about how to run the file.

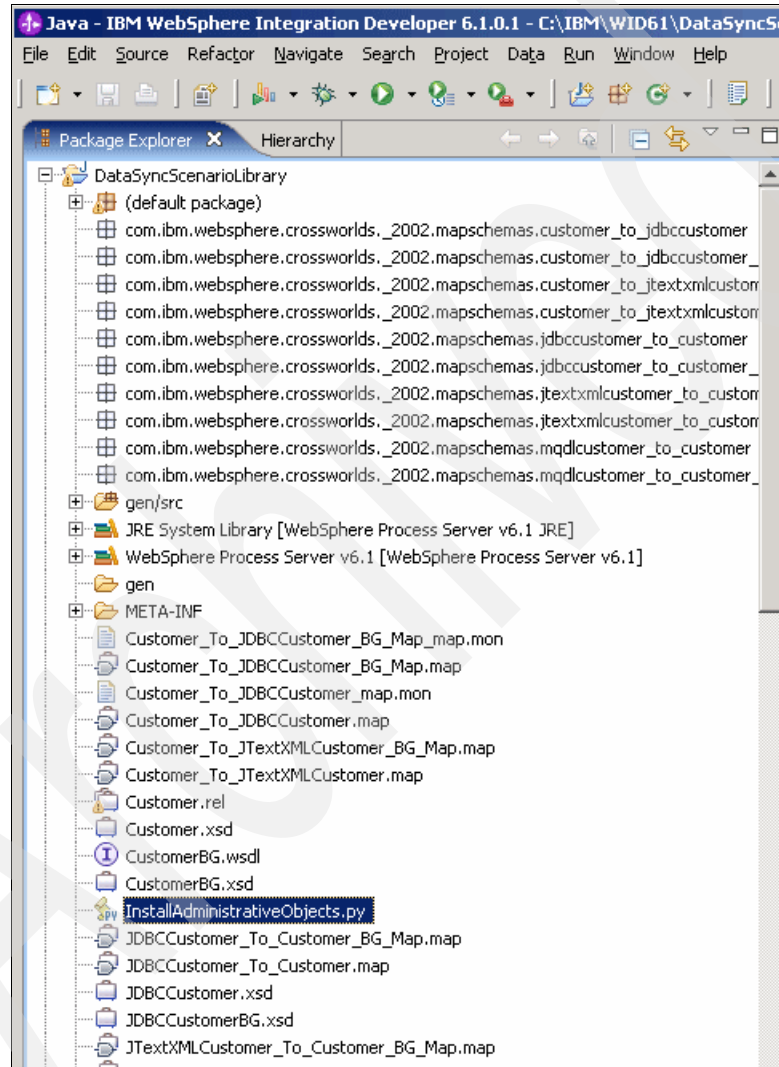


Figure 16-55 Jython script

3. Switch back to the Business Integration perspective to start the embedded instance of WebSphere Process Server. Under the **Servers** tab, right-click the server instance, and select **Start**.
4. Run the Jython script by using the following command from the bin folder of the WebSphere Process Server installation. Modify the file path if your workspace location is different:

```
C:\IBM\WID62\pf\wps\bin> wsadmin -lang jython -f
C:\DataSyncScenarioWorkspace\DataSyncScenarioLibrary\InstallAdministrativeObjects.py
```

Running this script creates a new JDBC provider and corresponding data sources that are necessary for relationships that are being used in the WebSphere InterChange Server project.

5. Click **WBI 6.0 Migration DB2 Universal JDBC Driver Provider** to open it (Figure 16-56). The new DB2 JDBC provider is created in the cell scope.

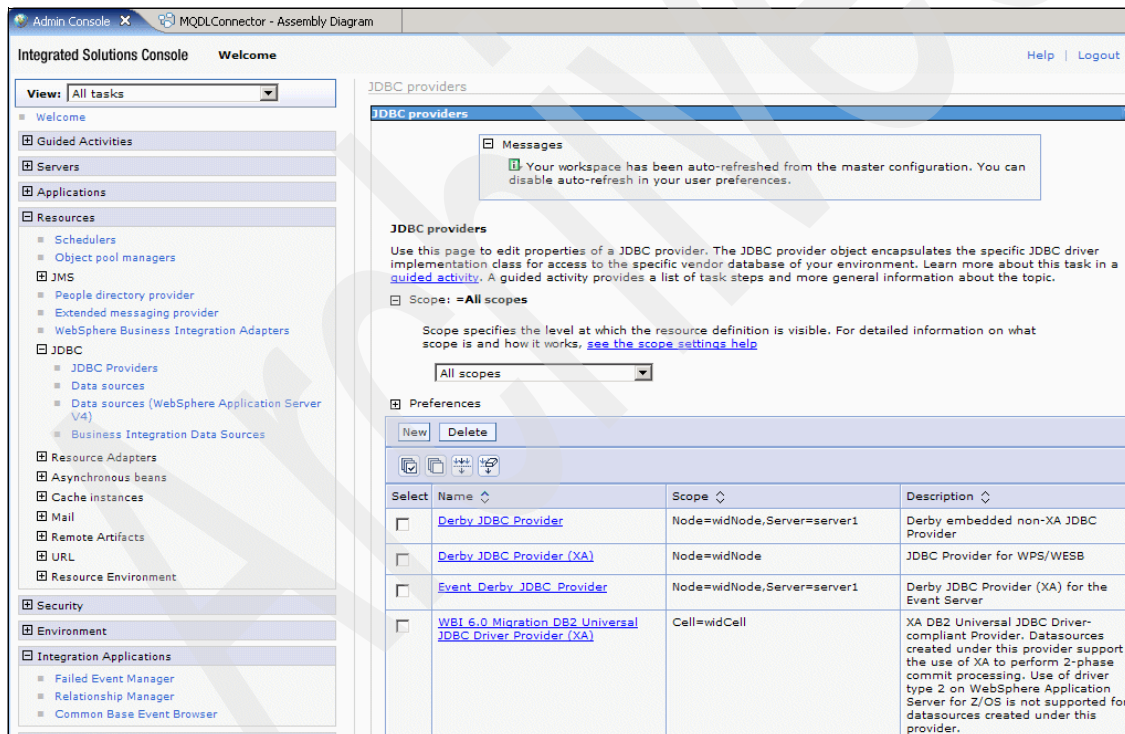


Figure 16-56 JDBC provider created in the cell scope

6. Under Additional Properties, click **data sources**, and open the **Customer** data source (Figure 16-57).

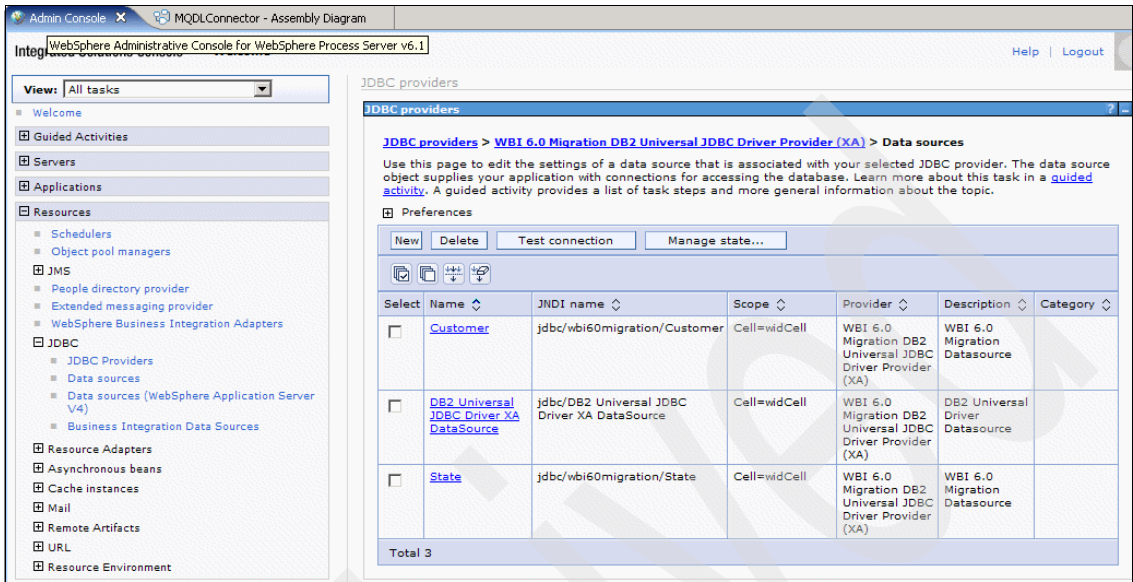


Figure 16-57 Data source created

7. Under DB2 Universal data source properties (Figure 16-58), make sure that Driver type is set to 4.

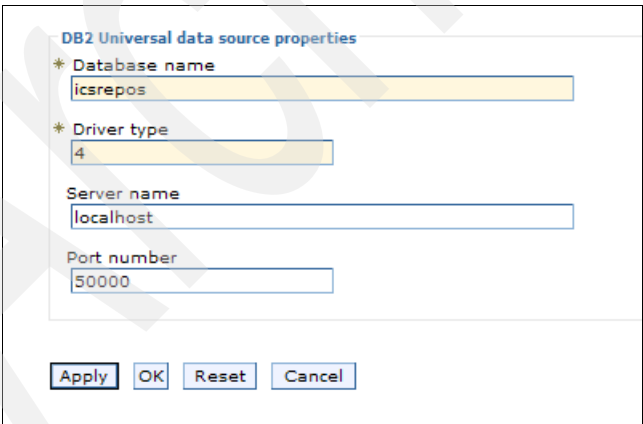


Figure 16-58 Driver type

- Under Data store helper class name (Figure 16-59), confirm that the correct helper class is selected, and save the changes.

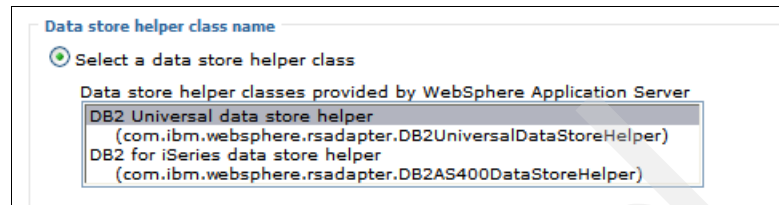


Figure 16-59 Helper class

- Under General Properties for JDBC providers (Figure 16-60), set the correct password for J2C security for the Administrator_Customer and Administrator_State datasource aliases. Save the changes.

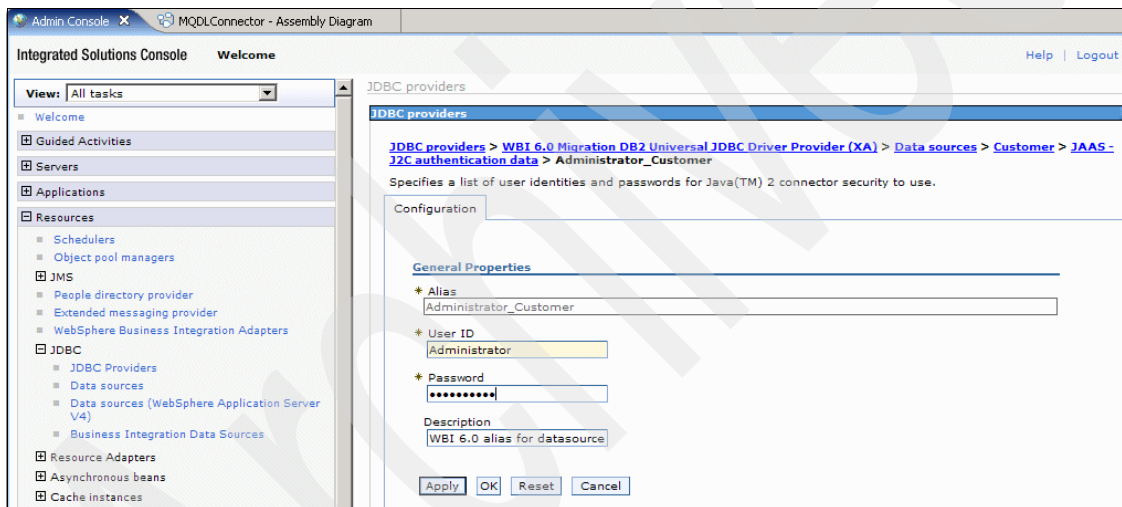


Figure 16-60 Setting the correct password for authentication

10. For the DB2 Universal JDBC Driver XA DataSource data source, set the Component-managed authentication alias to **Administrator_Customer**. For the Database name, type `icsrepos`, and for Server name, type `localhost` (Figure 16-61).

The screenshot shows the 'Admin Console' interface with the 'MQDLConnector - Assembly Diagram' tab selected. The left sidebar displays a tree view of the 'Integrated Solutions Console' with categories like 'Welcome', 'Guided Activities', 'Servers', 'Applications', 'Resources', 'JMS', 'JDBC', 'Resource Adapters', 'Security', 'Environment', 'Integration Applications', 'System administration', 'Users and Groups', 'Monitoring and Tuning', 'Troubleshooting', and 'Service integration'. The 'JDBC' category is expanded, showing 'JDBC Providers', 'Data sources', 'Data sources (WebSphere Application Server V4)', and 'Business Integration Data Sources'. The 'Data sources' sub-category is selected, leading to the 'XA DataSource properties' configuration page. The page has a 'Welcome' header and a 'View: All tasks' dropdown. The main content area contains several sections: 'Specify a user-defined data store helper' (with a text input field), 'Component-managed authentication alias' (with a dropdown menu set to 'Administrator_Customer'), 'Authentication alias for XA recovery' (with a radio button for 'Use component-managed authentication alias' and a 'Specify:' dropdown set to '(none)'), 'Container-managed authentication' (with a dropdown set to '(none)'), 'Mapping-configuration alias' (with a dropdown set to '(none)'), and 'DB2 Universal data source properties' (with fields for 'Database name' set to 'icsrepos', 'Driver type' set to '4', 'Server name' set to 'localhost', and 'Port number' set to '50000'). At the bottom are 'Apply', 'OK', 'Reset', and 'Cancel' buttons.

Figure 16-61 XA DataSource properties

11. Test all of the datasource connections, and verify if they connect successfully as shown in Figure 16-62.

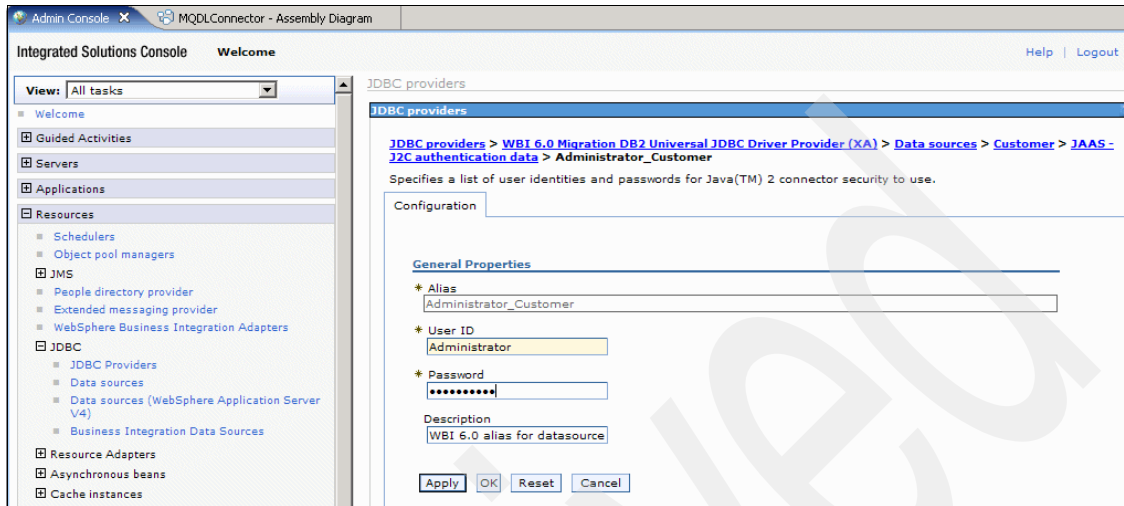


Figure 16-62 Data source connection testing

16.2.6 Deployment

After completing the steps to migrate the data synchronization scenario from WebSphere InterChange Server to WebSphere Integration Developer, the integration system is ready to be deployed to WebSphere Process Server.

Automatic project build: By default, WebSphere Integration Developer is set to build projects automatically. If the workspace is not set to build automatically, run the Clean and Build functions from the Project menu before deploying the modules.

To deploy the components:

1. From WebSphere Integration Developer, stop and start the embedded instance of WebSphere Process Server. Under the **Servers** tab, right-click the instance and select **Start** (Figure 16-63 on page 408).

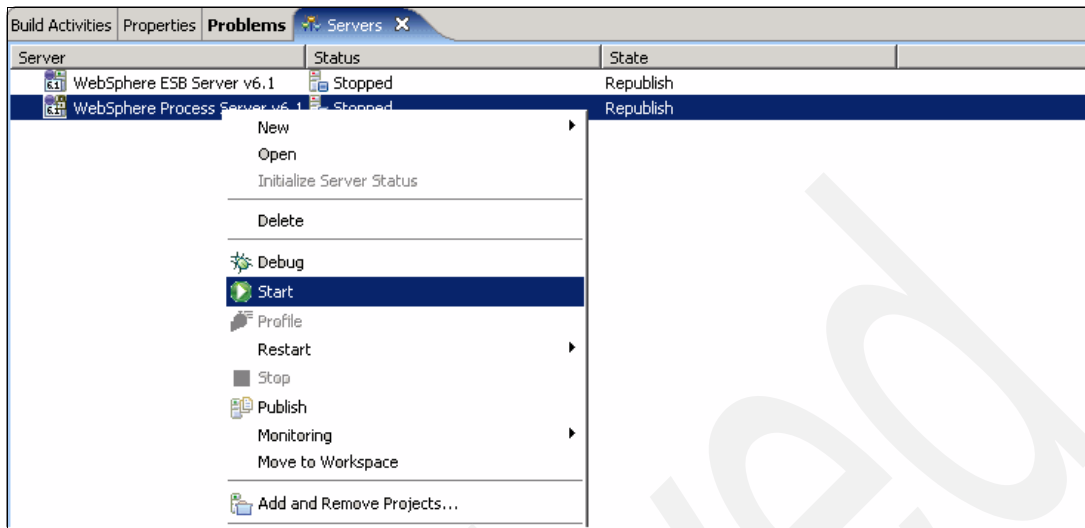


Figure 16-63 Starting WebSphere Process Server

2. Disable the default security settings in WebSphere Process Server to simplify the deployment and testing of the data synchronization scenario:
 - a. Launch the WebSphere Process Server administrative console.
Right-click an embedded instance of WebSphere Process Server, and select **Run** (Figure 16-64). Log in by using the default administrative user and password.

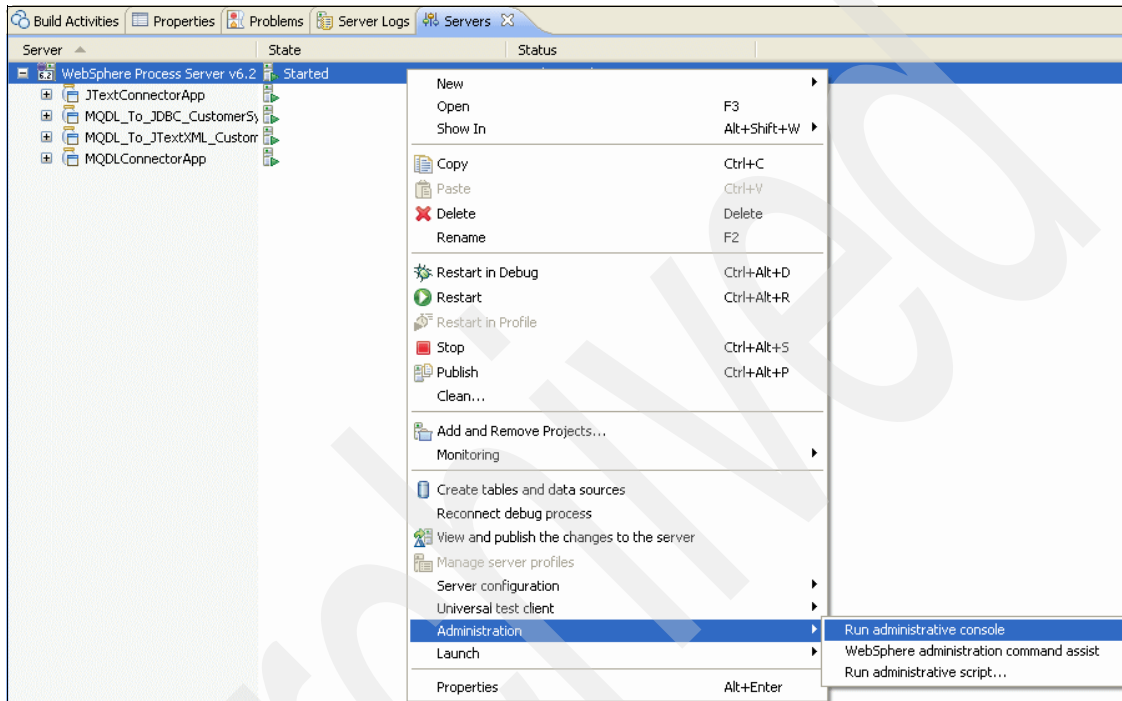


Figure 16-64 Running the administrative console

- b. In the left pane, under Security, choose **Secure administration, applications, and infrastructure**. In the right pane, clear the **Enable administrative security**, **Enable application security**, and **Use Java 2 security to restrict application access to local resources** check boxes (Figure 16-65). Click **Apply** and save the changes.

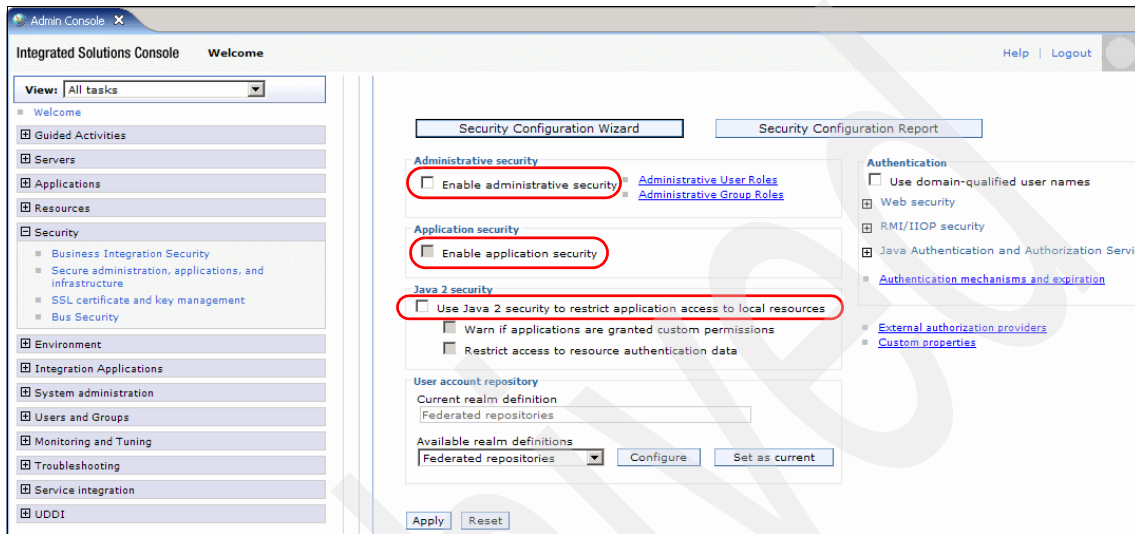


Figure 16-65 Disabling administrative security

- c. In the left pane, under Security, select **Bus Security**. In the right pane, disable the security for all of the buses. Click **Apply**, and save the changes (Figure 16-66).

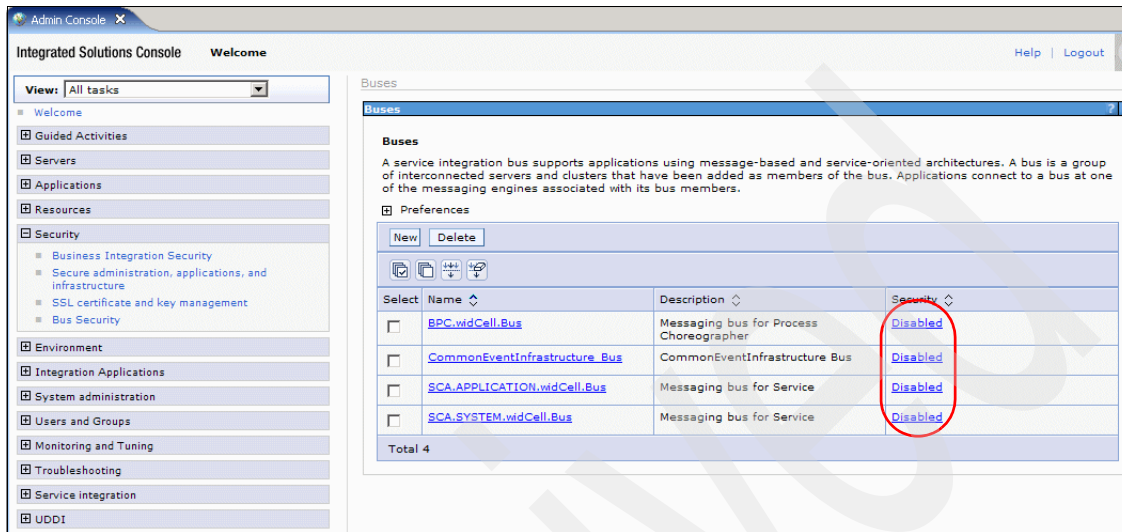


Figure 16-66 Disabling bus security

- d. Log out of the WebSphere Process Server administrative console and close the console window.

- e. Open the profile configuration for WebSphere Process Server. From the **Servers** tab in WebSphere Integration Developer, right-click the appropriate WebSphere Process Server instance, and select **Open** (Figure 16-67).

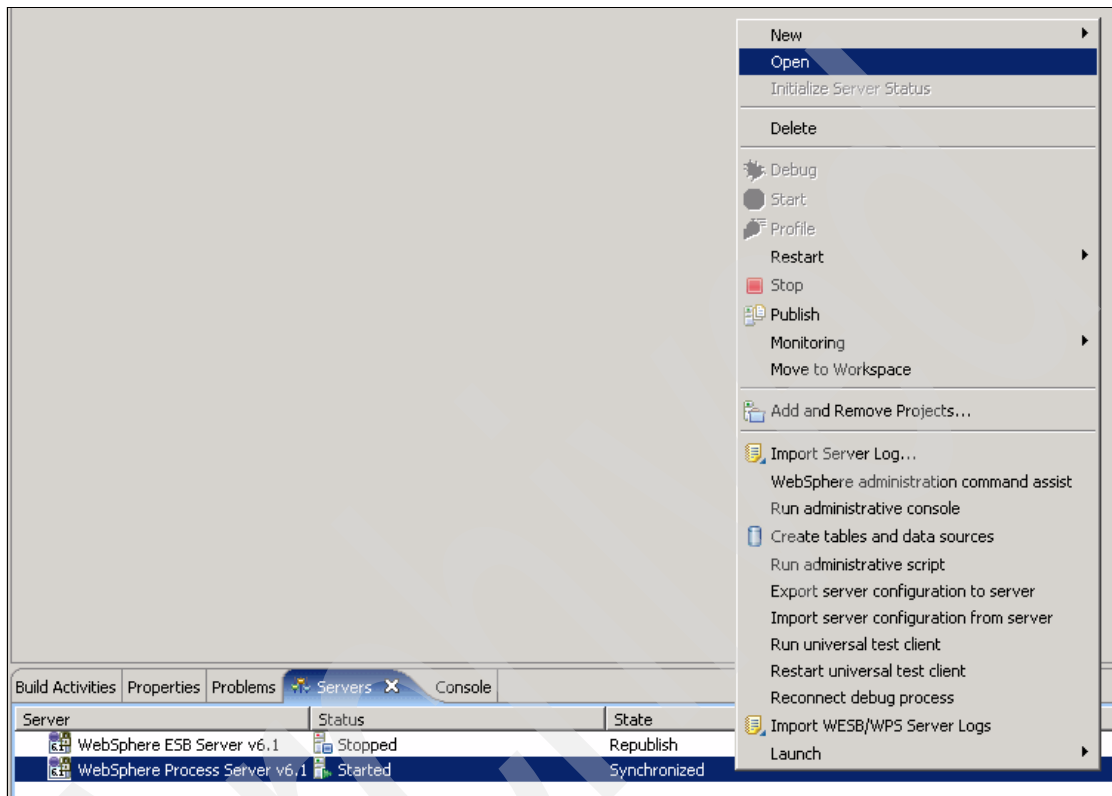


Figure 16-67 Opening the profile configuration

- f. Disable the security settings in the profile configuration. Under the Security section, clear the **Security is enabled on this server** check box (Figure 16-68). Close the profile configuration window, and save the settings.

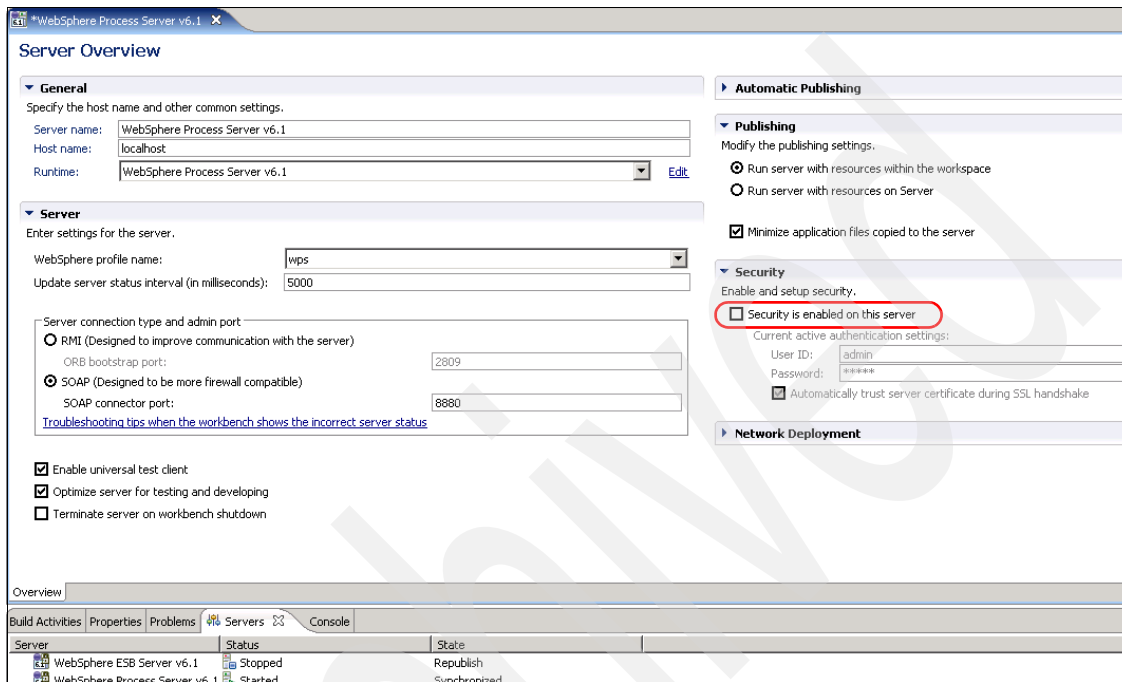


Figure 16-68 Disabling profile security

- g. Restart the embedded instance of WebSphere Process Server (Figure 16-69). From the **Servers** tab, right-click the instance, and select **Restart**.

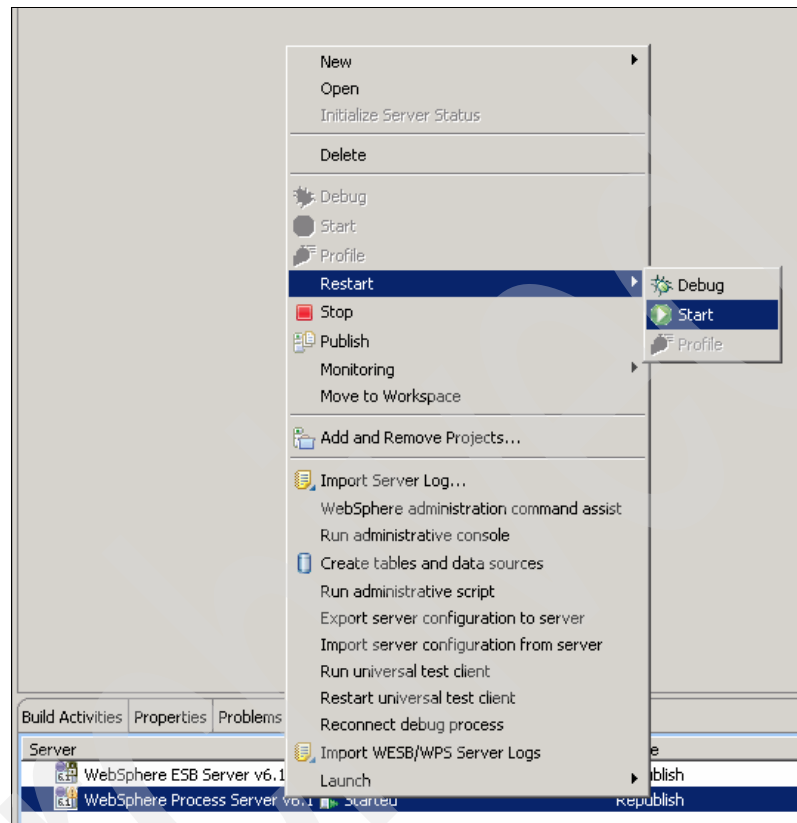


Figure 16-69 Restarting the WebSphere Process Server

3. Deploy the projects in the workspace. From the **Servers** tab in WebSphere Integration Developer, right-click the appropriate WebSphere Process Server instance, and select **Add and Remove Projects** (Figure 16-70).

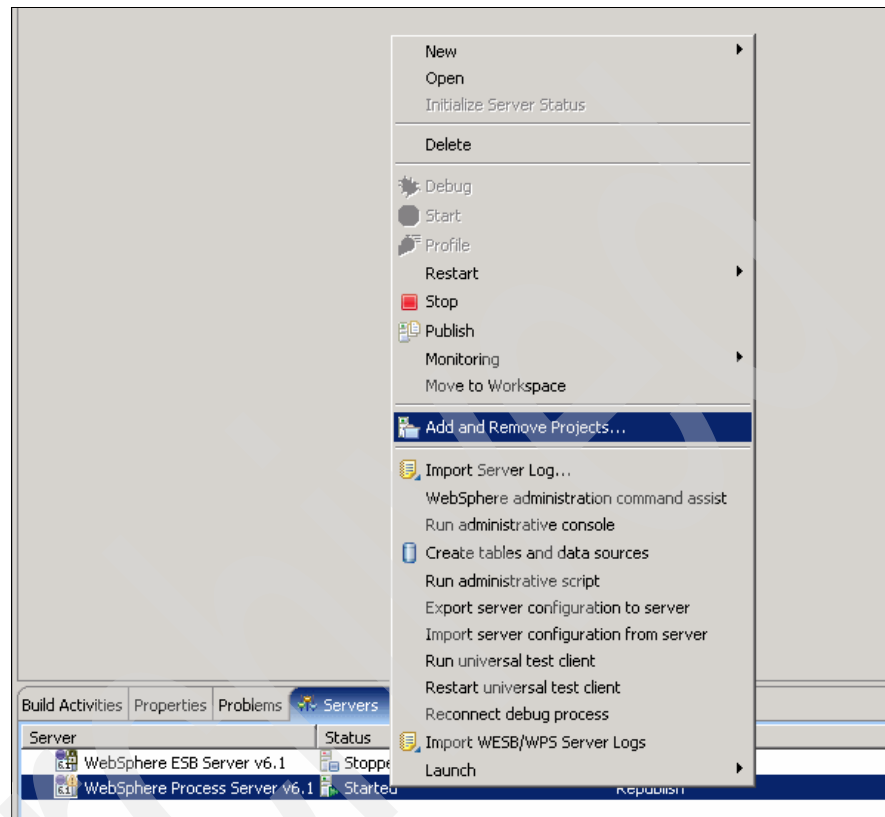


Figure 16-70 Adding projects to WebSphere Process Server

4. In the Add and Remove Projects window (Figure 16-71), click **Add All >>** to add all projects in the workspace to the server. Then, click **Finish** to initiate the deployment.

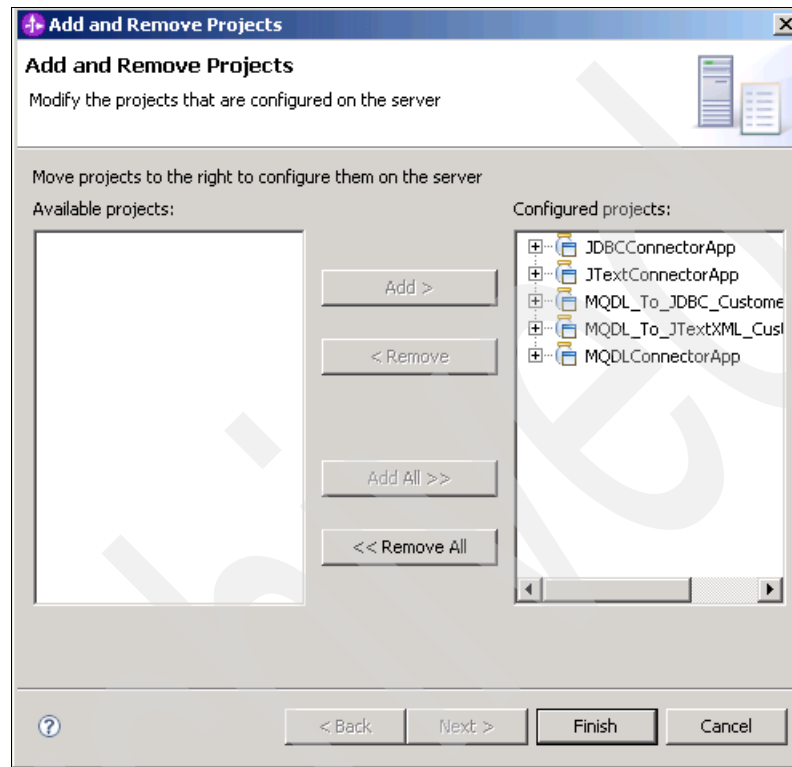


Figure 16-71 Deploying to WebSphere Process Server

5. Verify that all of the modules are deployed and have started correctly in WebSphere Process Server. Expand the embedded WebSphere Process Server instance to show all of the deployed modules. After the deployment has finished, the modules all show a status of Started (Figure 16-72 on page 417).

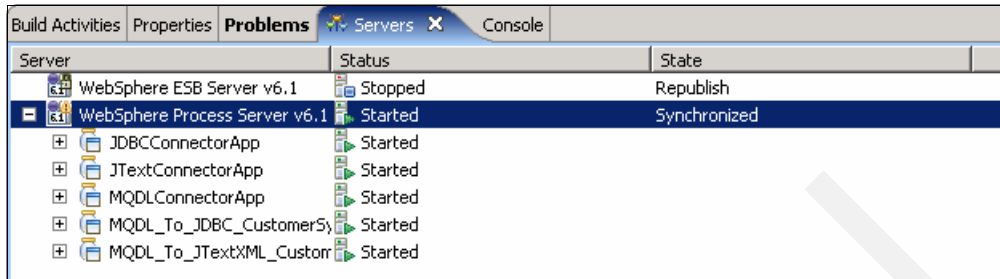


Figure 16-72 Verifying that all modules started

6. Before you test the actual applications, ensure that the relationships have migrated correctly:
 - a. Restart the embedded instance of WebSphere Process Server. From the **Servers** tab, right-click the instance, and select **Restart** → **Start**.
 - b. After the server is started, open the administrative console. On the **Servers** tab, right-click the running server and select **Run administrative console**.
 - c. Click **Log in**. Because security is turned off, there is no need to provide any user ID.
 - d. In the left pane, navigate to **Integration Applications** → **Relationship Manager** → **Relationships**. The two migrated relationships are displayed in the right panel (Figure 16-73).
 - e. Select the **State** relationship, and click **Query**.

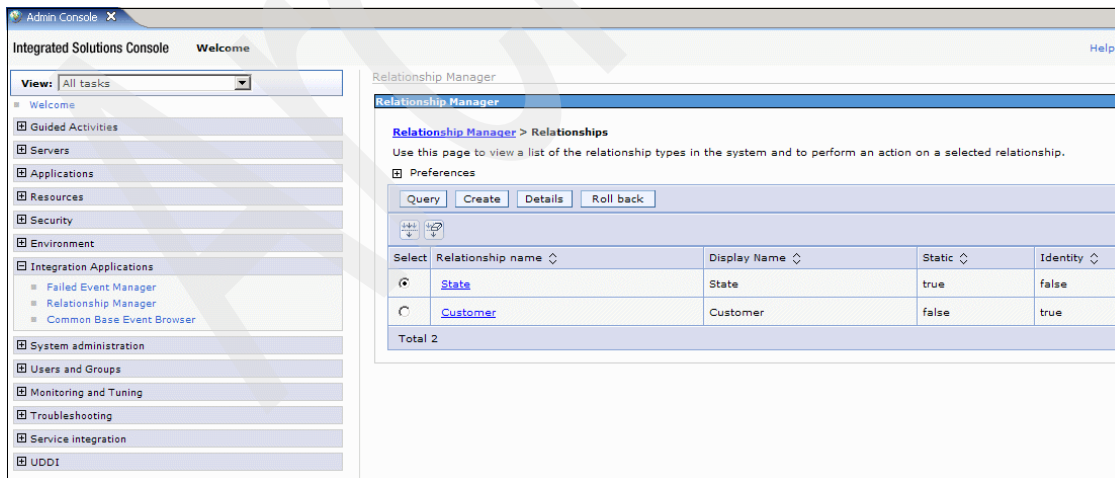


Figure 16-73 Verifying the migrated relationships

- f. In the next window, click **OK**. If state static relationship data was populated as a premigration step in Chapter 14, “Preparation for the technical solutions” on page 279, the query returns all 50 instances as shown in Figure 16-74. If the query returns no rows, it is possible that the relationship data is not populated.

The screenshot shows the Admin Console interface. On the left is a navigation tree with categories like Welcome, Guided Activities, Servers, Applications, Resources, Security, Environment, Integration Applications, System administration, Users and Groups, Monitoring and Tuning, Troubleshooting, Service integration, and UDDI. The 'Relationship Manager' link is highlighted under Integration Applications. The main panel is titled 'Relationship Manager' and shows a breadcrumb trail: 'Relationship Manager > Relationships > Query Relationship: State > Relationship Instances'. Below this is a table of relationship instances. The table has columns for 'Select', 'Relationship instance ID', and 'Property'. The 'Select' column contains radio buttons. The 'Relationship instance ID' column contains numbers 1 through 20. The 'Property' column is empty. At the bottom of the table, it says 'Page: 1 of 3' and 'Total 50'.

Select	Relationship instance ID	Property
<input type="radio"/>	1	
<input type="radio"/>	2	
<input type="radio"/>	3	
<input type="radio"/>	4	
<input type="radio"/>	5	
<input type="radio"/>	6	
<input type="radio"/>	7	
<input type="radio"/>	8	
<input type="radio"/>	9	
<input type="radio"/>	10	
<input type="radio"/>	11	
<input type="radio"/>	12	
<input type="radio"/>	13	
<input type="radio"/>	14	
<input type="radio"/>	15	
<input type="radio"/>	16	
<input type="radio"/>	17	
<input type="radio"/>	18	
<input type="radio"/>	19	
<input type="radio"/>	20	

Figure 16-74 Static relationship instance data in Relationship Manager

7. If relationship data is not populated, download the scripts (STATE_STATECD_T_InsertScript.sql and STATE_STATENM_T_InsertScript.sql), as explained in Appendix D, “Additional material” on page 651, to populate this static relationship. Then, extract the scripts to the c:\setuptemp folder as explained in Chapter 14, “Preparation for the technical solutions” on page 279. Execute the scripts by copying and pasting the content of these files to the DB2 Command Center.

Remember that these scripts only populate the tables. They do not create the tables. For these tables to be created, the WebSphere InterChange Server solution must be deployed to WebSphere InterChange Server successfully. Data can also be populated by using the version of Relationship Manager that comes with WebSphere InterChange Server tools or that is packaged with the WebSphere Process Server administrative console.

If any other errors occur while querying, you might have skipped a migration step and the migrated applications will not work.

Important: You must execute the Jython script prior to deploying the applications to the server the first time. Also, after deploying the applications the first time and before testing the relationships as described previously, remember to restart the server.

16.3 Testing the end-to-end solution

To perform the end-to-end testing as explained in this section, you must have all of the required applications deployed and running on WebSphere Process Server.

Note: Ensure that you have one solution set (applications and adapters) running at a time and that your messages are not consumed by other adapters that are polling the same input queue.

Note: In this sample, we only test Create event. All of the previous implementation steps only cover Create event.

To test the end-to-end solution:

1. Start WebSphere MQ Explorer. The Queue Manager, Channel, Listener, and Queues that are required for this scenario are created as part of the premigration setup in Chapter 14, “Preparation for the technical solutions” on page 279.

2. From the navigator pane (Figure 16-75), right-click and start the **CUST.QUEUE.MANAGER** Queue Manager if it is not already running. Check to ensure that the Listener is running on the port for which the MQDL Connector is configured (1416 in this case).

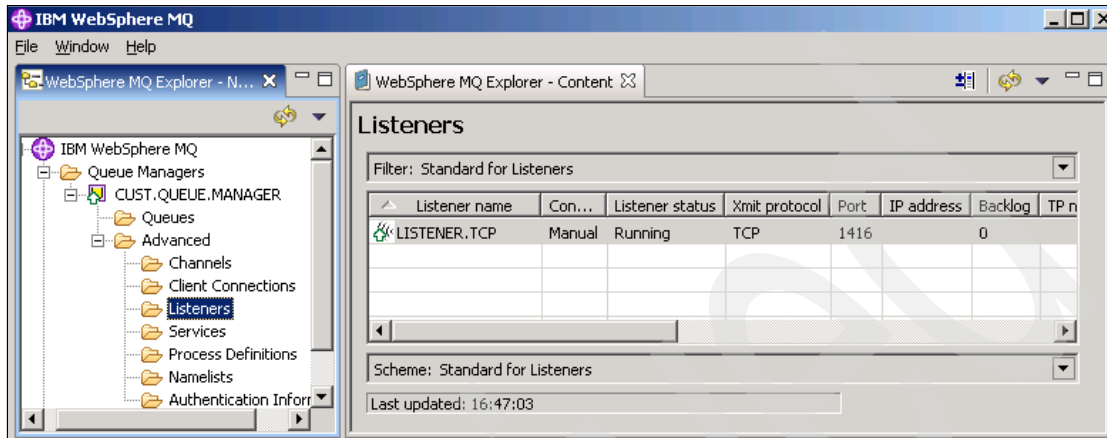


Figure 16-75 WebSphere MQ Explorer

3. To create a new test event:
 - a. Download the utility for testing WebSphere MQ (RfhUtil) from this Web site:
http://www-01.ibm.com/support/docview.wss?rs=0&q1=SupportPac+IH03&uid=swg24000637&loc=en_US&cs=utf-8&cc=us&lang=en
 - b. Extract the downloaded zip file.
 - c. Run rfutil.exe, and then, RfhUtil is started in a pop-up window (Figure 16-76 on page 421).
 - d. Select **CUST.QUEUE.MANAGER** in the Queue Manager Name drop-down list.
 - e. Select **WICS.INPUT.Q** in the Queue Name drop-down list.
 - f. Click **Open File**, and select **MQDLCustomerTestData.txt**, which contains sample test data. This file is available for download as explained in Appendix D, “Additional material” on page 651.

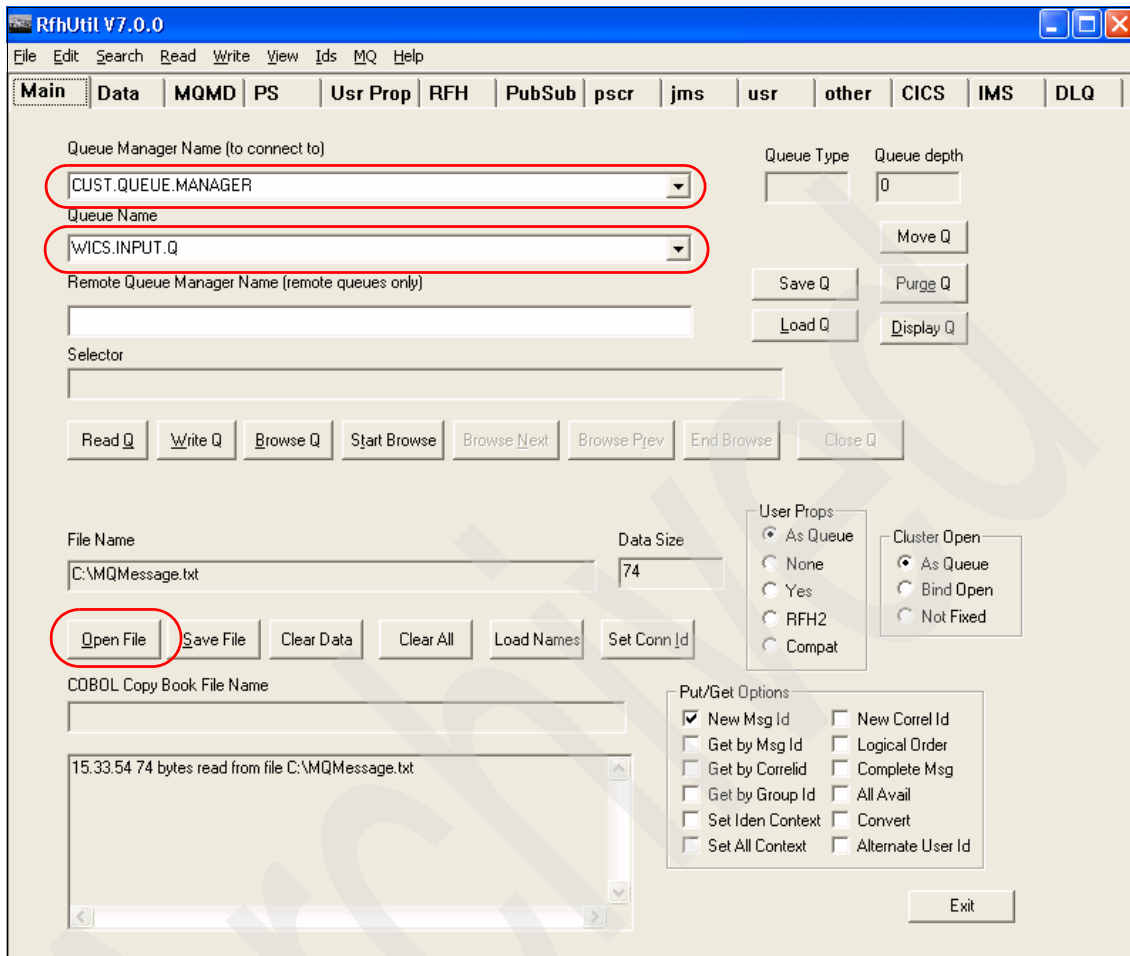


Figure 16-76 RfhUtil

- g. Switch to the **MQMD** tab, and enter CUST_C for the MQ Message Type (Figure 16-77 on page 422).

RfhUtil V7.0.0

File Edit Search Read Write View Ids MQ Help

Main | Data | **MQMD** | PS | Usr Prop | RFH | PubSub | pscr | jms | usr | other | CICS | IMS | DLQ

MQ Message Format: CUST_C User Id: Code Page: 437 Backout Count: 0 Priority: Orig Len: -1

Put Date/Time: Expiry: MsgType: Feedback: Int Fmt: ☒ PC ☐ Host ☐ Unix PD Fmt: ☒ PC ☐ Host

Message ID: Correl ID: Offset: Seq No: 1 Group: ☐ Yes ☐ Last Segment: ☐ Yes ☐ Last ☐ Allowed

Group Id: Application Identity: Id Display: ☐ Ascii ☐ Ebodic ☒ Hex

Appl Origin: Appl Type: Put Application Name: Report Options: Except: ☐ No ☐ Yes ☐ Data ☐ Full Expire: ☐ No ☐ Yes ☐ Data ☐ Full COA: ☐ No ☐ Yes ☐ Data ☐ Full COD: ☐ No ☐ Yes ☐ Data ☐ Full PAN: ☐ PAN ☐ NAN Activity: ☐ Activity ☐ Pass MsgId ☐ Pass Correl Discard: ☐ Discard ☐ Discard/Expiry

Reply To Queue Manager: Reply To Queue: Accounting Token: Persistent Msg: ☒ No ☐ Yes ☐ As Queue Reset Ids Copy Msg Id to Correl Id

Figure 16-77 Set Message Type

- h. Switch back to the **Main** tab in RfhUtil, and click **Write Q**. Then, the message will be put into WICS.INPUT.Q, which means that an event is triggered by MQDLConnector.

4. To verify the success of the test, check the contents of the JDBCCustomer table, the c:\temp\JTextConn\Default\out folder, and the dynamic relationship participant tables:

Remember: Change the test data for every subsequent test that is run, especially the customer ID in the case of a create operation.

- a. Confirm that the row was created in the JDBCCustomer table. Open the **JDBCCUSTOMER** table in the DB2 Control Center (Figure 16-78).

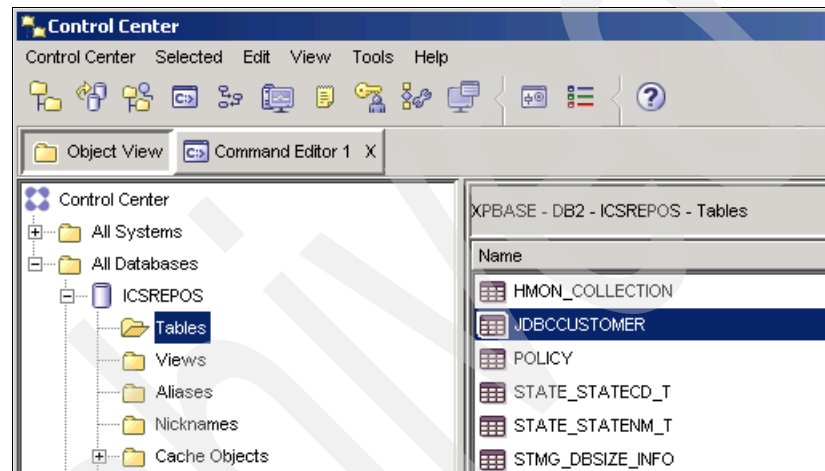


Figure 16-78 JDBCCUSTOMER table in DB2 Control Center

- b. In the Open Table window (Figure 16-79), view the contents of the table.

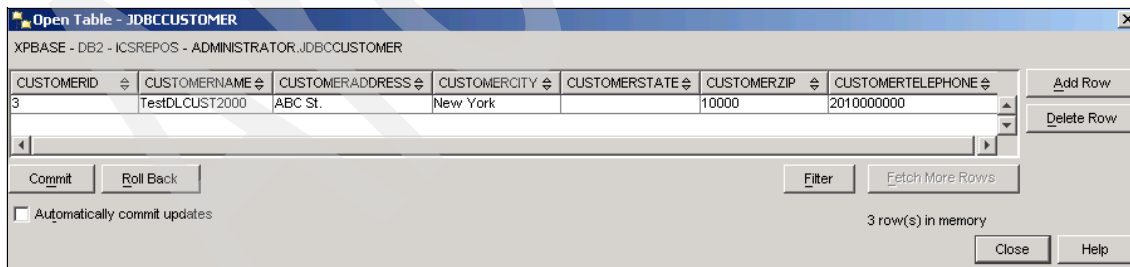


Figure 16-79 JDBCCUSTOMER table data

CUSTOMERID: The CUSTOMERID in the JDBCCUSTOMER table is built by using a DB2 sequence. Therefore, it is different from the generic ID that is used in the business process. The row is inserted into the JDBCCUSTOMER as part of the Create action. The generated CUSTOMERID is returned as part of the response business object and recorded in the relationship instance data.

- c. Confirm that the JTextXMLCustomer_n.out file is created in the C:\temp\JTextConn\Default\out directory (Figure 16-80).

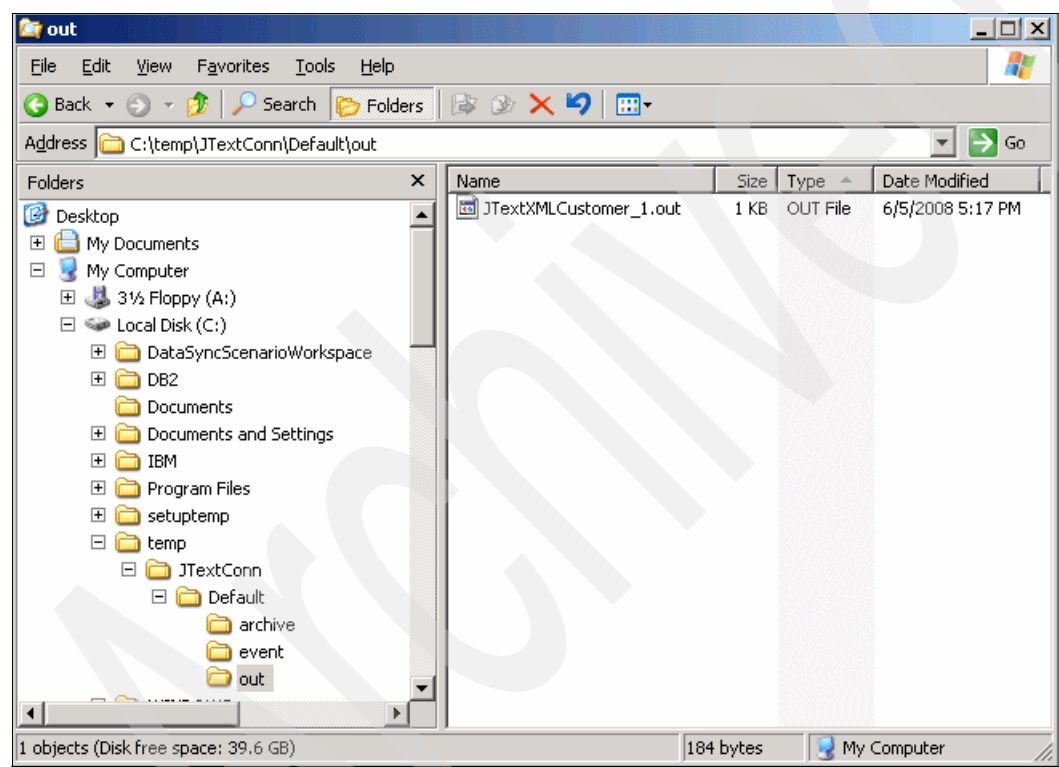


Figure 16-80 Created JTextXMLCustomer output file

- d. Open the file to check its contents (Figure 16-81).

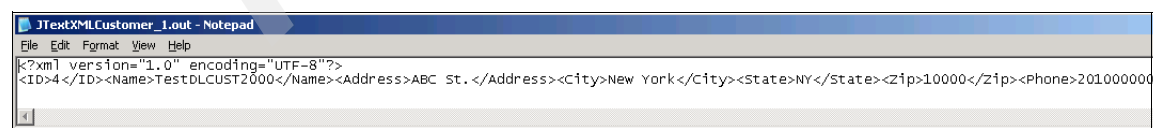


Figure 16-81 Created JTextXMLCustomer output data

Identity relationship: The ID in JTextXMLCustomer_00.out corresponds to the generic ID that passed through the business processes. No identity relationship is used for the WebSphere Adapter for FlatFile interaction.

- e. Confirm that the identity relationship instances are recorded properly:
 - i. Run the administrative console and log in.
 - ii. Navigate to **Integration Applications** → **Relationship Manager**. In the right pane, select **Relationships**. See Figure 16-82.

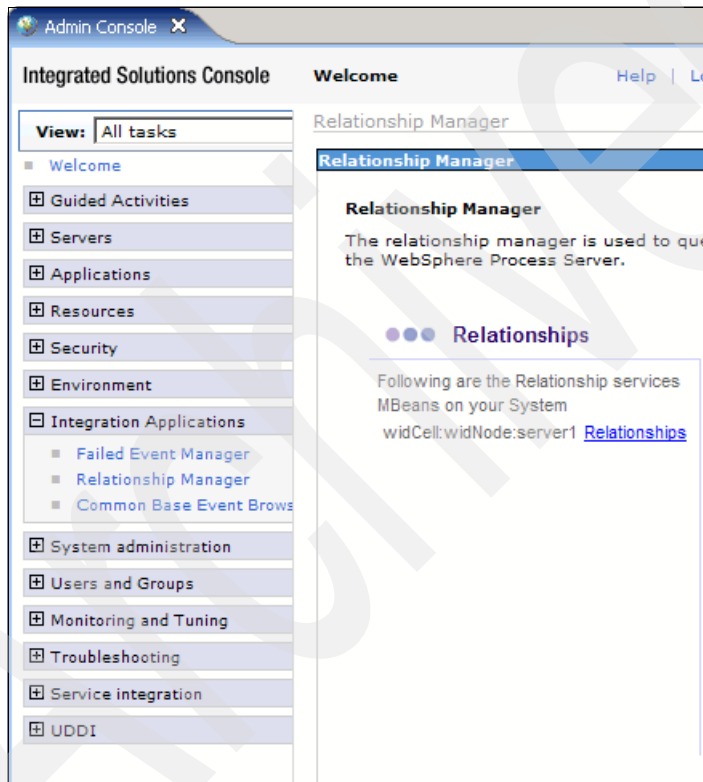


Figure 16-82 Selecting Relationship Manager in the administrative console

- iii. In the Relationships pane (Figure 16-83), select **Customer**, and click **Query**.



Figure 16-83 Querying the customer relationship

- iv. Confirm that the Logical state is set to **All**, and click **OK** (Figure 16-84).

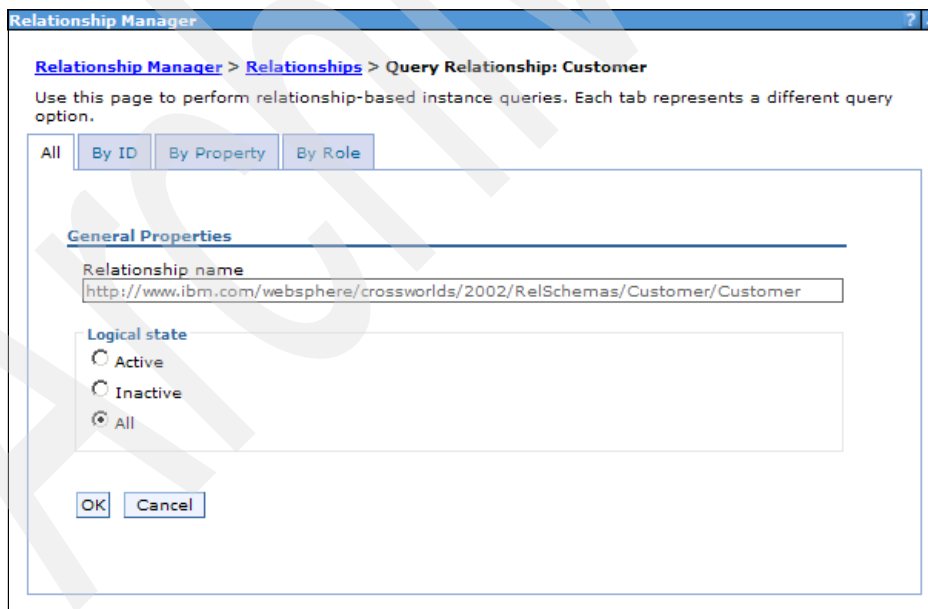


Figure 16-84 Querying all existing customer relationship instances

- v. Identify the generic CustomerID that is used in the console while the business process executed, and select the Relationship instance ID that corresponds to the generic CustomerID (Figure 16-85).

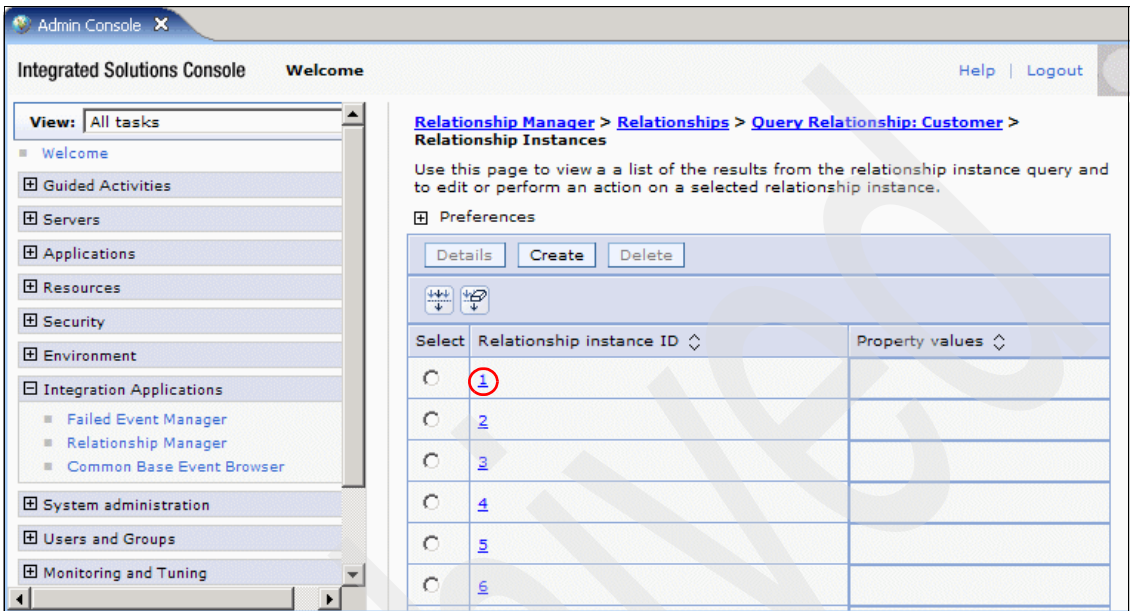


Figure 16-85 Selecting the relationship instance ID created in the test run

- vi. Confirm that the MQDLCust role Key Attribute ID is set to original test data value 2000, which is submitted to WebSphere MQ as shown in Figure 16-86. Also, confirm that the JDBCCust role Key Attribute CUSTOMERID is set to the value created in the JDBCCUSTOMER table. The table is shown in Figure 16-79 on page 423.

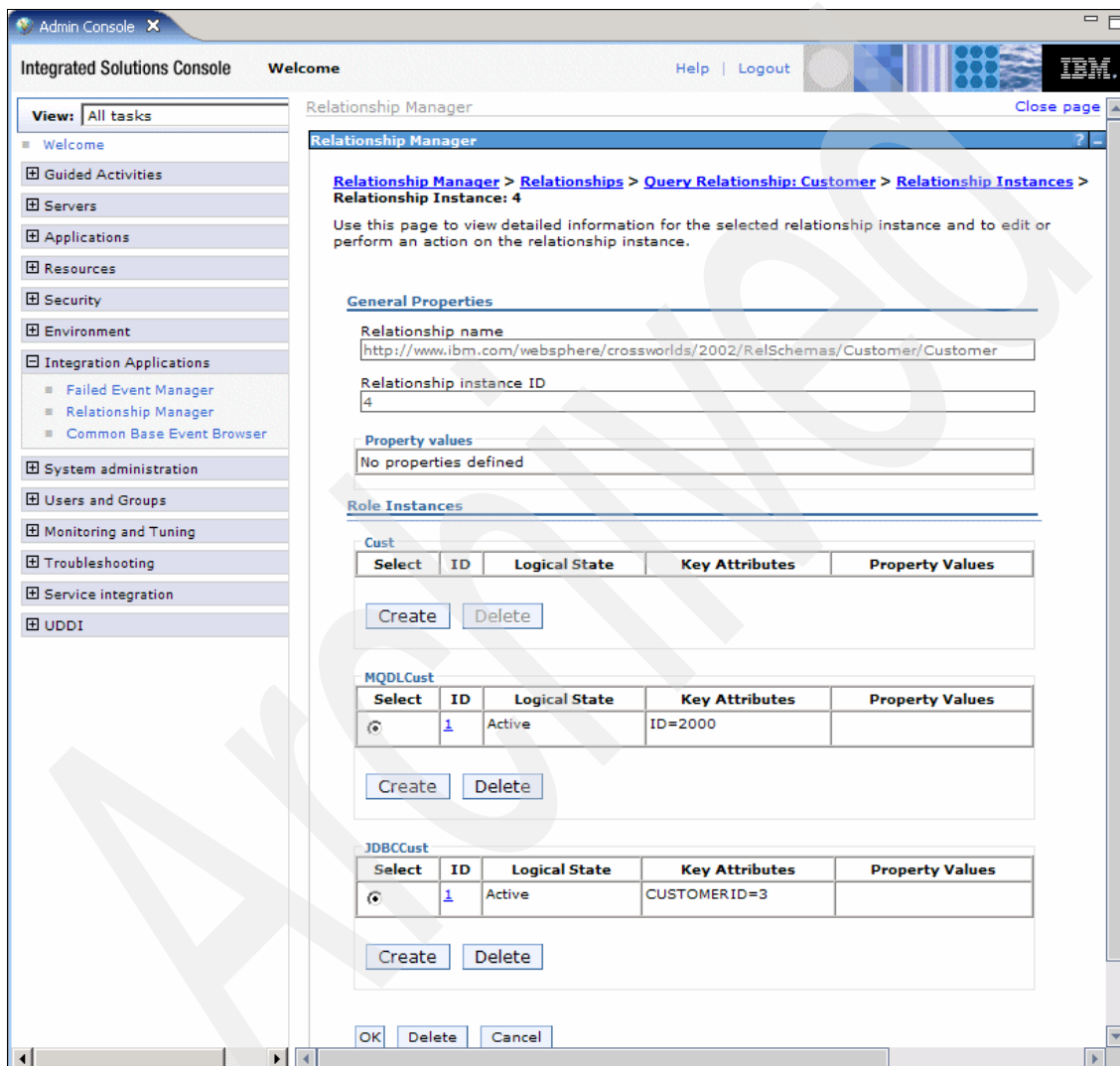


Figure 16-86 Dynamic relationship instance data in Relationship Manager

Note: A valid CustomerID value is required to create a customer in the data synchronization scenario. In addition, the value must be unique. So, subsequent tests must use different CustomerID values.

16.4 Conclusion

In this chapter, we illustrated a step-by-step migration of a data synchronization solution with two collaborations. The migrated WebSphere InterChange Server solution involved different adapters for back-end connectivity. The migration started by using the WebSphere Integration Developer Migration Wizard to get the project working as is in a WebSphere Process Server environment.

We also performed an end-to-end test to check the functionality of the migrated components after each step.

Data synchronization scenario with application adapters

The data synchronization migration scenario outlined in this chapter illustrates an end-to-end scenario in which data entities are synchronized between back-end systems through a typical set of WebSphere InterChange Server components. Together with WebSphere InterChange Server, these components form an integration system.

This chapter focuses on the steps that are necessary to migrate an entire data synchronization integration system that was originally designed for WebSphere InterChange Server to WebSphere Process Server. By performing the steps illustrated in this chapter, an end-to-end data synchronization scenario can be completely migrated to WebSphere Process Server, optimized, and tested.

The data synchronization scenario uses two WebSphere Business Integration Adapters: the WebSphere Business Integration Adapter for mySAP.com and the WebSphere Business Integration Adapter for PeopleSoft. In addition, the scenario uses business objects, maps, relationships, and collaborations.

The scenario in this chapter simulates an integration system that is designed to provide synchronization between two back-end systems:

- ▶ An Enterprise Resource Planning (ERP) system delivering the customer information from one sales order through SAP
- ▶ An Enterprise Resource Planning (ERP) system that receives the customer information through PeopleSoft

Data synchronization is a common WebSphere InterChange Server pattern where two systems exchange data by using business objects, supporting the four basic verbs or actions of create, retrieve, update, and delete. In WebSphere InterChange Server, an integration specialist uses a collaboration template that synchronizes data between System A and System B. Currently, System A and System B are both configured as ERP systems.

The data synchronization scenario is completely independent of the other examples and scenarios in this book. However, many of the concepts and steps that are necessary to migrate individual artifacts are the same and are used here to illustrate the migration of a complete end-to-end scenario.

As a reminder, see Part 2, “Migration implementation concepts” on page 71, for an overall discussion about the possible options for migration implementation, and Part 3, “Migration tooling” on page 177, provides detailed information about the standard migration tools.

We include the following sections:

- ▶ 17.1, “Target environment” on page 433
- ▶ 17.2, “Implementation” on page 434
- ▶ 17.3, “Testing the end-to-end solution” on page 497
- ▶ 17.4, “Conclusion” on page 509

17.1 Target environment

In this chapter, we illustrate the migration of an end-to-end scenario from WebSphere InterChange Server to WebSphere Process Server. The WebSphere InterChange Server development was done with the product versions that are listed in Table 17-1. It was migrated step-by-step as shown in this chapter to run on the WebSphere Process Server Version 6.2.

17.1.1 Software environment

Table 17-1 lists the software products and versions that we used to demonstrate the example in this chapter.

Note: SAP ALE inbound is supported by WebSphere Integration Developer Version 6.2's latest fix pack. You need to apply this fix pack first.

Table 17-1 Software versions used

Software	Version
WebSphere Integration Developer	6.2
WebSphere Process Server	6.2
WebSphere MQ	6.0.2 Fix Pack 4
WebSphere InterChange Server	4.3.0 Fix Pack 5
DB2	8.1.16 (or 8.2.9)
WebSphere Business Integration Adapter Framework	2.6.0.12
WebSphere Business Integration Adapter for mySAP.com	6.0.8
WebSphere Business Integration Adapter for PeopleSoft	3.0.1
WebSphere Adapter for PeopleSoft	6.2
WebSphere Adapter for SAP Software	6.2
WebSphere Adapter Migration Tool	6.2
SAP Front-End GUI Tool	640 patch 14

17.2 Implementation

In this section, we describe the process of migrating the data synchronization scenario to WebSphere Integration Developer.

17.2.1 Premigration overview

The data synchronization scenario is a functional implementation of the data synchronization pattern that is typically implemented in WebSphere InterChange Server. This scenario is restricted to the following collaboration that is based on the same collaboration template: SAPJMS_To_PSFT_CustomerSync.

The SAPJMS_To_PSFT_CustomerSync collaboration synchronizes the customer information of sales order data from a SAP system that publishes changes (create, update, or delete) to a PeopleSoft system. This collaboration has a single inbound port (From) and a single outbound port (To). The From port is bound to an instance of the WebSphere Business Integration Adapter for mySAP.com. The To port is bound to an instance of the WebSphere Business Integration Adapter for PeopleSoft as illustrated in Figure 17-1.

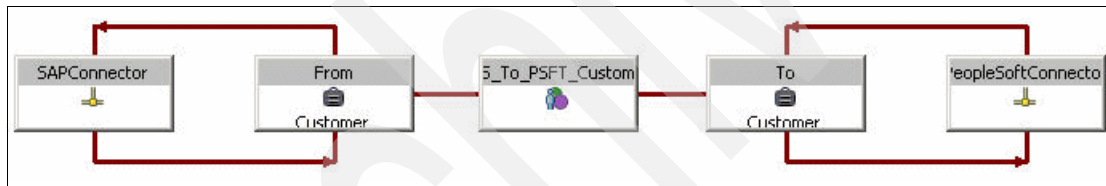


Figure 17-1 SAPJMS_To_PSFT_CustomerSync collaboration object

This scenario demonstrates the commonly used publish/subscribe integration pattern. A single published event is delivered to the subscribing collaborations, which in turn, deliver it to the back-end system.

17.2.2 Preparation

The goal of this chapter is to migrate an existing WebSphere InterChange Server example and test it in the WebSphere Process Server. As a prerequisite, we show how to test the DataSyncScenario example in WebSphere InterChange Server. Then, we explain the details of the migration of the example to WebSphere Process Server.

We begin by explaining the initial steps that you must complete in order to migrate the DataSyncScenario example to WebSphere Process Server.

Files for download: You can download all of the files that are mentioned in this section as explained in Appendix D, “Additional material” on page 651.

Testing the DataSyncScenario example on WebSphere InterChange Server V4.3

To test the DataSyncScenario example in the WebSphere InterChange Server:

1. Create a new Interchange component library, DataSyncICL, in the WebSphere InterChange Server System Manager (Figure 17-2 on page 436). Import the DataSyncScenarioWithSAPAndPSFT.jar file into it. The Java archive (JAR) file is in the c:\setuptemp\Chapter17samples directory. Right-click the **DataSyncICL** project, and select **Import from repository file**.
2. Make sure that the ERP system configuration information in SAPConnector is consistent with the ERP system that you use.
3. Make sure that the ERP system configuration information in PeopleSoftConnector is consistent with the ERP system that you use.

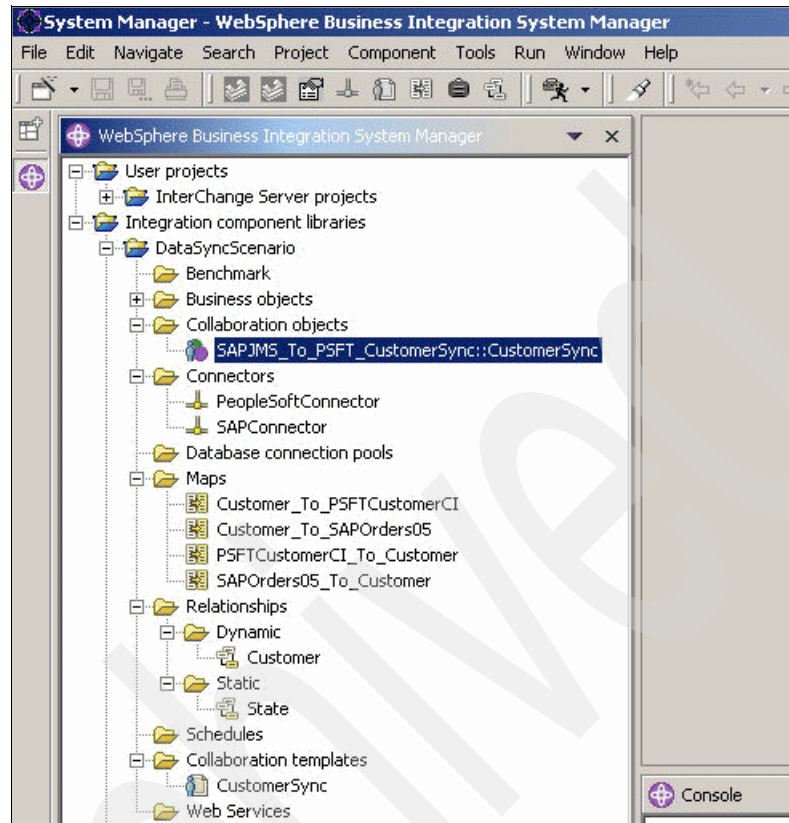


Figure 17-2 *DataSyncScenario* project as viewed in System Manager

4. Create a user project, called *DataSyncScenario*, in the System Manager:
 - a. Add all of the components from the newly created DataSyncICL project by selecting **DataSyncICL**.
 - b. Connect to the WebSphere InterChange Server instance.
 - c. Validate the project.
 - d. Right-click **DataSyncScenario**, and select **Deploy user project**.
 - e. In the Deploy wizard window, expand the **DataSyncScenario** project and select **Business Objects**. Select **Create schema**.

5. Confirm that the Customer and State Relationships are deployed successfully onto the WebSphere InterChange Server instance. Expand **InterChange Server instance** → **Relationships** → **Dynamic** → **Customer**. Under Relationships, also expand **Static** → **State**. The relationships show a green arrow to their left that confirms that the relationships have deployed successfully (Figure 17-3).

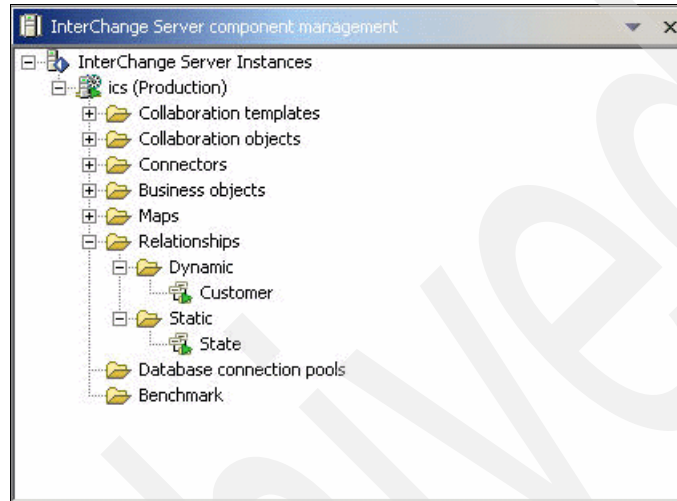


Figure 17-3 Relationships successfully deployed to WebSphere InterChange Server

6. Switch back to the DB2 Control Center. Right-click the **ICSREPOS** database, and select **Query**.
7. Insert the static Relationship instances by executing the SQL scripts for the State Static Relationship:
 - a. Click the Scripts **Open** button.
 - b. Browse to select the **C:\setuptemp\Chapter17samples\STATE_STATE_CD_T_InsertScript.sql** file.
 - c. Click the green arrow button to execute the script.
 - d. Repeat the previous three steps to execute the **STATE_STATENM_T_InsertScript.sql** script.
 - e. If you see a DB2 message to replace the current input with the contents, click **Yes**, and execute.

17.2.3 Migration

In this section, we explain how to migrate the data synchronization scenario to WebSphere Integration Developer.

The `ICSDataSyncScenarioWithSAPAndPSFT.jar` file, which you extract to the `c:\setuptemp` directory, is the WebSphere InterChange Server Repository file.

Exporting a WebSphere InterChange Server Repository file: From the WebSphere Business Integration System Manager, select an appropriate integration component library, right-click it, and select **Export as** → **Repository File**.

To migrate the data synchronization scenario to WebSphere Integration Developer:

1. Launch WebSphere Integration Developer with a new workspace called *DataSyncScenarioWorkspace*:
 - a. In WebSphere Integration Developer, switch to the **Business Integration** perspective.
 - b. Select **File** → **New** → **Library** to create a new library.
 - c. In the New Library window (Figure 17-4), for Library name, type `DataSyncScenarioLibrary` to create a new library in the workspace. Click **Finish**.

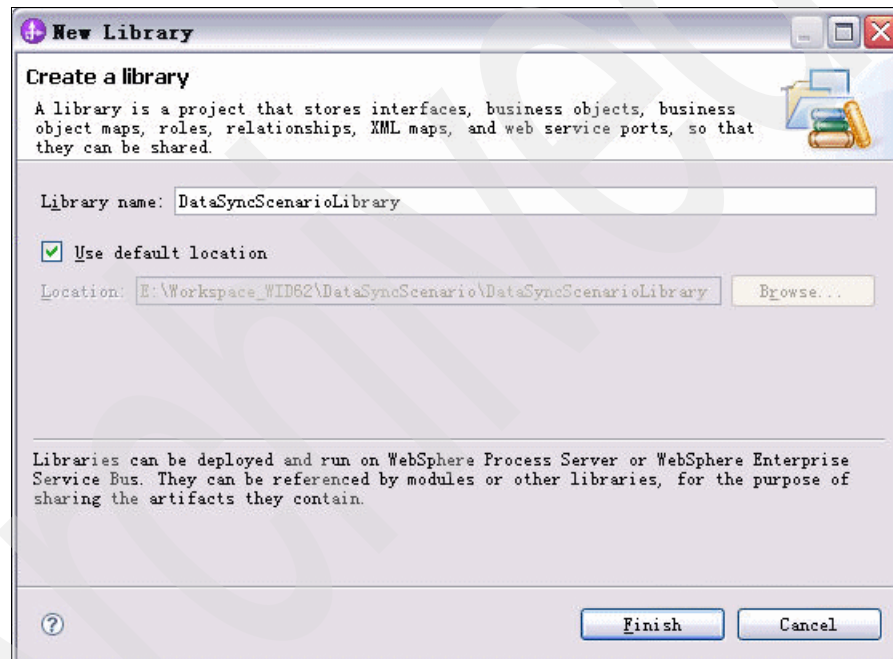


Figure 17-4 New data synchronization scenario library

2. Import the WebSphere InterChange Server Repository file named ICSDDataSyncScenarioWithSAPAndPSFT.jar by using the WebSphere InterChange Server Migration Wizard in WebSphere Integration Developer:
 - a. Select **File** → **Import**.
 - b. In the Import: Select window (Figure 17-5), under the Business Integration folder, select the **WebSphere InterChange Server Repository** JAR File. Click **Next**.

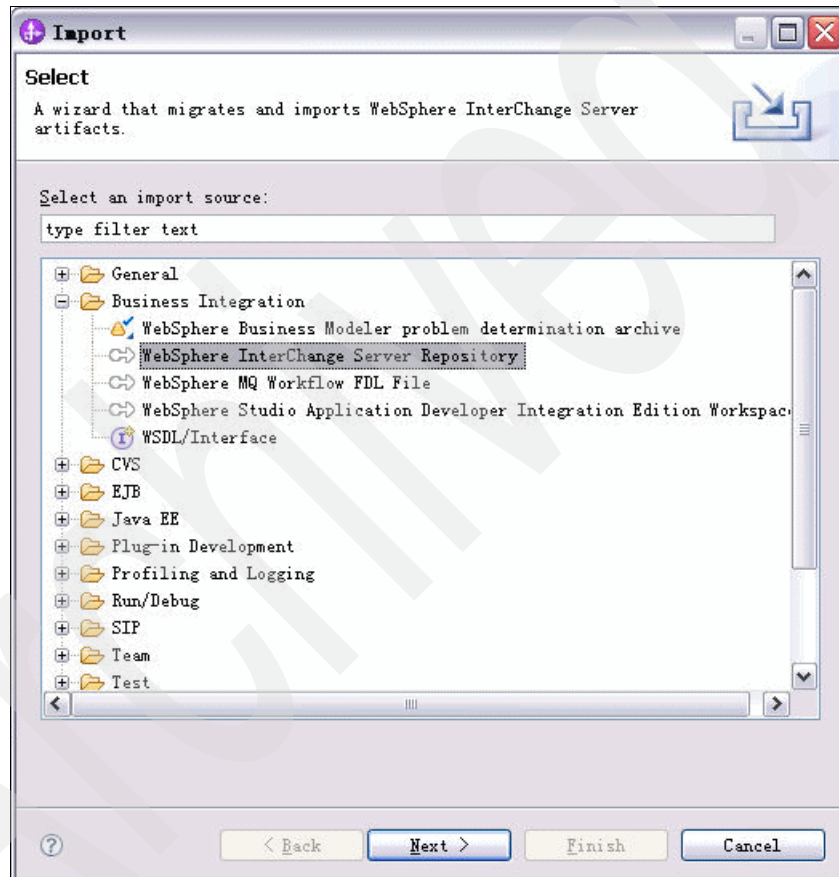


Figure 17-5 Import wizard: Selecting the WebSphere InterChange Server Repository JAR File

- c. In the Import Wizard: Select WebSphere InterChange Repository Details window (Figure 17-6), for the repository path, select the **DataSyncScenarioWithSAPAndPSFT.jar** file, and for the library, select **DataSyncScenarioLibrary**. Click **Next**.

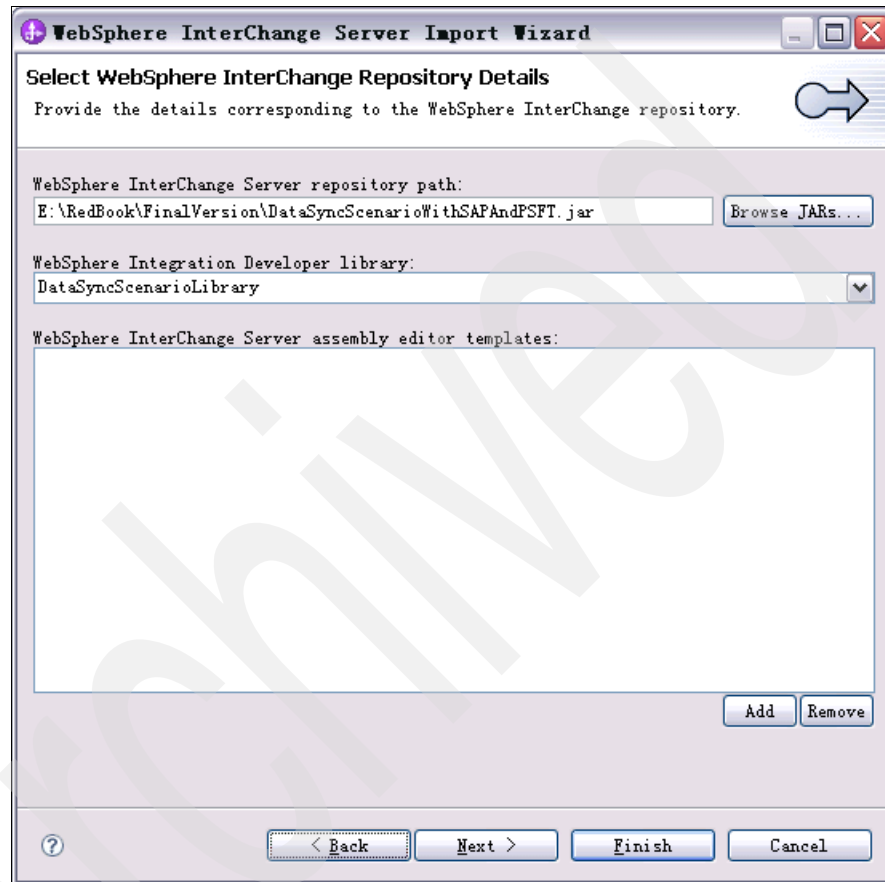


Figure 17-6 Import Wizard: Select WebSphere InterChange Repository Details

- d. In the Import Wizard: Configure Connector Migration window, select **PeopleSoftConnector** in the left panel. In the Binding field, choose **JMS to PeopleSoft WBI Adapter** in the drop-down list (Figure 17-7).

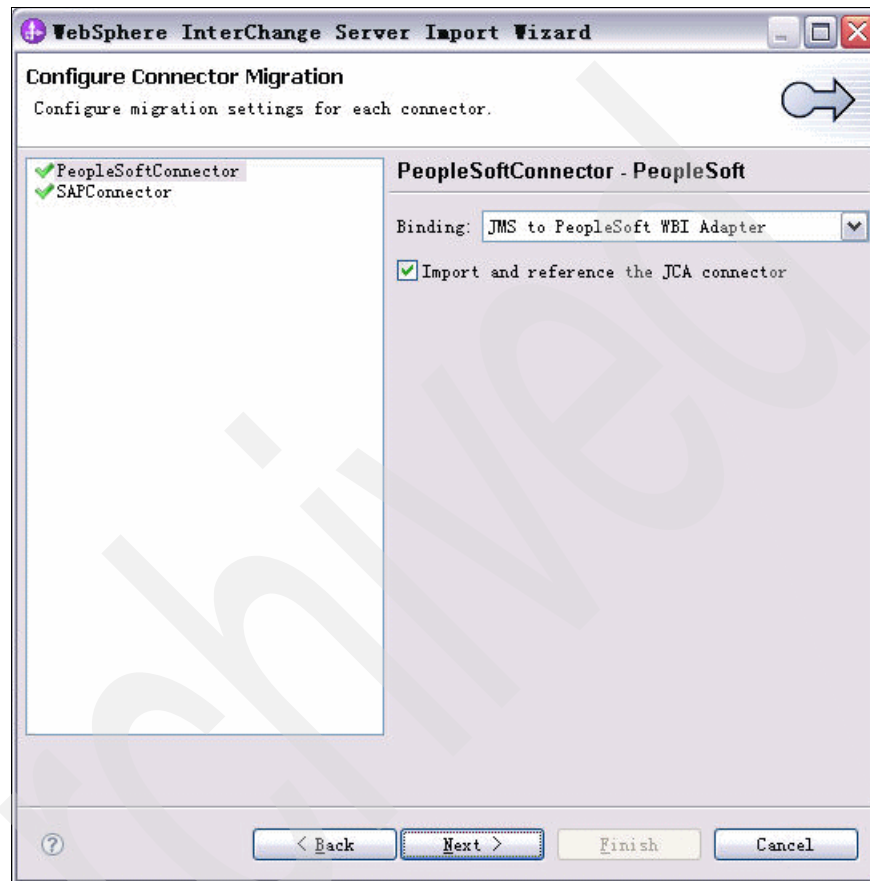


Figure 17-7 Import Wizard Configure Connector Migration: PeopleSoftConnector

- e. In the same window, select **SAPConnector** in the left panel. In the Binding field, choose **JMS to SAP WBI Adapter** in the drop-down list (Figure 17-8).

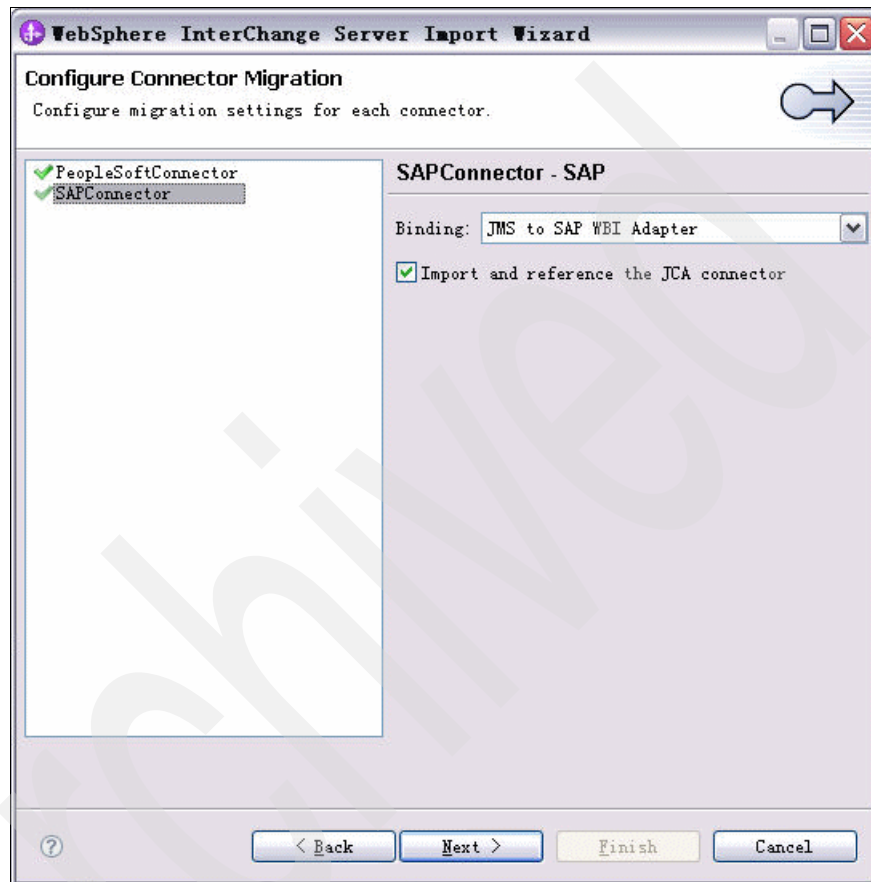


Figure 17-8 Import Wizard Configure Connector Migration: SAPConnector

- f. In the Import Wizard: Conversion Options window, select the options as shown in Figure 17-9 on page 444, and click **Next**.

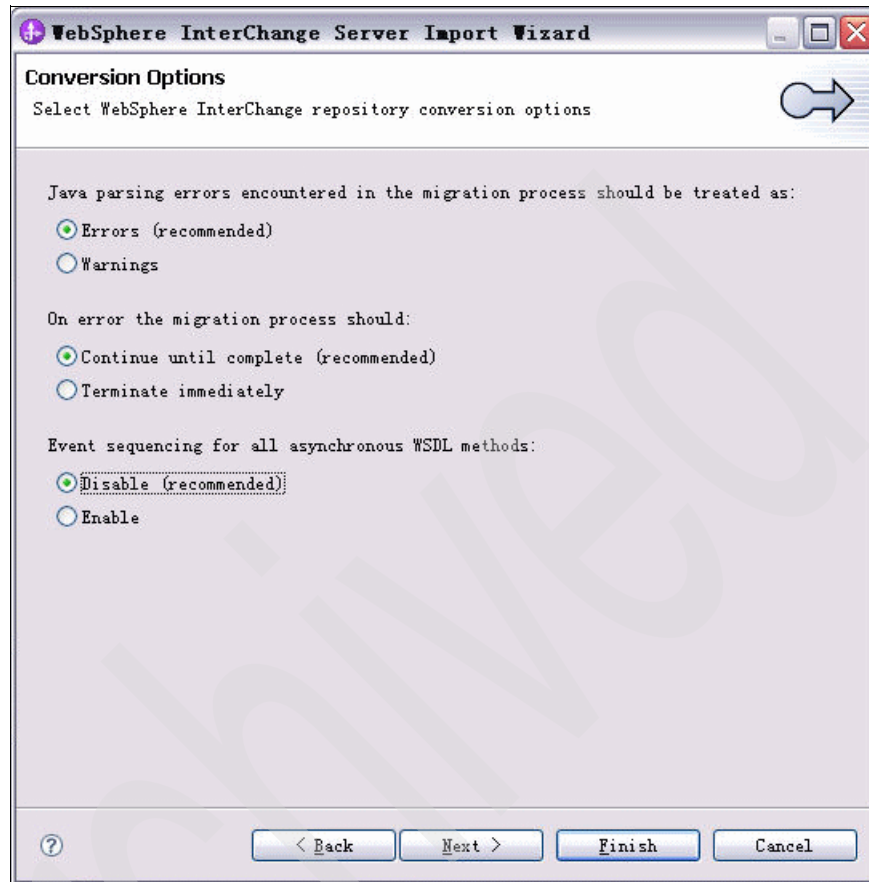


Figure 17-9 Import Wizard: Conversion Options

- g. In the Import Wizard: Migration Summary window, confirm that all of the information is correct (Figure 17-10 on page 445), and click **Finish**.

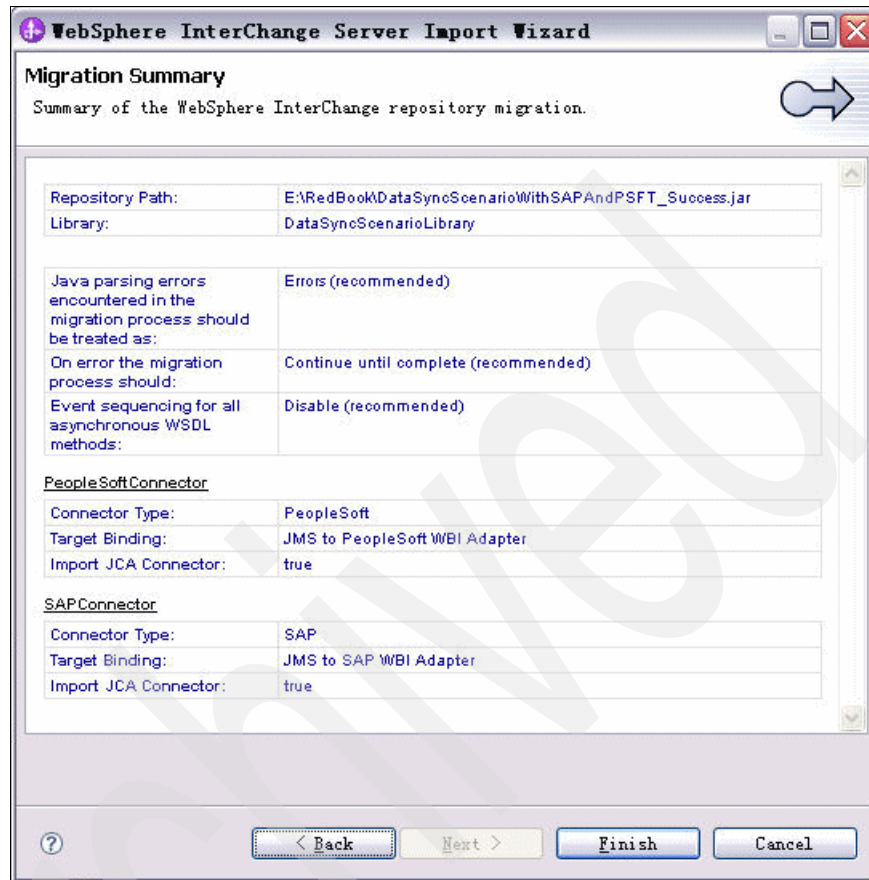


Figure 17-10 Import Wizard: Migration Summary

- h. Wait until the migration process completes. When the Migration Results window is shown (Figure 17-11 on page 446), click **Close**.

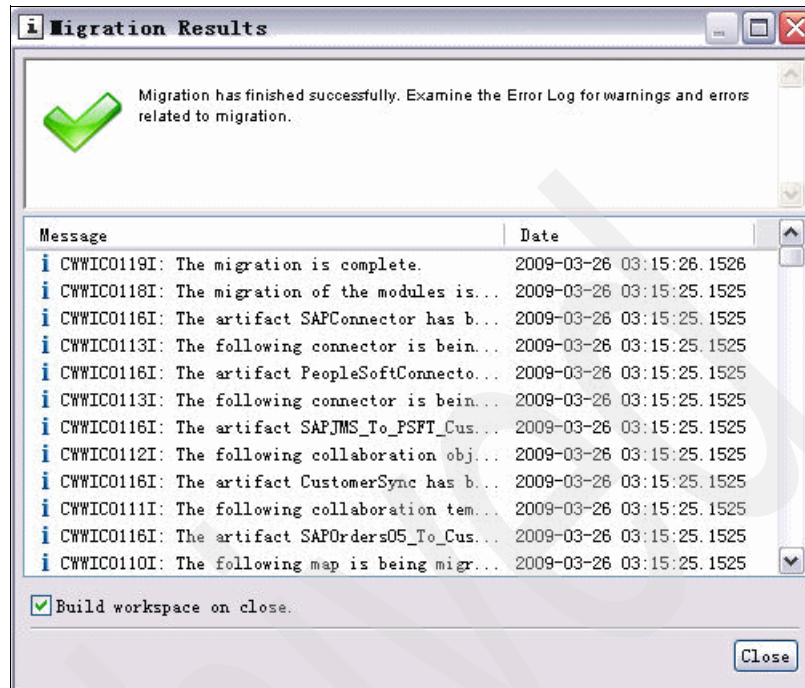


Figure 17-11 Import Wizard: Migration Results

- i. The building of projects in WebSphere Integration Developer can take a while. After the build completes, the workspace contains five new modules in addition to the DataSyncScenarioLibrary that was previously created (Figure 17-12 on page 447). Two of the new modules correspond to two WebSphere Business Integration Adapters that are used in this scenario: SAPConnector and PeopleSoftConnector. One new module corresponds to the collaboration from WebSphere InterChange Server SAPJMS_To_PSFT_CustomerSync. The other two modules, CWAP_SAPAdapter_Tx and CWES_PeopleSoft, are resource files for WebSphere Adapter for SAP Software and WebSphere Adapter for PeopleSoft. We will update WebSphere Business Integration Adapters to WebSphere Adapters in following steps.

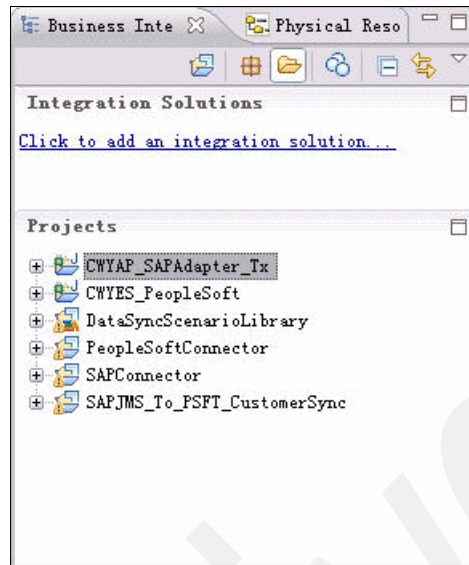


Figure 17-12 Module created

3. Because of a unique situation in the WebSphere Adapter Migration Tool, we need to copy the Customer business object (BO) file from **DataSyncScenarioLibrary** → **Data Types** to a temporary directory, for example, c:\temp (Figure 17-13 on page 448). After the WebSphere Business Integration Adapter for mySAP.com update, you need copy the Customer BO file back.

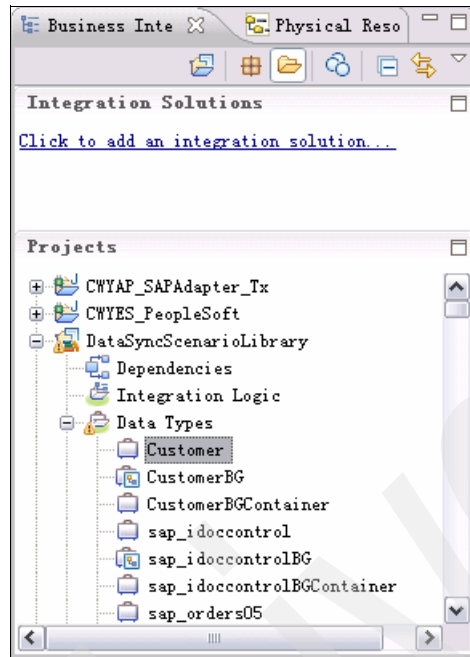


Figure 17-13 Copy Customer BO

4. Upgrade WebSphere Business Integration Adapter for mySAP.com to WebSphere Adapter for SAP Software:
 - a. In the Business Integration perspective, right-click the **SAPConnector** module. Select **Update** → **Migrate Adapter Artifacts** (Figure 17-14 on page 449).

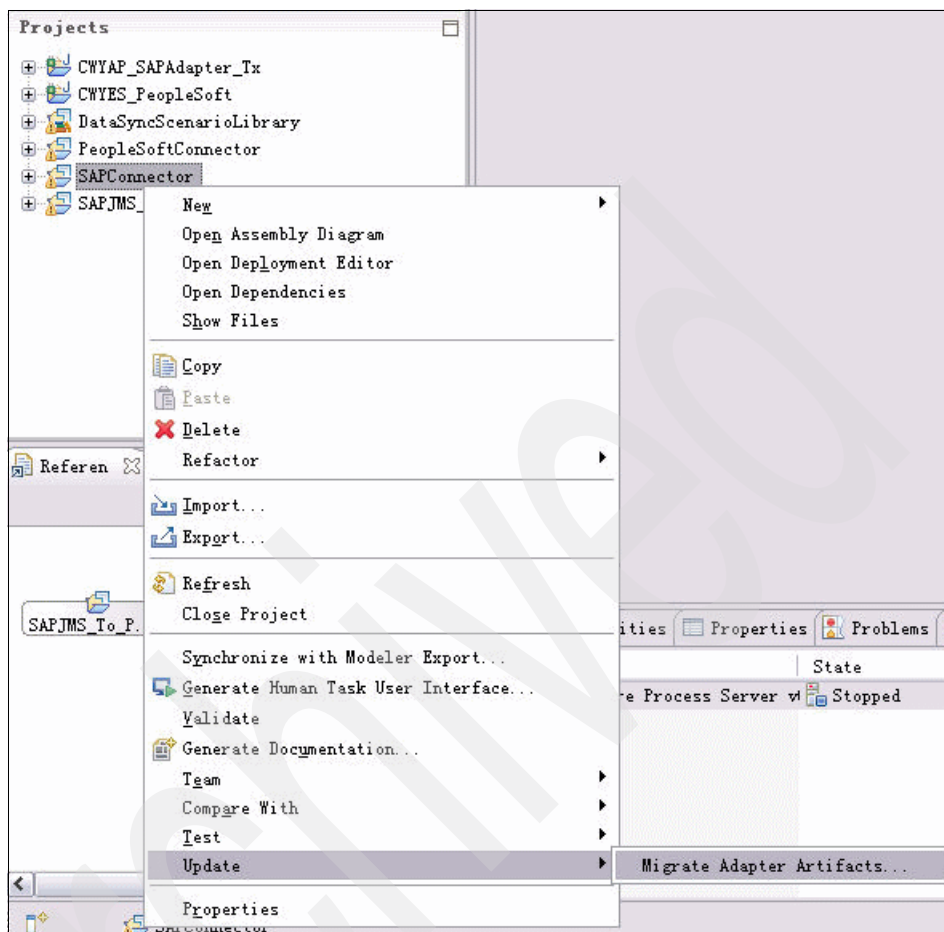


Figure 17-14 Upgrade WebSphere Business Integration Adapter for mySAP.com

- b. In the Migration Wizard: Select Projects window (Figure 17-15 on page 450), accept the default value, and click **Next**.

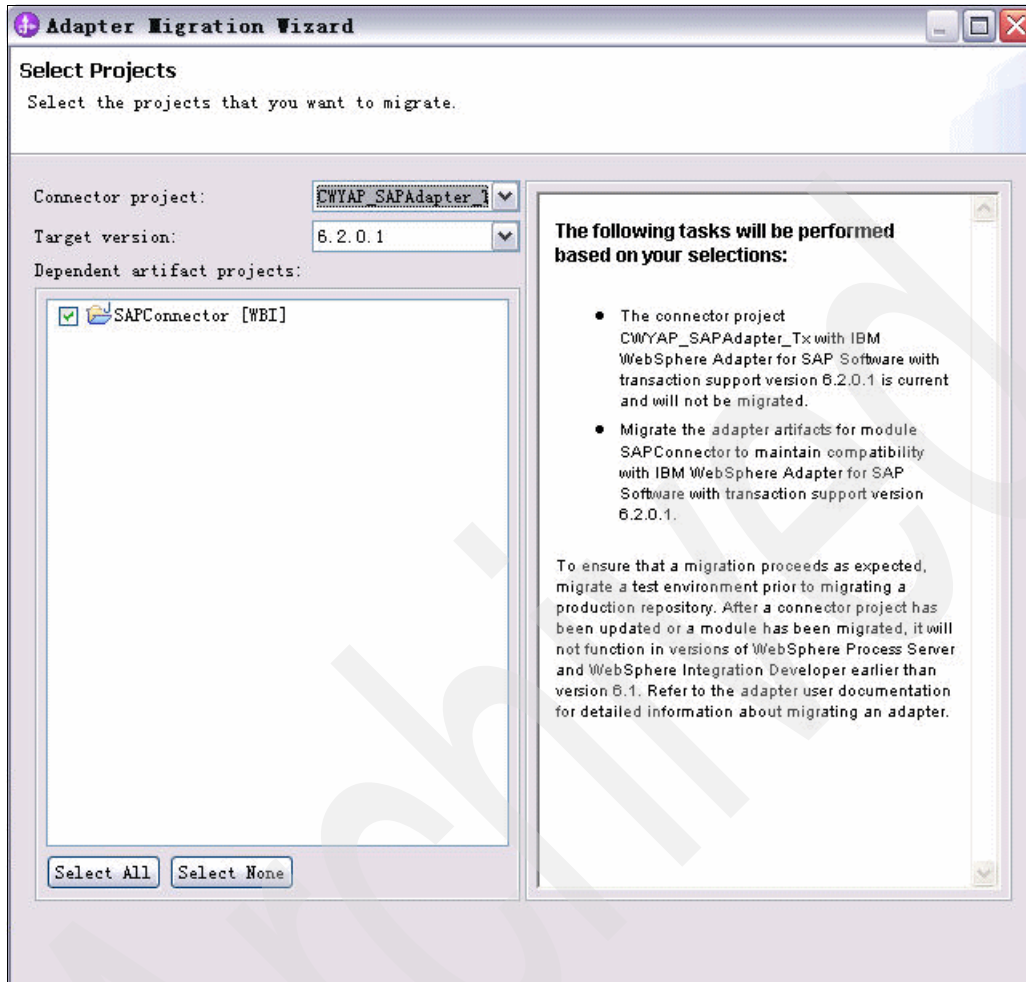


Figure 17-15 Upgrade WebSphere Business Integration Adapter for mySAP.com: Select Projects

- c. In the Migration Wizard Warning window (Figure 17-16), click **OK** to continue.

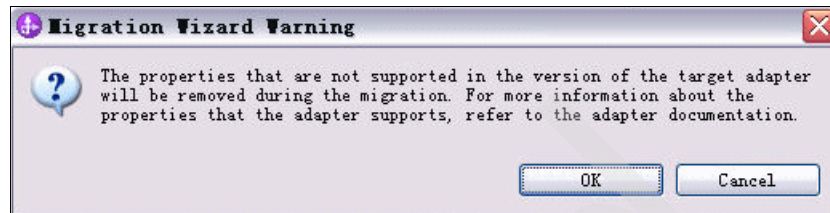


Figure 17-16 Upgrade WebSphere Business Integration Adapter for mySAP.com: Warning

- d. The Migration Wizard takes a second to collect information. When you see the Adapter Migration Wizard: Review Changes window (Figure 17-17 on page 452), click **Finish**.

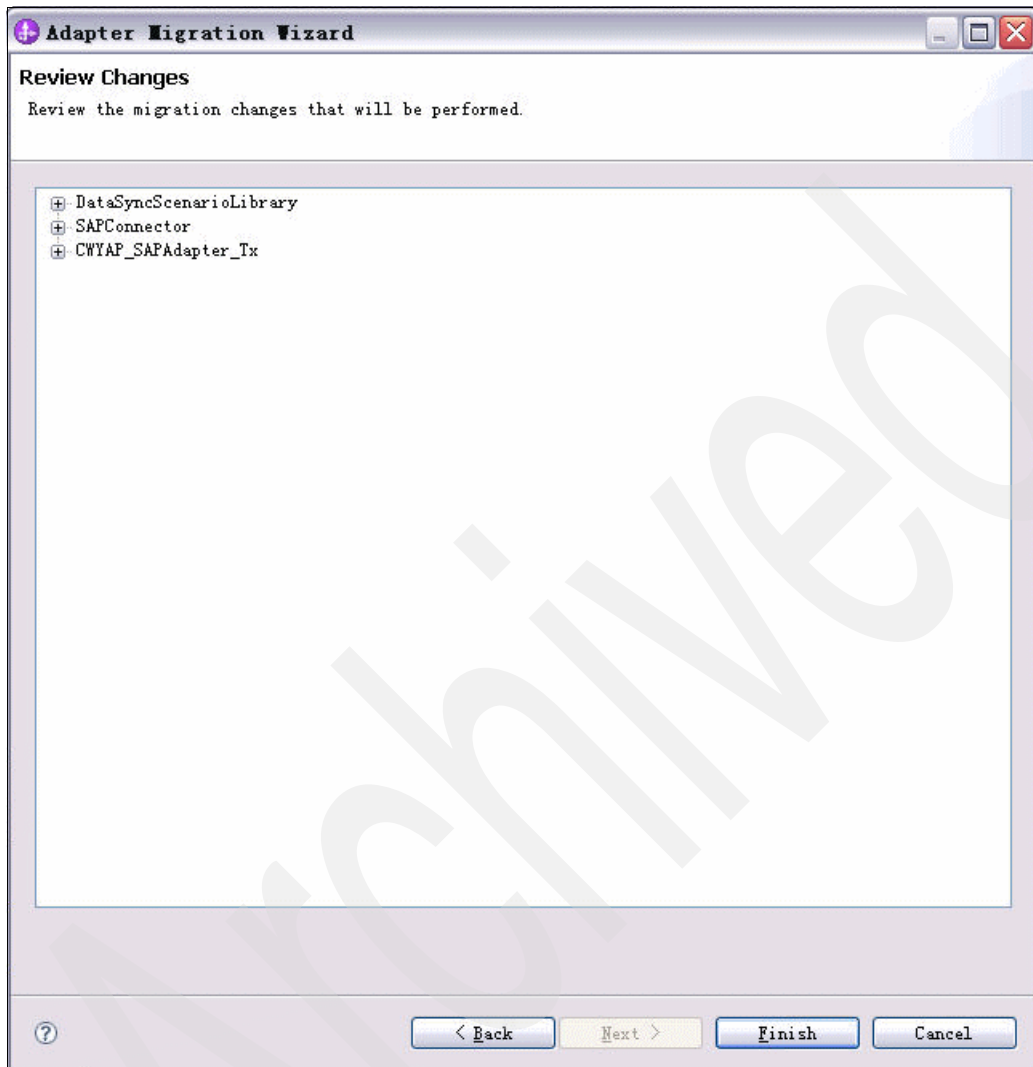


Figure 17-17 Upgrade WebSphere Business Integration Adapter for mySAP.com_Review Changes

- e. The migration executes, and the windows close when it finishes.
5. Copy the Customer BO file back from the c:\temp directory to **DataSyncScenarioLibrary** → **DataTypes**.

6. Because of a unique situation in WebSphere Adapter Migration Tool, several BG files do not appear after the WebSphere Business Integration Adapter for PeopleSoft migration. You must copy all of the SAP-related BO and BG files to a temp directory `c:\tempdir` from **DataSyncScenarioLibrary** → **Data Types** (Figure 17-18). After the WebSphere Business Integration Adapter for PeopleSoft update, you must copy back the BG and the BO files.

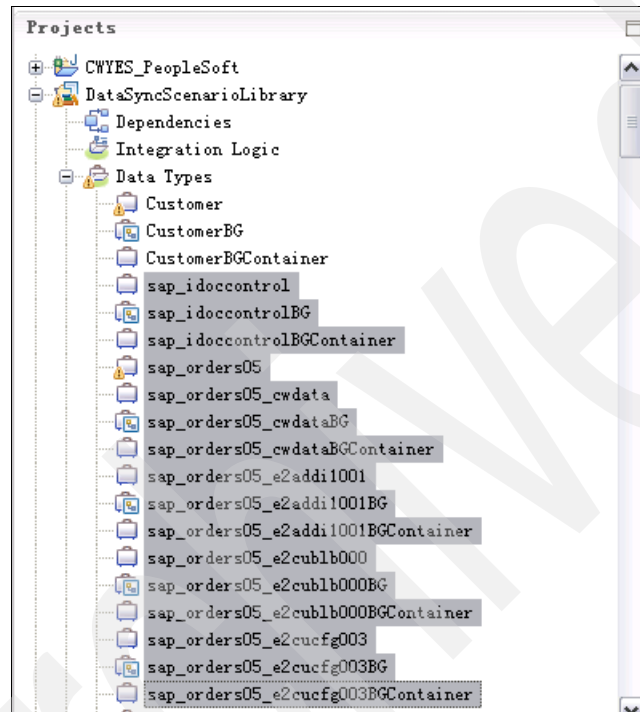


Figure 17-18 Upgrade WebSphere Business Integration Adapter for mySAP.com:
Copy the BO and BG files

7. Upgrade WebSphere Business Integration Adapter for PeopleSoft to WebSphere Adapter for PeopleSoft:
 - a. In the Business Integration perspective, right-click the **PeopleSoftConnector** module, and select **Update** → **Migrate Adapter Artifacts** (Figure 17-19 on page 454).

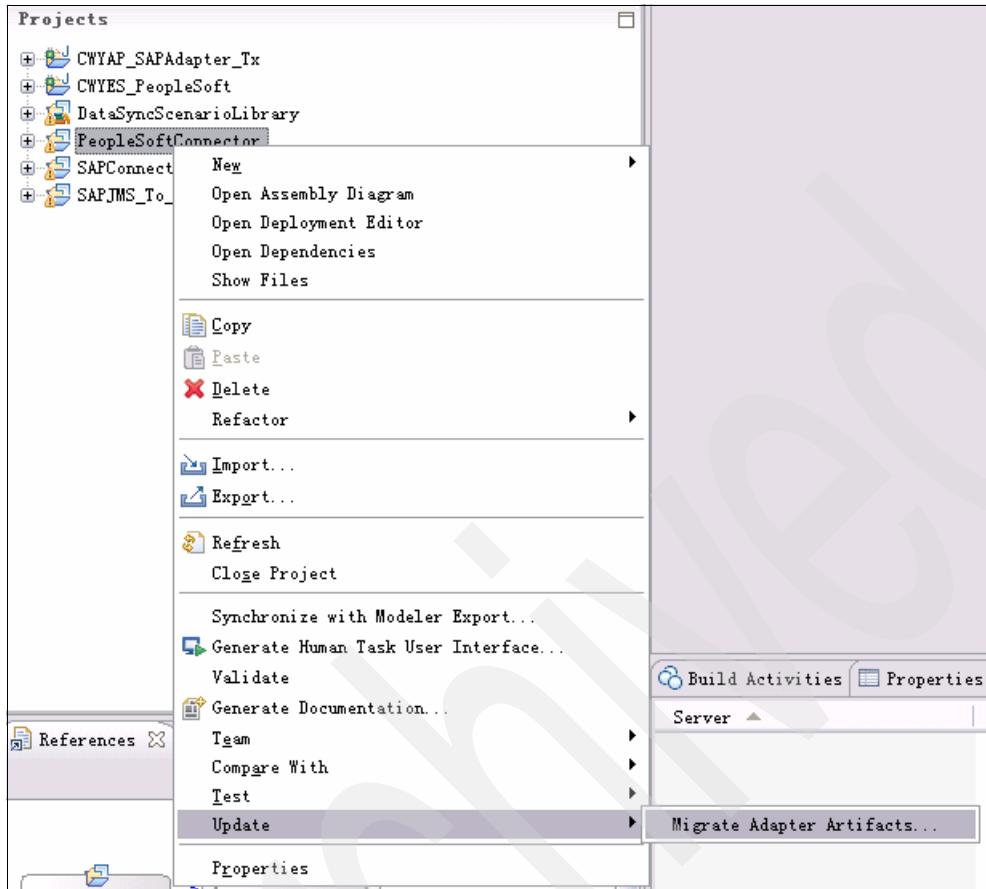


Figure 17-19 Upgrade WebSphere Business Integration Adapter for PeopleSoft

- b. In the Migration Wizard: Select Projects window (Figure 17-20 on page 455), accept the default value, and click **Next**.

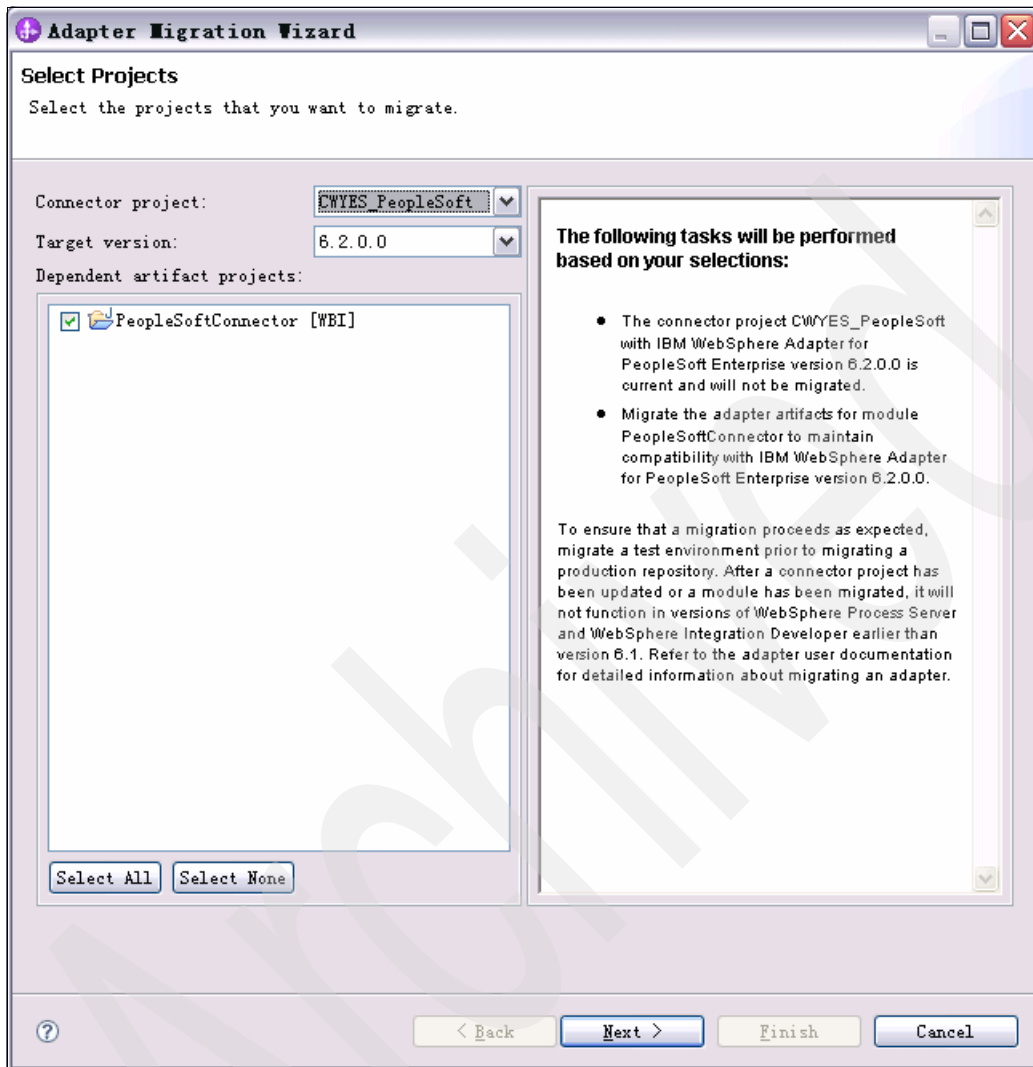


Figure 17-20 Upgrade WebSphere Business Integration Adapter for PeopleSoft: Select Projects

- c. In the Migration Wizard Warning window (Figure 17-21 on page 456), click **OK** to continue.

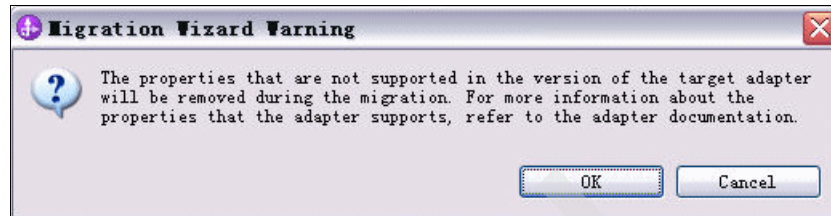


Figure 17-21 Upgrade WebSphere Business Integration Adapter for PeopleSoft: Warning

- d. The Migration Wizard will take a second to collect information. When you see the Adapter Migration Wizard: Review Changes window (Figure 17-22 on page 457), click **Finish**.

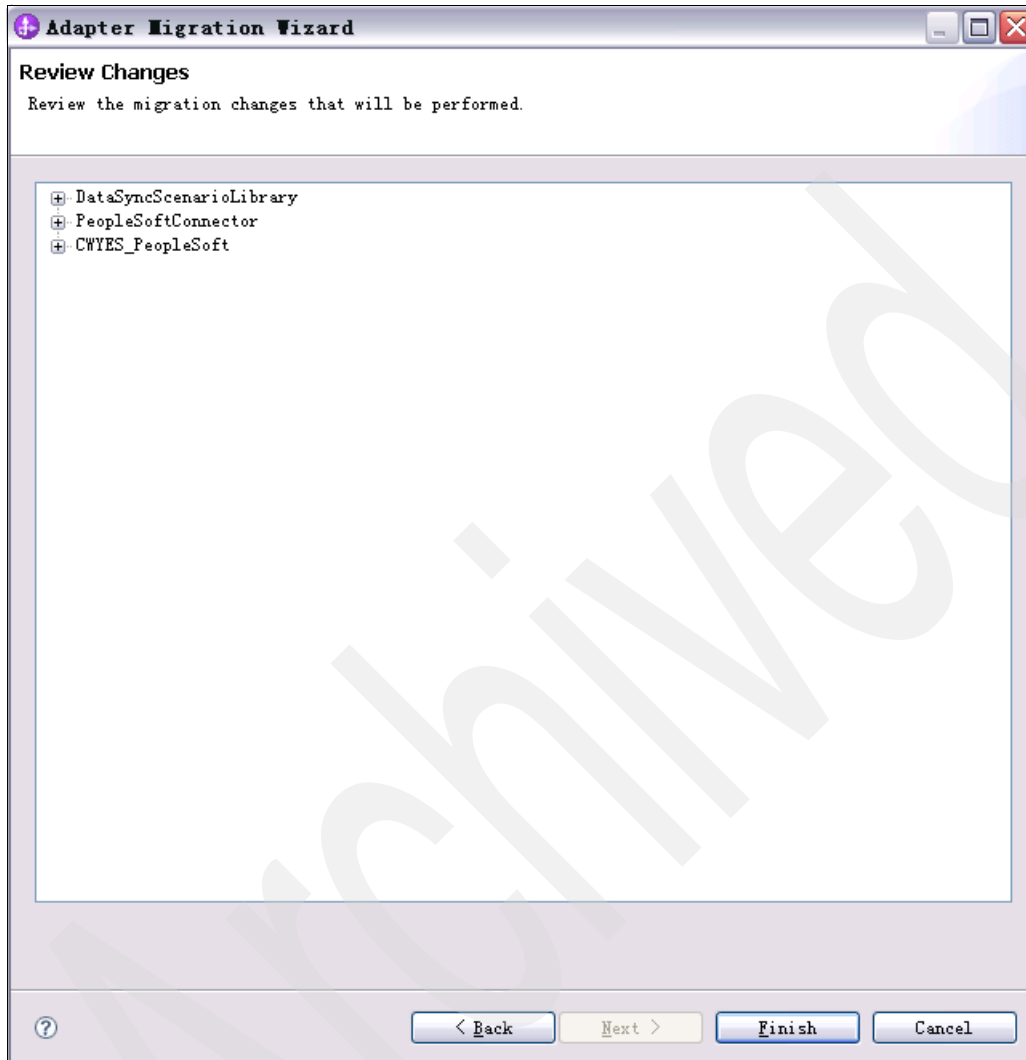


Figure 17-22 Upgrade WebSphere Business Integration Adapter for PeopleSoft: Review Changes

- e. The migration executes, and the windows close when it finishes.
8. Copy all of the SAP-related BO and BG files to **DataSyncScenarioLibrary** → **DataTypes** from the temp directory `c:\tempdir`, and select **Clean** to clean the workspace (Figure 17-23 on page 458).

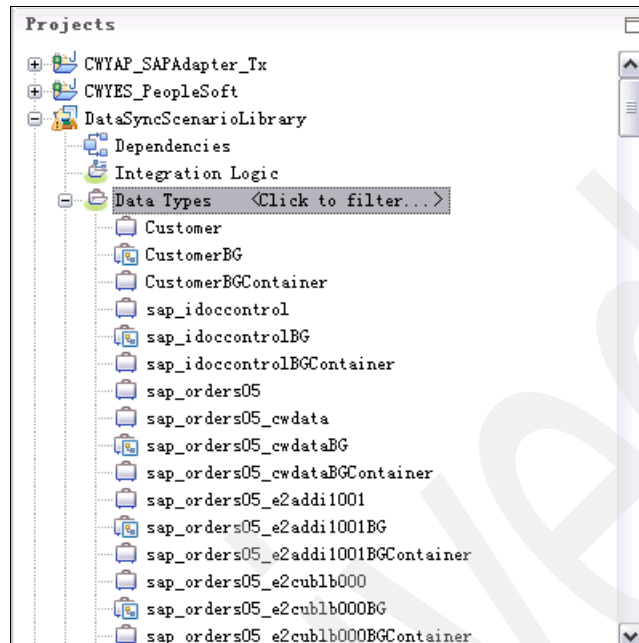


Figure 17-23 Upgrade WebSphere Business Integration Adapter for PeopleSoft: Copy back BO and BG files

9. Because the ALE inbound did not support the synchronize invoke, we need remove the synchronized EIS binding:
 - a. In the Business Integration view, open the assembly module for the WebSphere Adapter for SAP Software (Figure 17-24 on page 459).

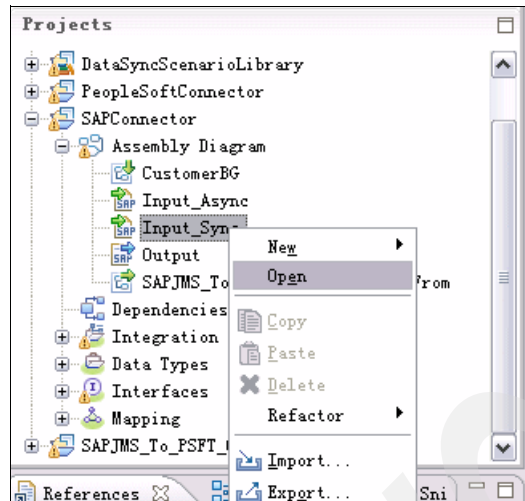


Figure 17-24 In the Business View, opening the assembly module of WebSphere Adapter for SAP Software

- b. Delete the synchronized EIS binding (Figure 17-25), and then, select **Clean** to clean the workspace.

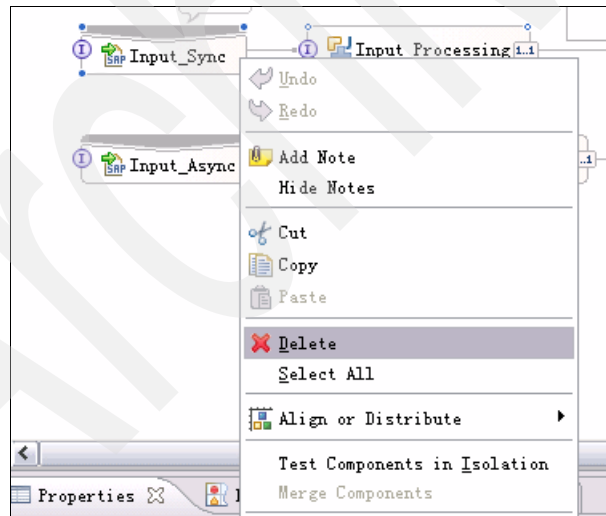


Figure 17-25 Deleting the synchronized EIS binding of WebSphere Adapter for SAP Software

10. Because the BO handling between WebSphere Business Integration Adapter for mySAP.com and WebSphere Adapter for SAP Software differs, you must add code to update several values of the BO:
- In the Business Integration view, open the **Input_Async_Processing** Java component (Figure 17-26).

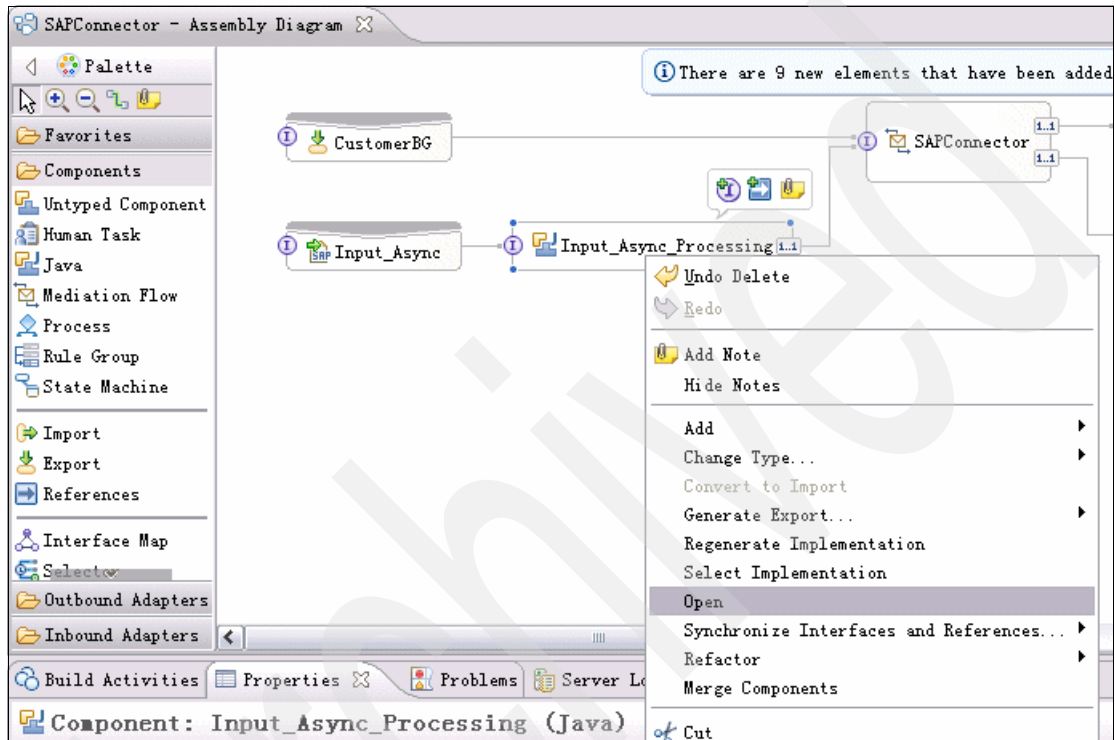


Figure 17-26 Open the *Input_Async_Processing* Java component

- b. Modify the emitCreateAfterImagesap_orders05 method of the Input_Async_Processing class, as in Figure 17-1 on page 434.

Example 17-1 Input_Async_Processing Java component

```
public void emitCreateAfterImagesap_orders05(
DataObject emitCreateAfterImagesap_orders05Input) throws Exception
{
DataObject orders05 =
emitCreateAfterImagesap_orders05Input.getDataObject("sap_orders05");
DataObject dataRecord = orders05.getDataObject("Data_record");
DataObject orders1005 =
dataRecord.getDataObject("sap_orders05_e2edk01005");
orders05.setString("Dummy_key", orders1005
.getString("Action_code_for_the_whole_EDI_message"));
invokeService(emitCreateAfterImagesap_orders05Input);
}
```

11. Because the BO definition between WebSphere InterChange Server and WebSphere Process Server differs, the copy BO API does not work. You must rewrite the code to use the WebSphere Process Server API instead. Follow these steps to modify the Java snippet in BPEL:
 - a. In the Business Integration view, expand the **SAPJMS_To_PSFT_CustomerSync** module, expand **Integration Logic**, expand **Processes**, and double-click **CustomerSync_From** to open the process.
 - b. Click the process name on the top of the right side to modify the process properties (Figure 17-27 on page 462).

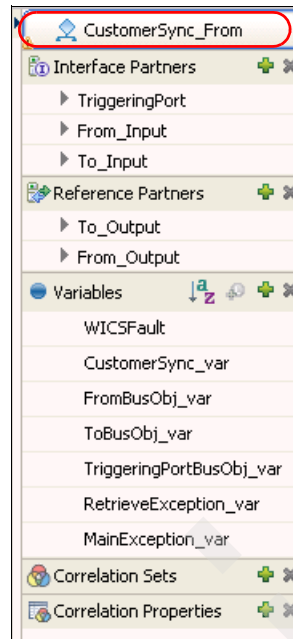


Figure 17-27 Modify BPEL: Select process

- c. In the Process: CustomerSync_From properties view, select **Java Imports**, and modify the following line (refer to Figure 17-28):

```
import com.ibm.websphere.bo.BOFactory;
```

to:

```
import com.ibm.websphere.bo.*;
```

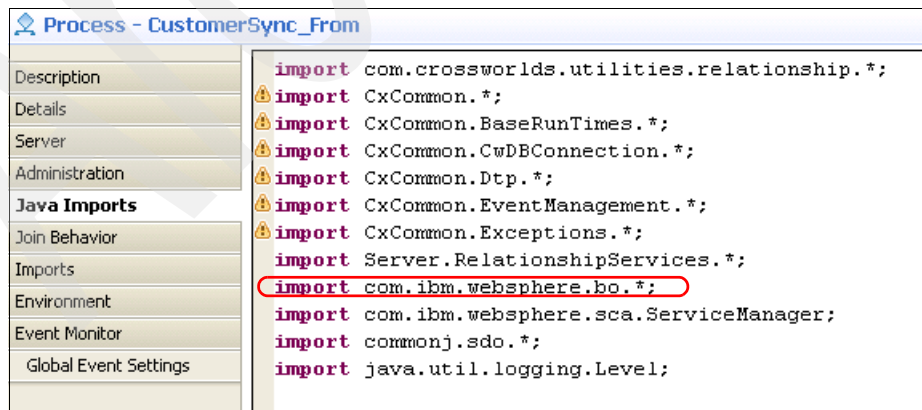


Figure 17-28 Modify BPEL: Modify Java Imports

- d. Right-click the canvas, and select **Expand All Activities**.
- e. Right-click the canvas again, and select **Align parallel Activities Contents Automatically**.
- f. Find the node **Initialize_Parameters_1** (see Figure 17-29).

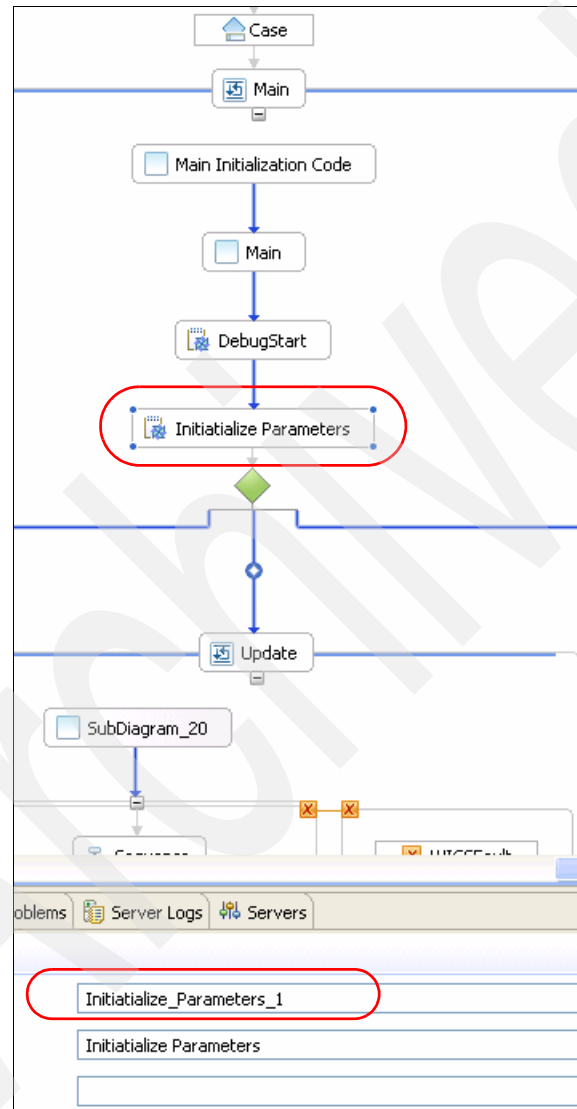


Figure 17-29 Modify BPEL: Select node

- g. In the Snippet _ Initialize Parameters property view, select the **Detail** tab.

- h. Modify the Java snippet as shown in Example 17-2.

Example 17-2 Modifying CustomerSync_From _Modify node Initialize_parameters Java snippet

```
//var_2.copy(var_4);
//var_3.copy(var_14);
}
BOCopy boCopy =
(BOCopy)ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOC
opy");

    FromBusObj_var= boCopy.copy(TriggeringPortBusObj_var);
    ToBusObj_var = boCopy.copy(TriggeringPortBusObj_var);

//if (triggeringBusObj == null) { TriggeringPortBusObj_var = null; }
else { TriggeringPortBusObj_var = triggeringBusObj.getBusinessGraph();
}
//if (FromBusObj == null) { FromBusObj_var = null; } else {
FromBusObj_var = FromBusObj.getBusinessGraph(); }
//if (ToBusObj == null) { ToBusObj_var = null; } else { ToBusObj_var =
ToBusObj.getBusinessGraph(); }
```

- i. Save and close the CustomerSync_From process.

12. Modify the .mon file for each process:

- a. Switch to the Java perspective. In the WebSphere Integration Developer menu bar, click **Window** → **Open Perspective** → **Other**. In the Open Perspective window (Figure 17-30 on page 465), select **Java**, and click **OK**.

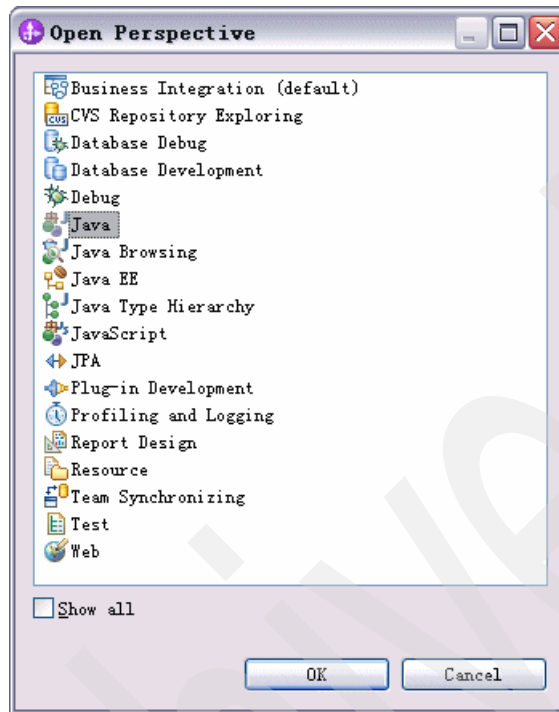


Figure 17-30 *Open Perspective: Java*

- b. Expand the **SAPJMS_To_PSFT_CustomerSync** project. Double-click the **CustomerSync_From_bpel.mon** file (Figure 17-31).

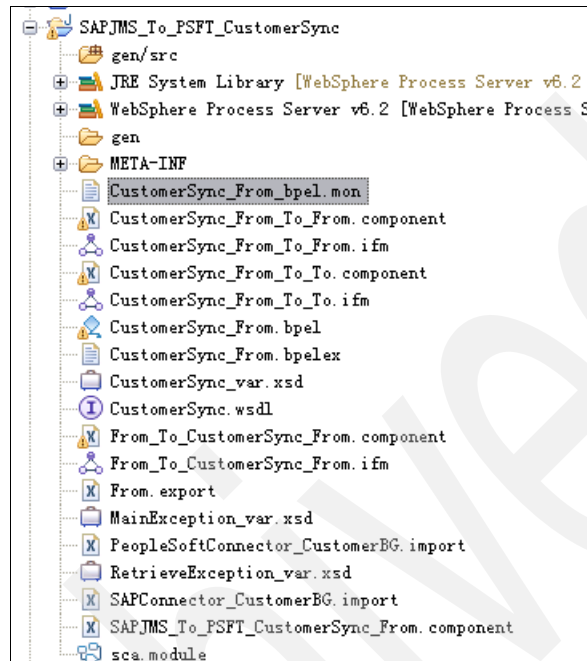


Figure 17-31 Select the *CustomerSync_From_bpel.mon* file

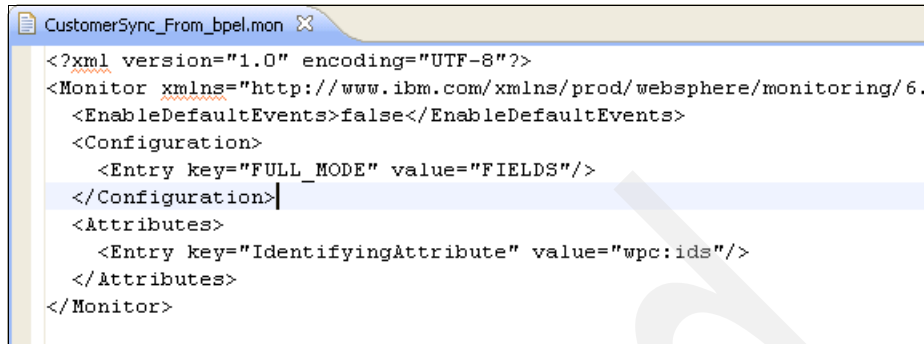
- c. Modify the following line:

```
<Entry key="IdentifyingAttribute" value="name"/>
```

to:

```
<Entry key="IdentifyingAttribute" value="wpc:ids"/>
```

See Figure 17-32 on page 467.



```

<?xml version="1.0" encoding="UTF-8"?>
<Monitor xmlns="http://www.ibm.com/xmlns/prod/websphere/monitoring/6.
  <EnableDefaultEvents>false</EnableDefaultEvents>
  <Configuration>
    <Entry key="FULL_MODE" value="FIELDS"/>
  </Configuration>
  <Attributes>
    <Entry key="IdentifyingAttribute" value="wpc:ids"/>
  </Attributes>
</Monitor>

```

Figure 17-32 Modify the CustomerSync_From_bpel.mon file

Note: The file with the .mon suffix does not exist after the migration. It will be generated after the modification on the BPEL process BPEL when you build the project. Make sure that this step is performed after step 5.

17.2.4 Work-around for dynamic relationship

After the migration, dynamic relationships do not work because of differences in the transaction management approaches between WebSphere InterChange Server and WebSphere Process Server. In WebSphere Process Server, the transactions are managed by the underlying WebSphere Application Server. WebSphere InterChange Server does not have this capability. Because of this key difference, you must modify the stored procedures that manage the dynamic relationship tables so that there are no explicit commits or rollbacks when the tables are used with WebSphere Process Server.

In this specific scenario, we have one dynamic relationship, Customer, which uses three stored procedures. You only need to modify two of them: MQDLCUST_RELATIONSHIP_SP and JDBCCUST_RELATIONSHIP_SP. These stored procedures have one rollback statement and one commit statement each.

By using the DB2 Development Center, modify the stored procedures by either commenting out or removing the rollback and commit statements:

1. Launch the DB2 Development Center. Select **Start** → **Programs** → **IBM DB2** → **Development Tools** → **Development Center**. The stored procedure wizard automatically launches.
2. In the Development Center Launchpad window (Figure 17-33 on page 468), click **Create Project**.

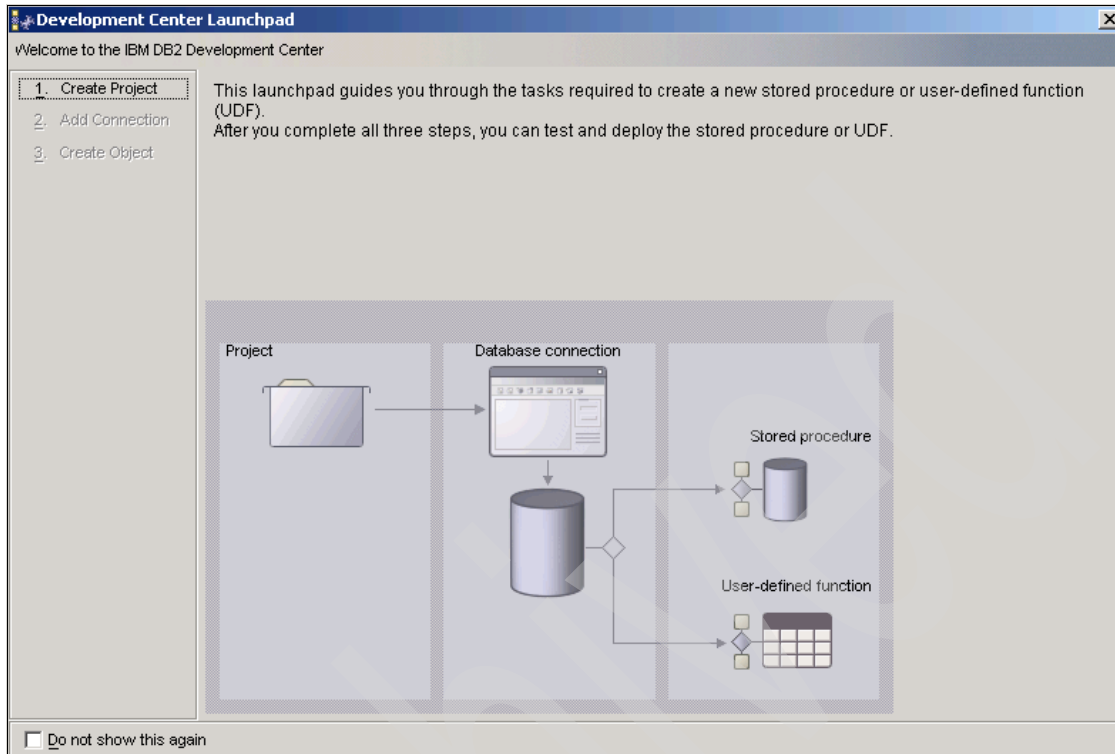


Figure 17-33 DB2 Development Center: Launchpad for the stored procedure

3. In the Open Project window (Figure 17-34), select a project name. Use the default value of Project1 if it does not already exist. Click **OK**.

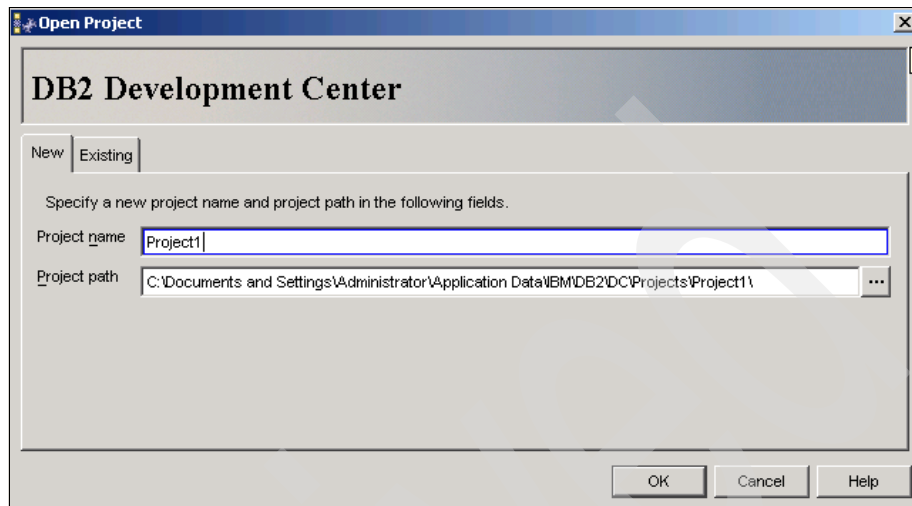


Figure 17-34 DB2 Development Center: Creating a project

4. In the Development Center Launchpad window (Figure 17-35), click **Add Connection**.

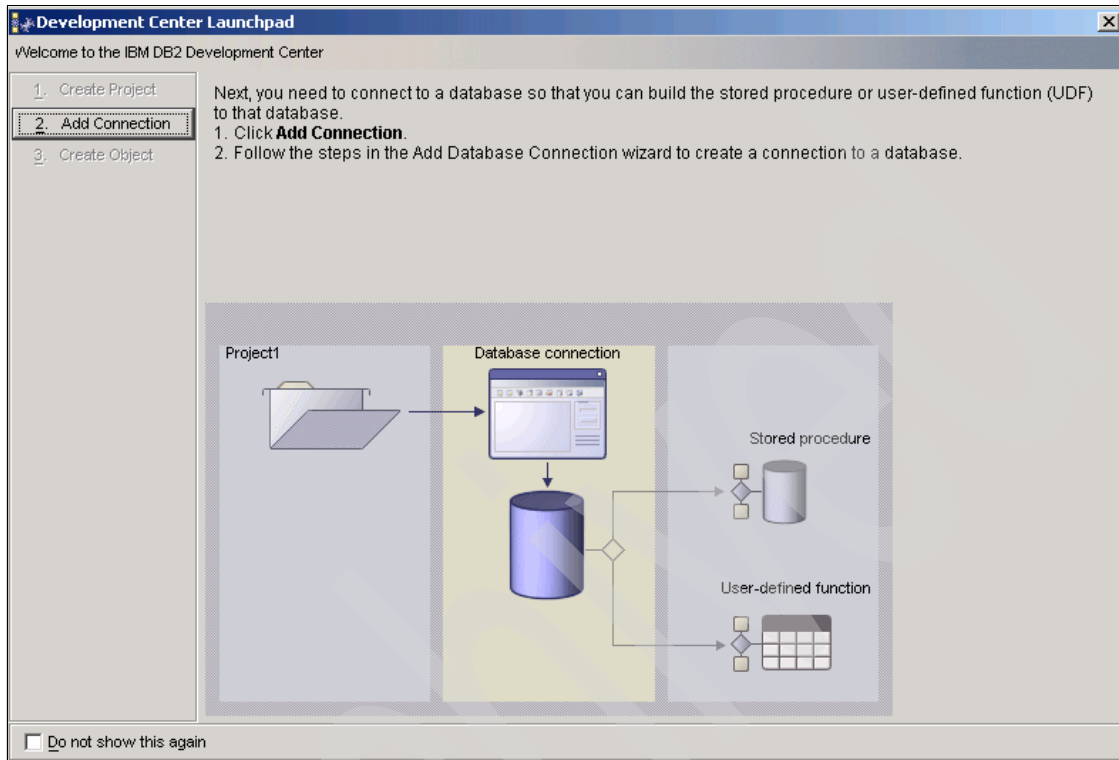


Figure 17-35 DB2 Development Center: Add Connection

5. In the Add Database Connection Wizard window (Figure 17-36), in the Connection Type pane, select **Online**. Click **Next**.

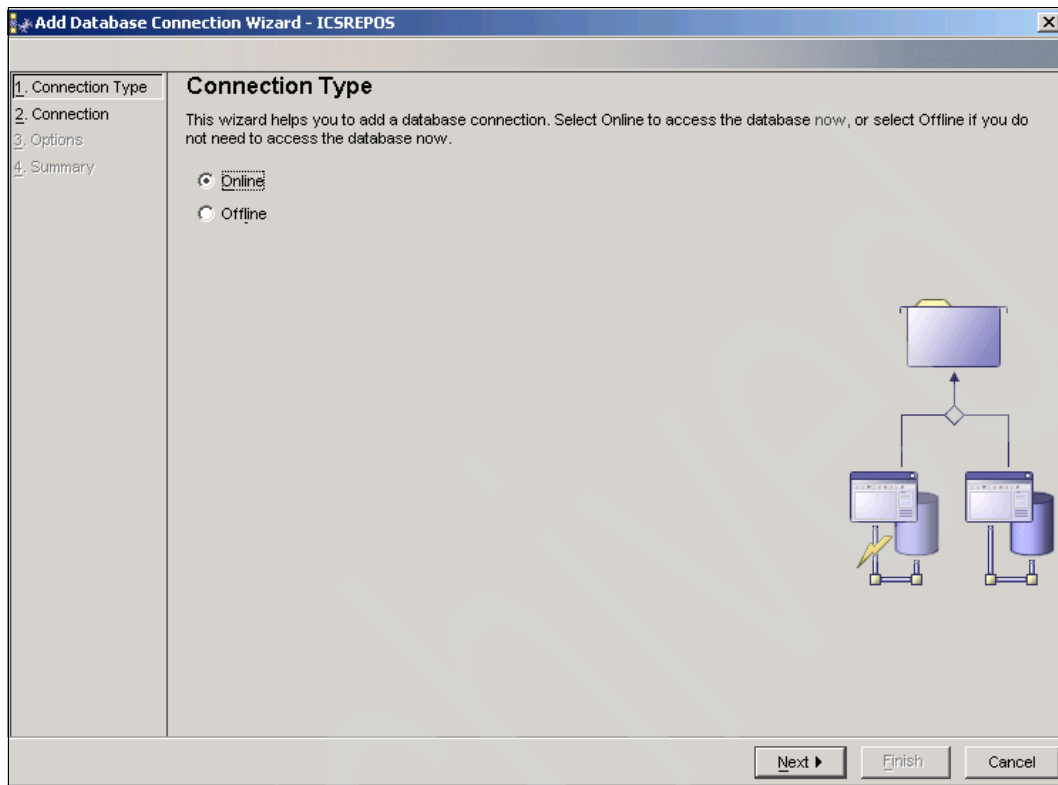


Figure 17-36 DB2 Development Center: Connection Type

6. In the Connection Wizard window (Figure 17-37), in the Connection pane, provide the connection details for the database. Click **Finish**, and close the connection wizard.

Add Database Connection Wizard - ICSREPOS

1. Connection Type
2. **Connection**
3. Options
4. Summary

Connection
Specify a driver and the database alias that you want to use for your connection.

Driver: IBM DB2 alias

Database:
Database: ICSREPOS
Comments:
Host name:
Port number:
Location (URL): jdbc:db2:ICSREPOS
Driver class: COM.ibm.db2.jdbc.app.DB2Driver

User information:
☐ Use your current user ID and password
User ID: Administrator
Password: *****

Test Connection

Navigation: Back Next Finish Cancel

Figure 17-37 DB2 Development Center: Connection

7. With the project now visible in the Project View (Figure 17-38), right-click **Stored Procedures**, and select **Import**.

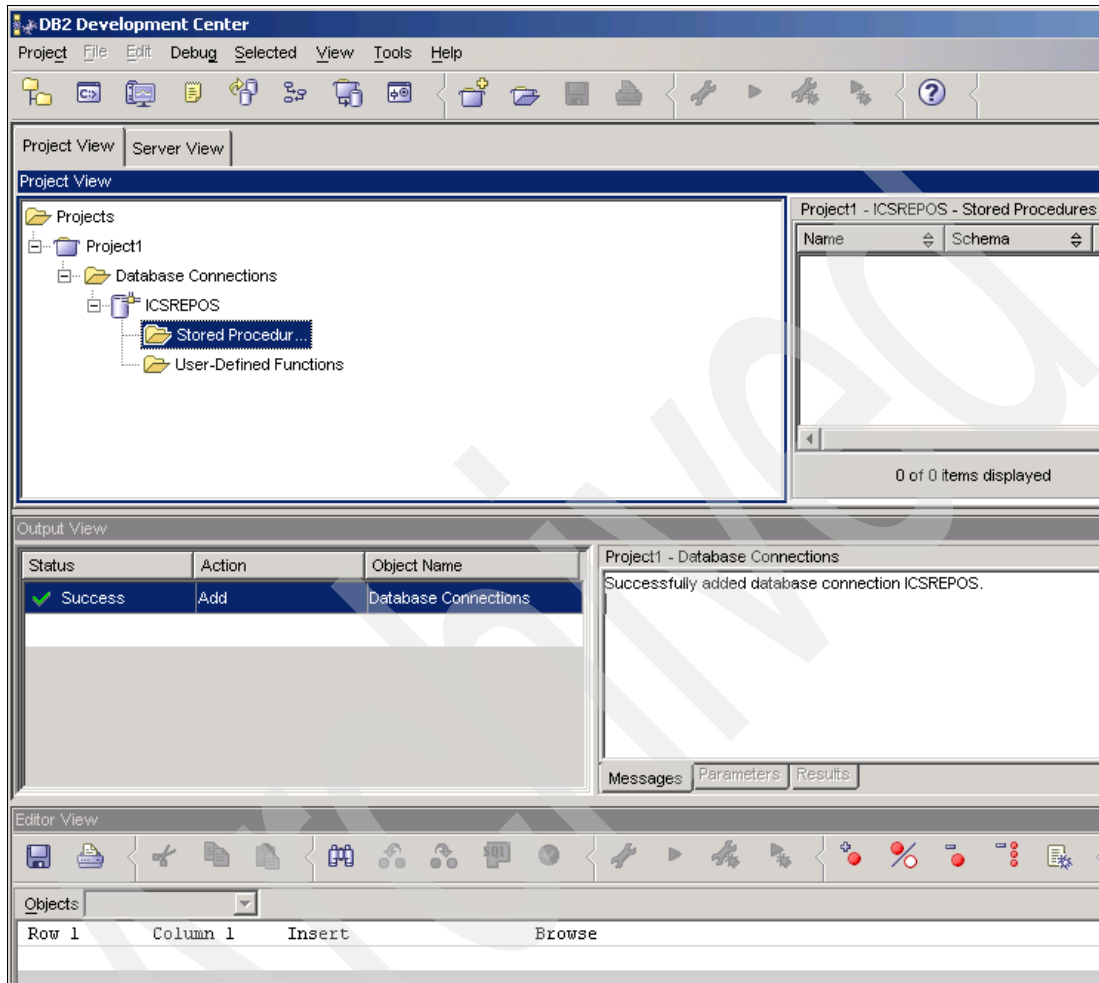


Figure 17-38 DB2 Development Center: Project created

8. In the Import window (Figure 17-39), select **Database** and **Stored Procedure**, and click **OK**.

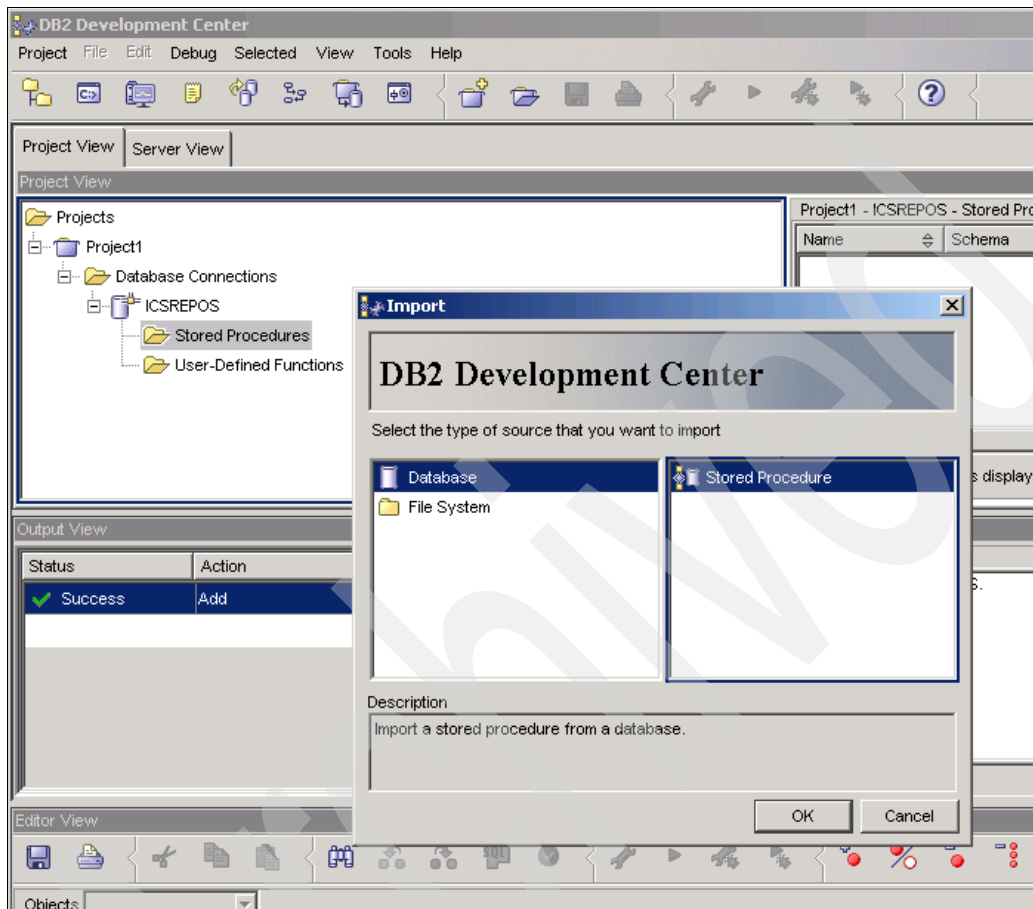


Figure 17-39 DB2 Development Center: Importing stored procedures

9. In the Source Database pane of the Import Wizard window (Figure 17-40), select the source database from the list. Click **Next**.

The screenshot shows the 'Import Wizard' window with the 'Source Database' pane selected. The pane contains a list of steps on the left: 1. Source Database, 2. Filter, 3. Objects, 4. Options, and 5. Summary. The main area is titled 'Source Database' and contains a description: 'This wizard helps you import objects such as stored procedures and user-defined functions from a database. Specify the alias and login information for the source database. The target database is your current connection.' Below this, there are several input fields: 'Driver' (set to 'IBM DB2 alias'), 'Database' (set to 'ICSREPOS'), 'Comments', 'Host name', 'Port number', 'Location(URL)' (set to 'jdbc:db2:ICSREPOS'), and 'Driver class' (set to 'COM.ibm.db2.jdbc.app.DB2Driver'). There is also a 'User information' section with a checkbox 'Use your current user ID and password' (unchecked), a 'User ID' field (set to 'Administrator'), and a 'Password' field (masked with asterisks). On the right side of the pane, there is a 3D cylinder icon representing a database. At the bottom right, there are three buttons: 'Next >', 'Finish', and 'Cancel'.

Figure 17-40 DB2 Development Center: Specifying the source database

10. In the Filter pane of the Import Wizard window (Figure 17-41), accept the defaults, and click **Next**.

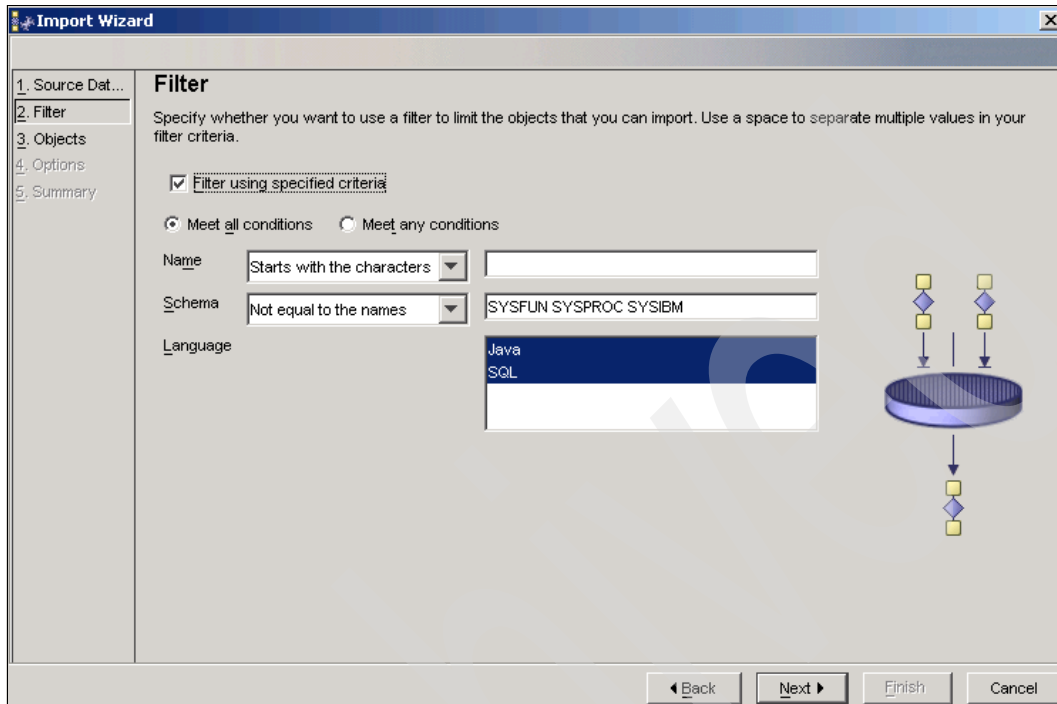


Figure 17-41 DB2 Development Center: Providing the filter details

11. In the Objects pane of the Import Wizard window (Figure 17-42), under the Available heading, select the two dynamic relationship stored procedures **username.SAPORDER_RELATIONSHIP_SP** and **username.PSFTCUST_RELATIONSHIP_SP** to modify. Click > to move them to the Selected pane. Then, click **Finish**.

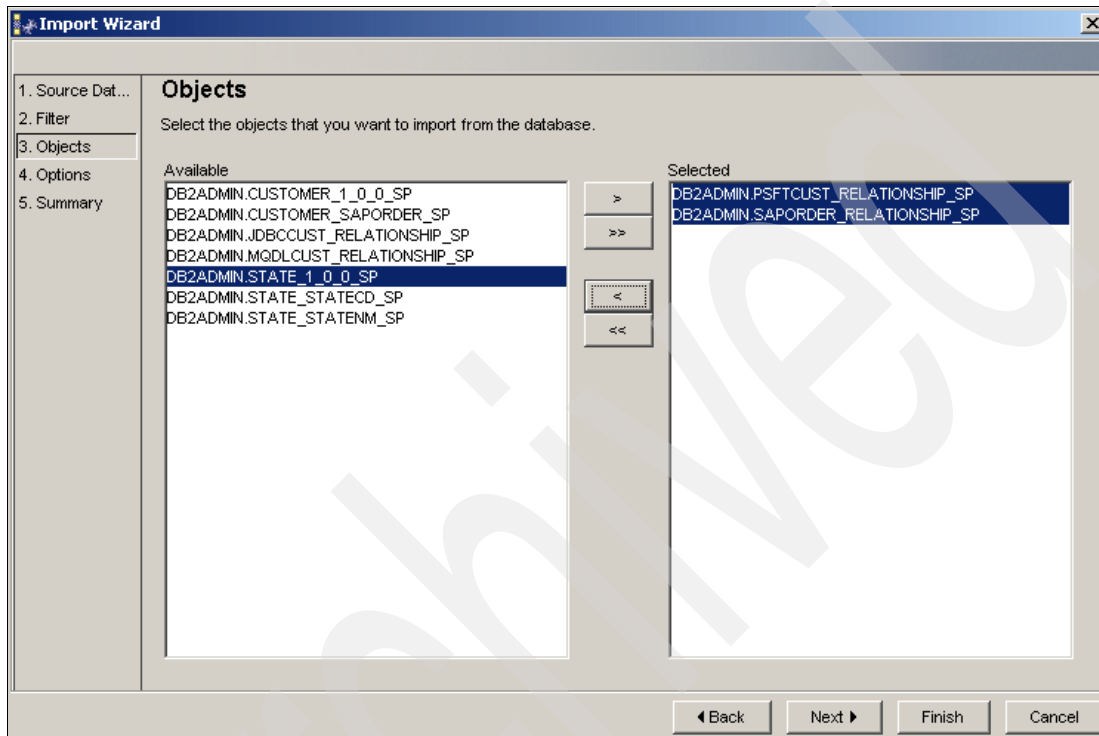


Figure 17-42 DB2 Development Center: Selecting the stored procedures that you want to modify

12. In the DB2 Development Center window (Figure 17-43), double-click a stored procedure to open it in the Editor View. Comment out the **Rollback** and **Commit** statements in the stored procedure. There is only one rollback statement and one commit statement in each stored procedure. Click the **build** icon in the Editor View. Click **Yes** if you see a message window.

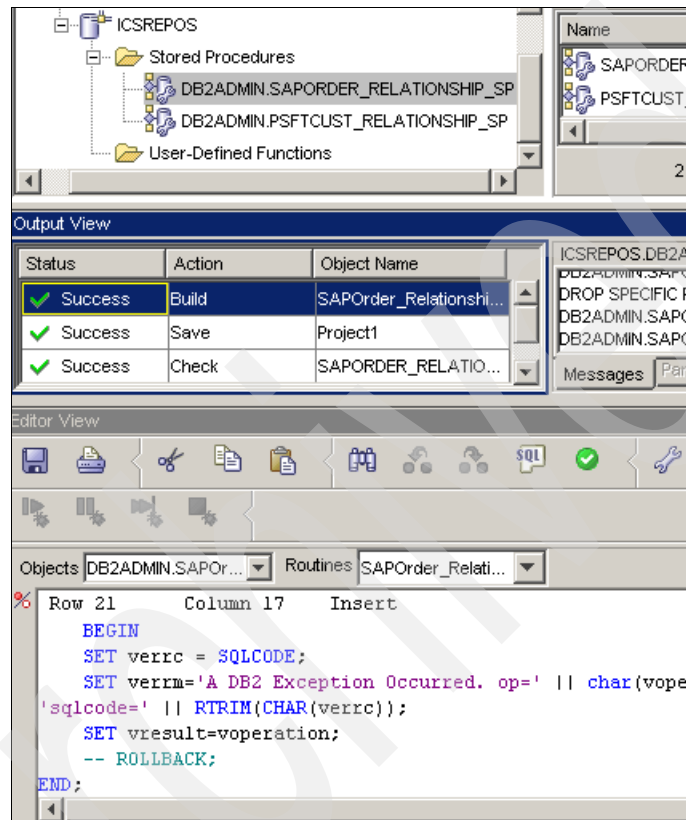


Figure 17-43 DB2 Development Center: Opening a stored procedure that you want to modify in the Editor View

17.2.5 Running a Jython script and modifying JDBC data sources

For WebSphere InterChange Server artifacts that have no corresponding artifacts in WebSphere Process Server, a Jython script is generated during the migration. For this scenario, the Jython script that is generated has references to relationships. You must run this script by using the **wsadmin** command to create WebSphere Process Server configuration definitions that correspond to the original WebSphere InterChange Server artifacts.

Saving and applying changes: In the following steps, apply and save every change that you make in the administrative console.

Follow these steps to run the Jython script and modify the JDBC data sources:

1. Switch your perspective in WebSphere Integration Developer (Figure 17-44).

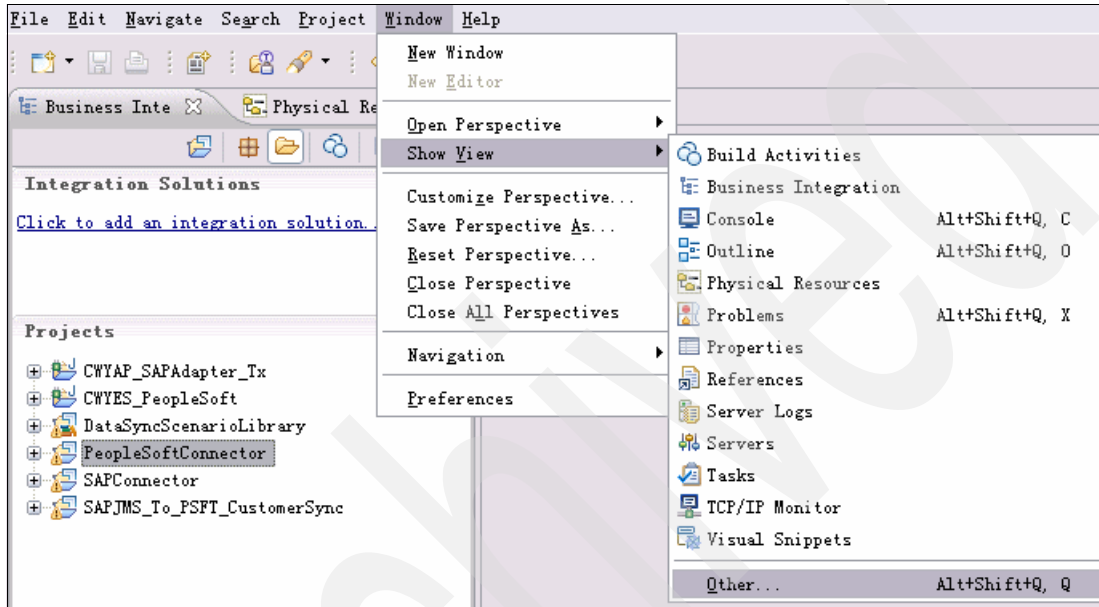


Figure 17-44 Switching perspectives

2. Select **Java**.

The Jython script `InstallAdministrativeObjects.py` is placed under the library project, `DataSyncScenarioLibrary`, in this scenario (Figure 17-45). The generated Jython script is self-explanatory and contains details about how to run the file.

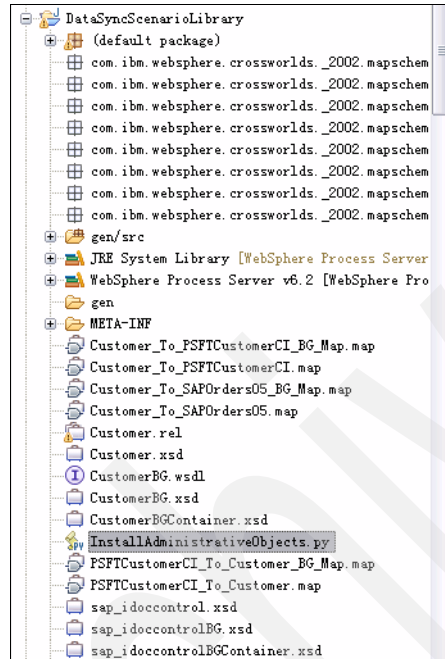


Figure 17-45 Jython script

3. Switch back to the Business Integration perspective to start the embedded instance of WebSphere Process Server. Under the **Servers** tab, right-click the server instance, and select **Start**.
4. Run the Jython script by using the following command from the bin folder of the WebSphere Process Server installation. Modify the file path if your workspace location is different:

```
C:\IBM\WID62\pf\wps\bin> wsadmin -lang jython -f  
C:\DataSyncScenarioWorkspace\DataSyncScenarioLibrary\InstallAdministrativeObjects.py
```

Running this script creates a new JDBC provider and corresponding data sources that are necessary for relationships that are being used in the WebSphere InterChange Server project.

- Click **WBI 6.0 Migration DB2 Universal JDBC Driver Provider** to open it (Figure 17-46). The new DB2 JDBC provider is created in the cell scope.

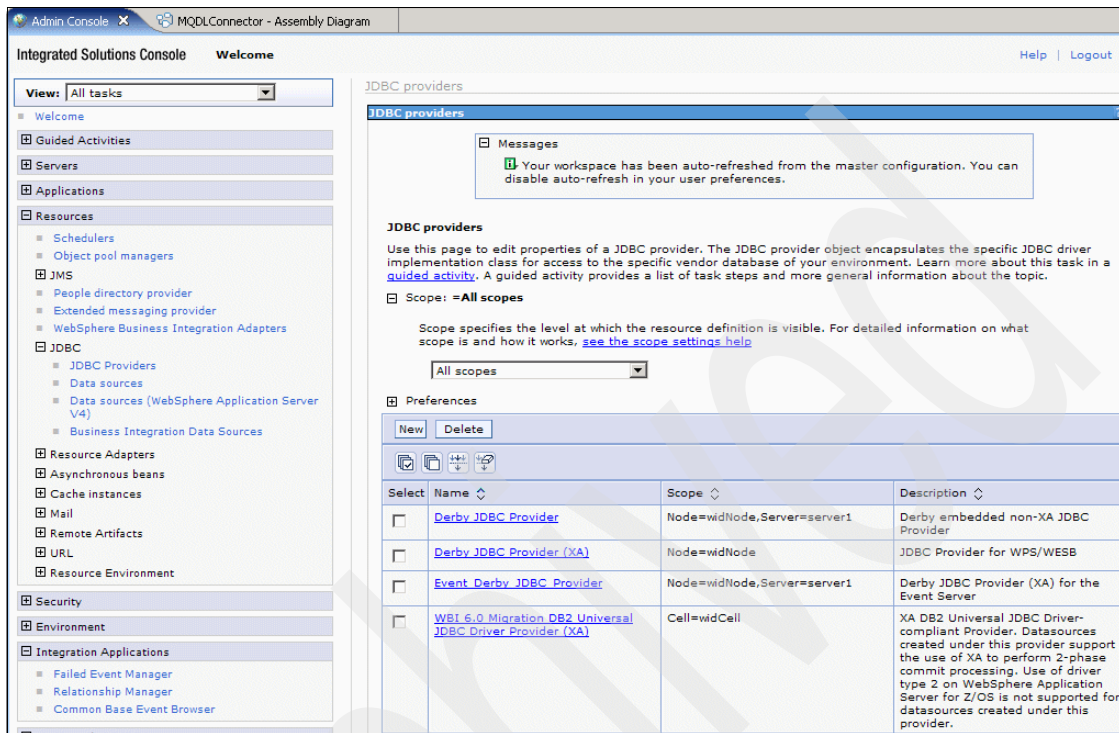


Figure 17-46 JDBC provider created in cell scope

- Under Additional Properties, click **data sources**, and open the **Customer** data source (Figure 17-47).

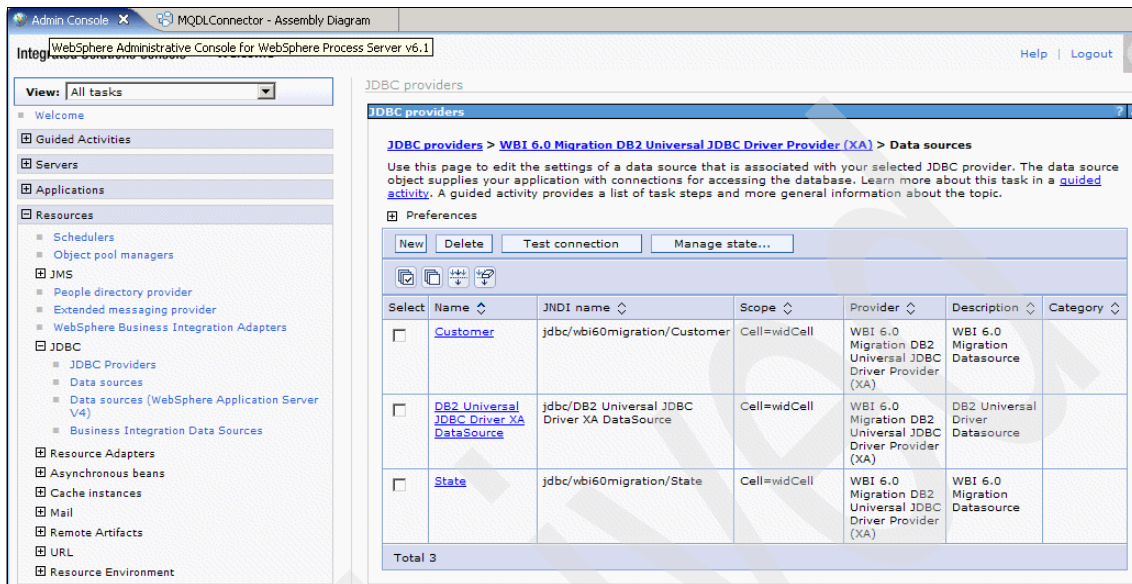


Figure 17-47 Data source created

- Under DB2 Universal data source properties (Figure 17-48), make sure that the Driver type is set to 4.

DB2 Universal data source properties

* Database name
icsrepos

* Driver type
4

Server name
localhost

Port number
50000

Apply OK Reset Cancel

Figure 17-48 Driver type

- Under Data store helper class name (Figure 17-49), confirm that the correct helper class is selected, and save the changes.

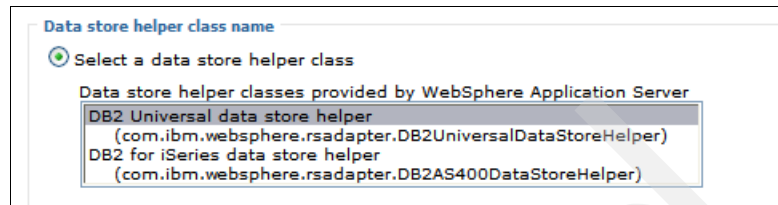


Figure 17-49 Helper class

- Under General Properties for JDBC providers (Figure 17-50), set the correct password for J2C security for the Administrator_Customer and Administrator_State data source aliases. Save the changes.

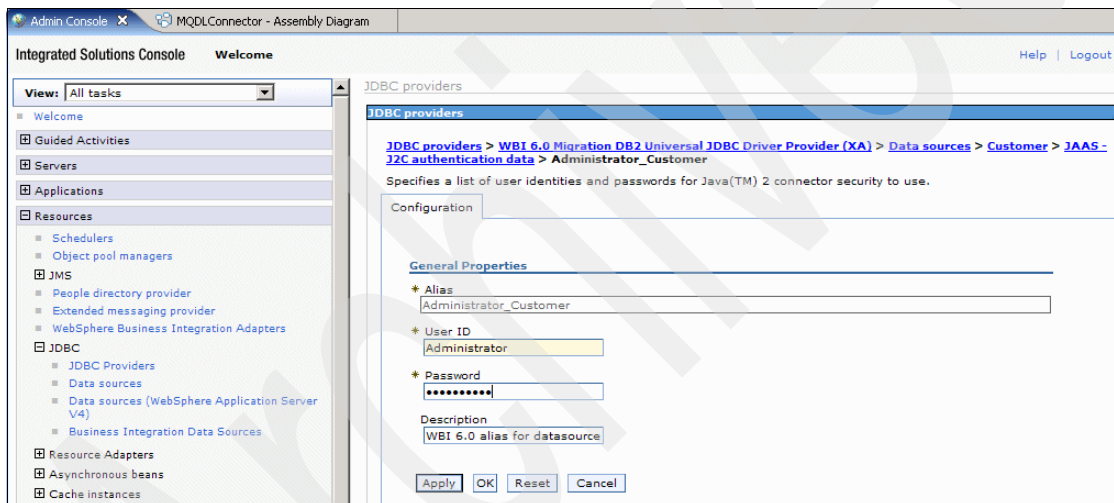


Figure 17-50 Setting the correct password for authentication

10. For the DB2 Universal JDBC Driver XA DataSource data source, set the Component-managed authentication alias to **Administrator_Customer**. For the Database name, type `icsrepos`, and for the Server name, type `localhost` (Figure 17-51).

The screenshot shows the Admin Console interface for configuring a data source. The left sidebar contains a navigation tree with the following items: Welcome, Guided Activities, Servers, Applications, Resources (expanded), Schedulers, Object pool managers, JMS, People directory provider, Extended messaging provider, WebSphere Business Integration Adapters, JDBC (expanded), JDBC Providers, Data sources, Data sources (WebSphere Application Server V4), Business Integration Data Sources, Resource Adapters, Asynchronous beans, Cache instances, Mail, Remote Artifacts, URL, Resource Environment, Security, Environment, Integration Applications, System administration, Users and Groups, Monitoring and Tuning, Troubleshooting, and Service integration. The main panel displays the configuration for the DB2 Universal JDBC Driver XA DataSource. The 'Component-managed authentication alias' is set to 'Administrator_Customer'. The 'Database name' is 'icsrepos', the 'Driver type' is '4', the 'Server name' is 'localhost', and the 'Port number' is '50000'. The 'Authentication alias for XA recovery' is set to '(none)'. The 'Container-managed authentication' is set to '(none)'. The 'Mapping-configuration alias' is set to '(none)'. The 'DB2 Universal data source properties' section is expanded, showing the following fields: Database name (icsrepos), Driver type (4), Server name (localhost), and Port number (50000). The 'Apply', 'OK', 'Reset', and 'Cancel' buttons are at the bottom.

Figure 17-51 XA DataSource properties

11. Test all data source connections, and verify that they connect successfully as shown in Figure 17-52.

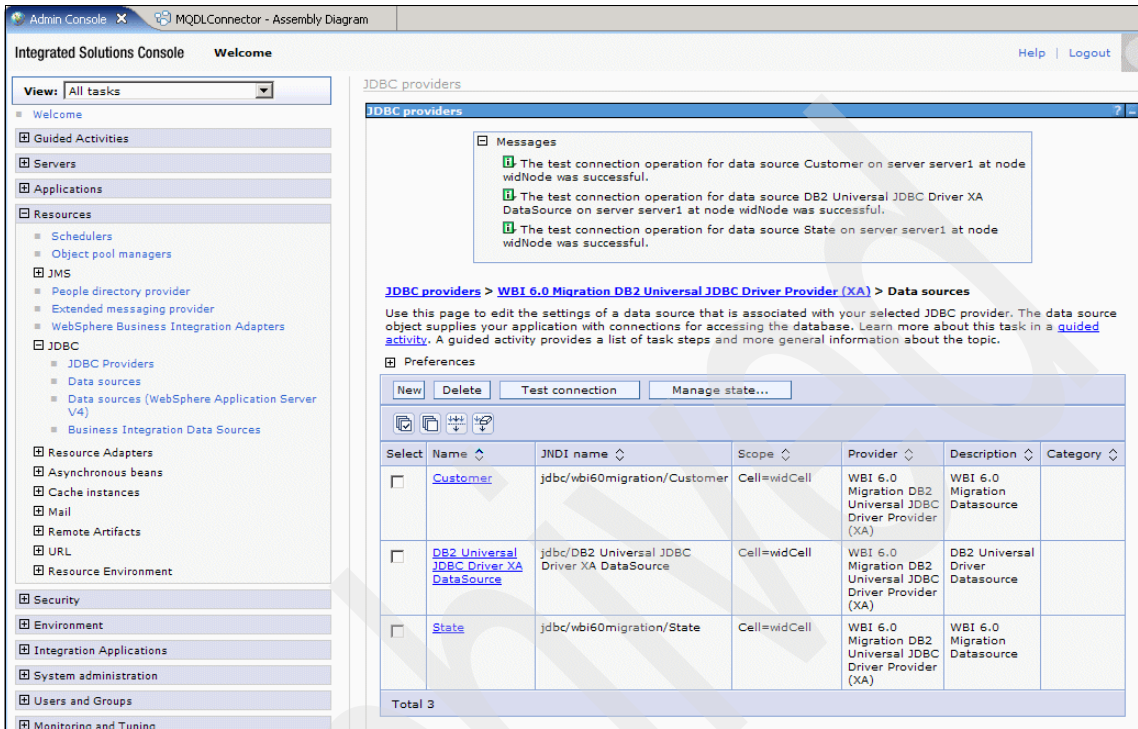


Figure 17-52 Data source connection testing

17.2.6 Deployment

After completing the steps to migrate the data synchronization scenario from WebSphere InterChange Server to WebSphere Integration Developer, the integration system is ready to be deployed to WebSphere Process Server.

Automatic project build: By default, WebSphere Integration Developer is set to build projects automatically. If the workspace is not set to build automatically, run the Clean and Build functions from the Project menu before deploying the modules.

To deploy the components:

1. From WebSphere Integration Developer, stop and start the embedded instance of WebSphere Process Server. Under the **Servers** tab, right-click the instance, and select **Start** (Figure 17-53).

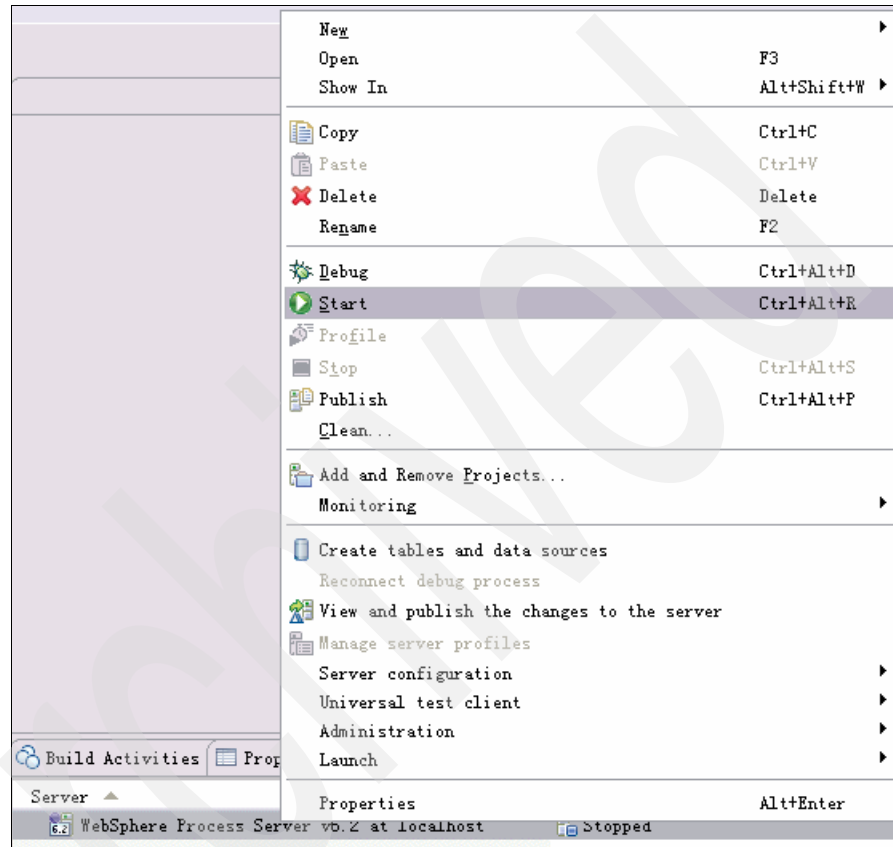


Figure 17-53 Starting WebSphere Process Server

2. Disable the default security settings in WebSphere Process Server to simplify the deployment and testing of the data synchronization scenario:
 - a. Launch the WebSphere Process Server administrative console.
Right-click an embedded instance of WebSphere Process Server, and select **Run** (Figure 17-54). Log in by using the default administrative user and password.

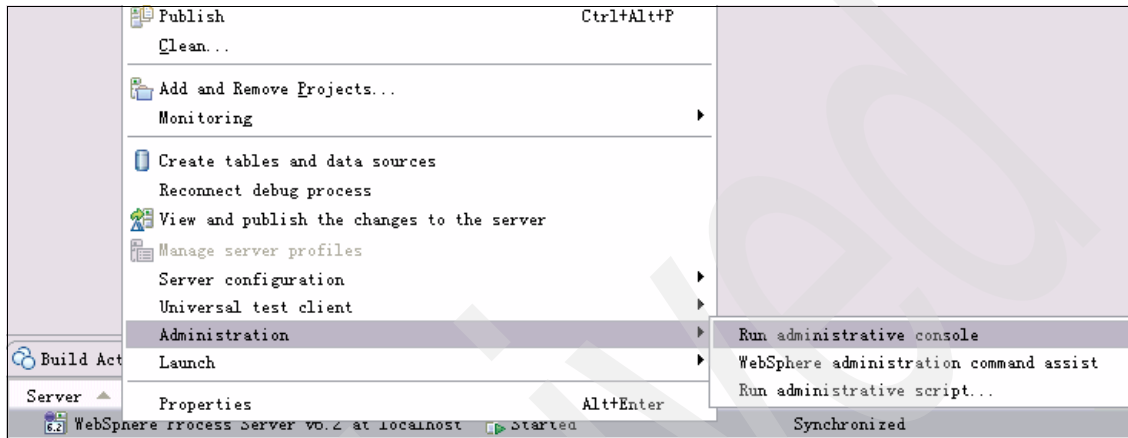


Figure 17-54 Running the administrative console

- b. In the left pane, under Security, choose **Secure administration, applications, and infrastructure**. In the right pane, clear the **Enable administrative security**, **Enable application security**, and **Use Java 2 security to restrict application access to local resources** check boxes (Figure 17-55). Click **Apply**, and save the changes.

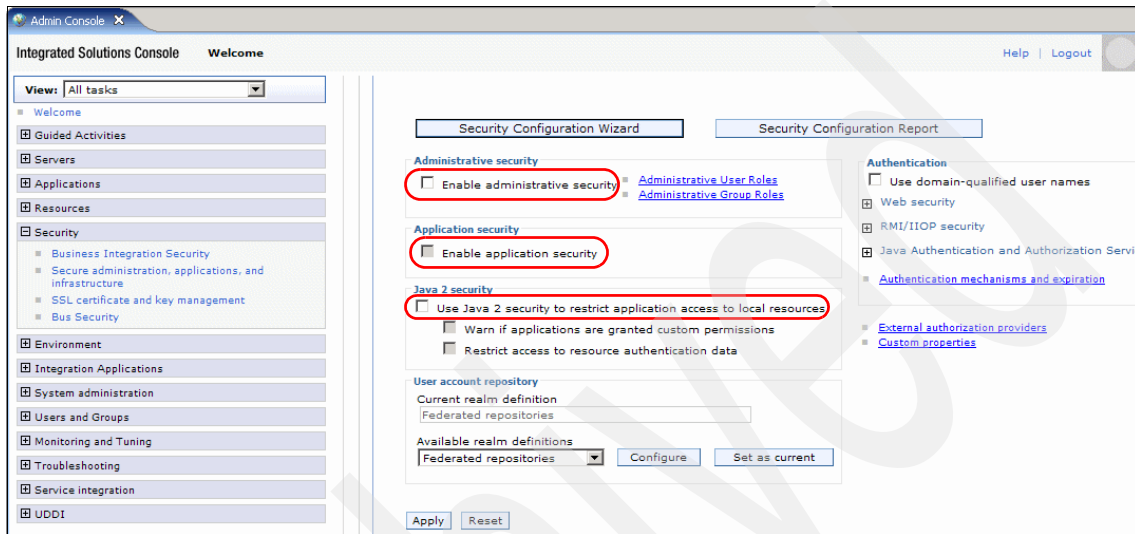


Figure 17-55 Disabling administrative security

- c. In the left pane, under Security, select **Bus Security**. In the right pane, disable security for all of the buses. Click **Apply**, and save the changes (Figure 17-56).

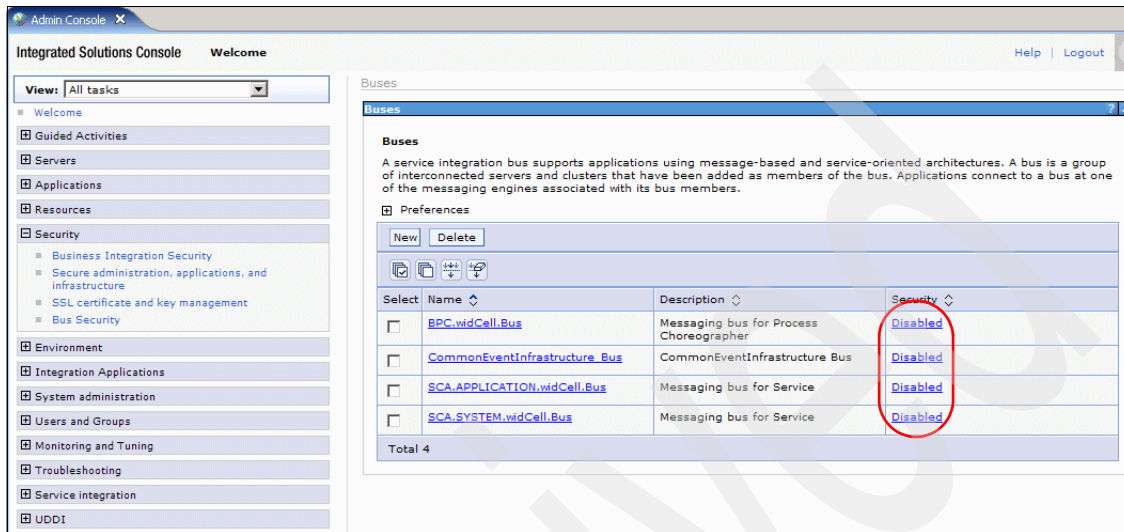


Figure 17-56 Disabling bus security

- d. Log out of the WebSphere Process Server administrative console, and close the console window.

- e. Open the profile configuration for WebSphere Process Server. From the **Servers** tab in WebSphere Integration Developer, right-click the appropriate WebSphere Process Server instance, and select **Open** (Figure 17-57).

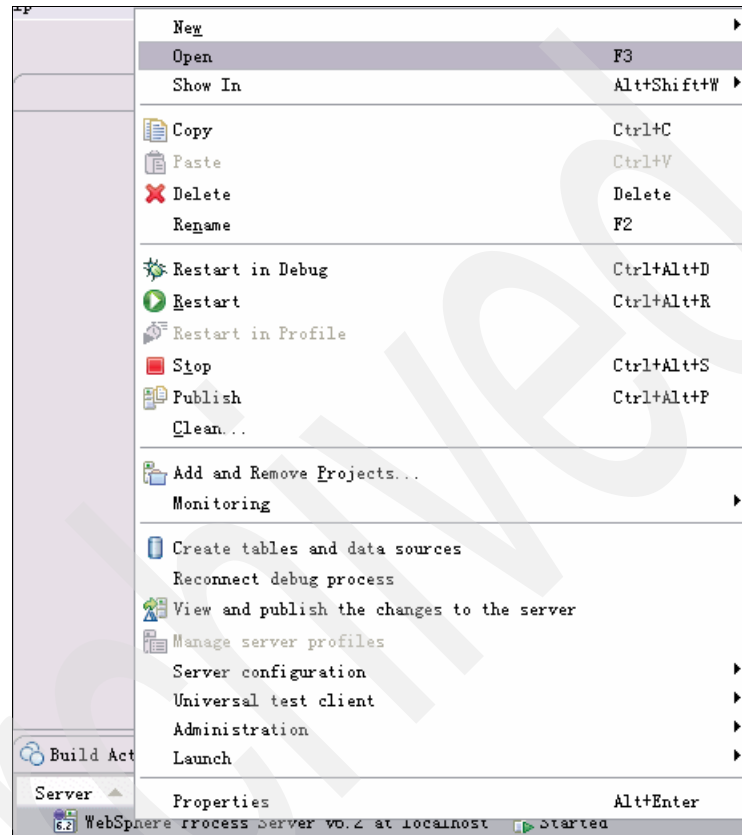


Figure 17-57 Opening the profile configuration

- f. Disable the security settings in the profile configuration. Under the Security section, clear the **Security is enabled on this server** check box (Figure 17-58). Close the profile configuration window, and save the settings.

The screenshot shows the 'Server Overview' configuration window for 'WebSphere Process Server v6.1'. The 'Security' section is expanded, and the checkbox 'Security is enabled on this server' is unchecked. The 'Publishing' section shows 'Run server with resources within the workspace' selected. The 'Automatic Publishing' section is also visible. The 'Server' section shows 'WebSphere profile name' as 'wps' and 'Update server status interval' as '5000'. The 'Server connection type and admin port' section shows 'SOAP (Designed to be more firewall compatible)' selected with a port of '8880'. The 'Network Deployment' section is also visible.

Server Overview

General
Specify the host name and other common settings.
Server name: WebSphere Process Server v6.1
Host name: localhost
Runtime: WebSphere Process Server v6.1 [Edit](#)

Server
Enter settings for the server.
WebSphere profile name: wps
Update server status interval (in milliseconds): 5000
Server connection type and admin port:
☐ RMI (Designed to improve communication with the server)
ORB bootstrap port: 2809
☒ SOAP (Designed to be more firewall compatible)
SOAP connector port: 8880
[Troubleshooting tips when the workbench shows the incorrect server status](#)
☒ Enable universal test client
☒ Optimize server for testing and developing
☐ Terminate server on workbench shutdown

Automatic Publishing

Publishing
Modify the publishing settings.
☒ Run server with resources within the workspace
☐ Run server with resources on Server
☒ Minimize application files copied to the server

Security
Enable and setup security.
☐ **Security is enabled on this server**
Current active authentication settings:
User ID: admin
Password: *****
☒ Automatically trust server certificate during SSL handshake

Network Deployment

Overview

Build Activities Properties Problems Servers Console

Server	Status	State
WebSphere ESB Server v6.1	Stopped	Republish
WebSphere Process Server v6.1	Started	Synchronized

Figure 17-58 Disabling profile security

- g. Restart the embedded instance of WebSphere Process Server (Figure 17-59). From the **Servers** tab, right-click the instance, and select **Restart**.

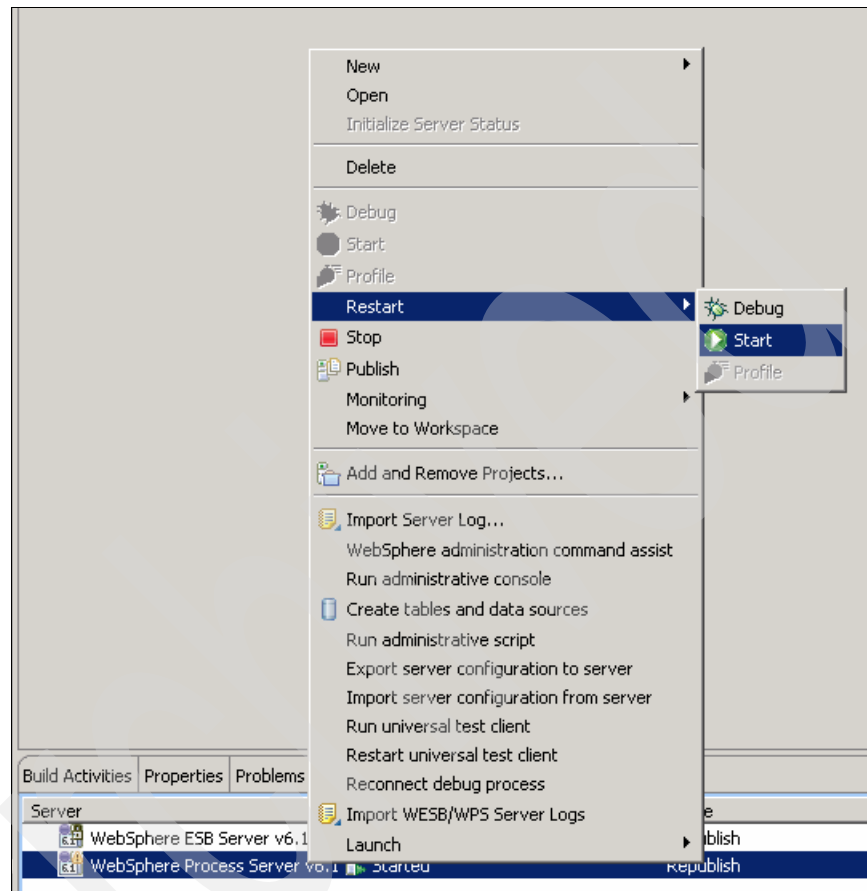


Figure 17-59 Restarting the WebSphere Process Server

3. Deploy the projects in the workspace. From the **Servers** tab in WebSphere Integration Developer, right-click the appropriate WebSphere Process Server instance, and select **Add and Remove Projects** (Figure 17-60).

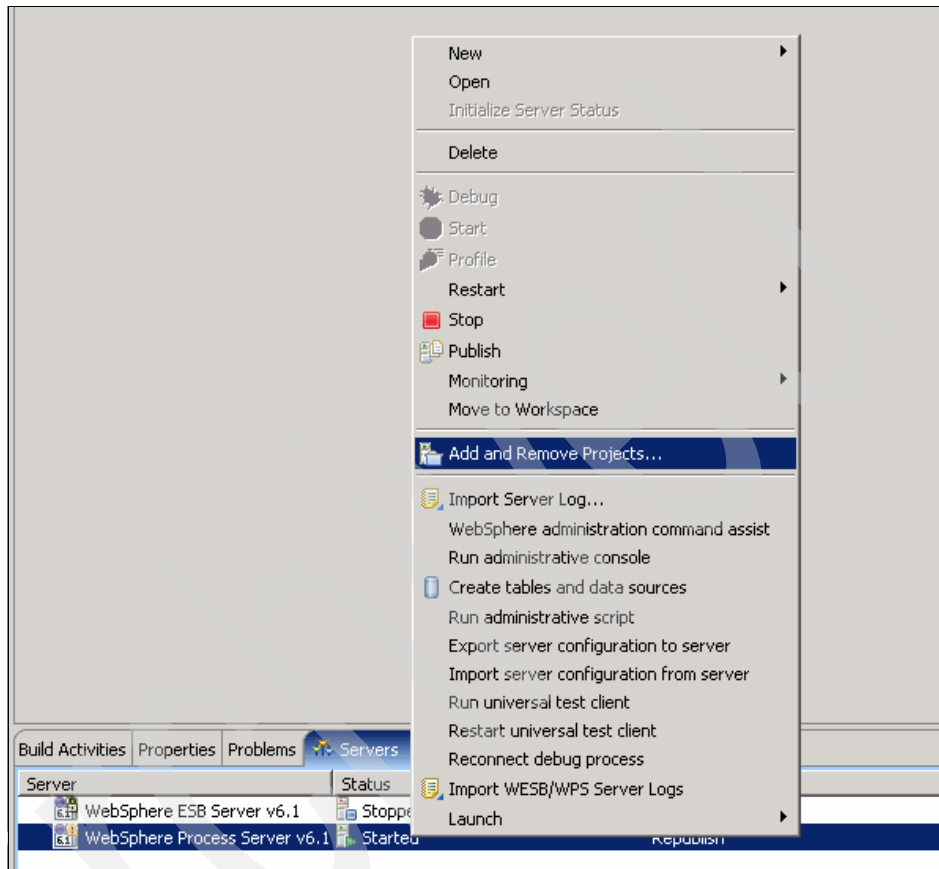


Figure 17-60 Adding projects to WebSphere Process Server

4. In the Add and Remove Projects window (Figure 17-61), click **Add All >>** to add all of the projects in the workspace to the server. Then, click **Finish** to initiate the deployment.

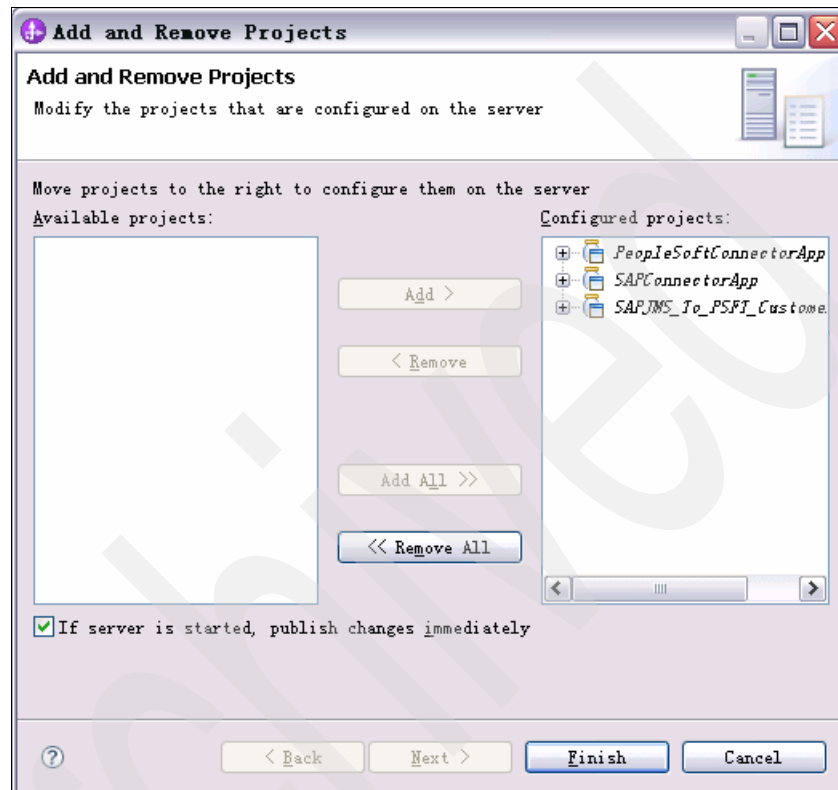


Figure 17-61 Deploying to WebSphere Process Server

5. Verify that all of the modules are deployed and that they have started correctly in WebSphere Process Server. Expand the embedded WebSphere Process Server instance to show all of the deployed modules. After the deployment has finished, the modules all show a status of Started (Figure 17-62).

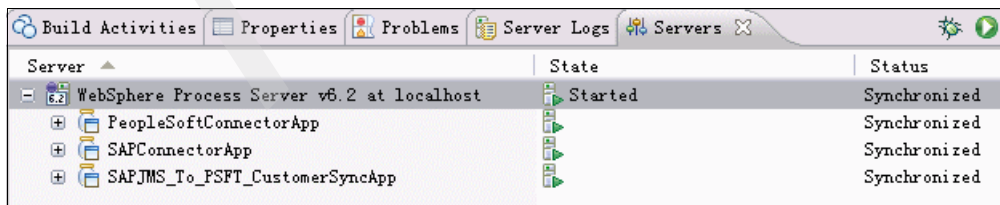


Figure 17-62 Verifying that all modules started

6. Before you test the actual applications, ensure that the relationships have migrated correctly:
 - a. Restart the embedded instance of WebSphere Process Server. From the **Servers** tab, right-click the instance, and select **Restart** → **Start**.
 - b. After the server is started, open the administrative console. On the **Servers** tab, right-click the running server, and select **Run administrative console**.
 - c. Click **Log in**. Because security is turned off, there is no need to provide any user ID.
 - d. In the left pane, navigate to **Integration Applications** → **Relationship Manager** → **Relationships**. The two migrated relationships are displayed in the right panel (Figure 17-63).
 - e. Select the **State** relationship, and click **Query**.

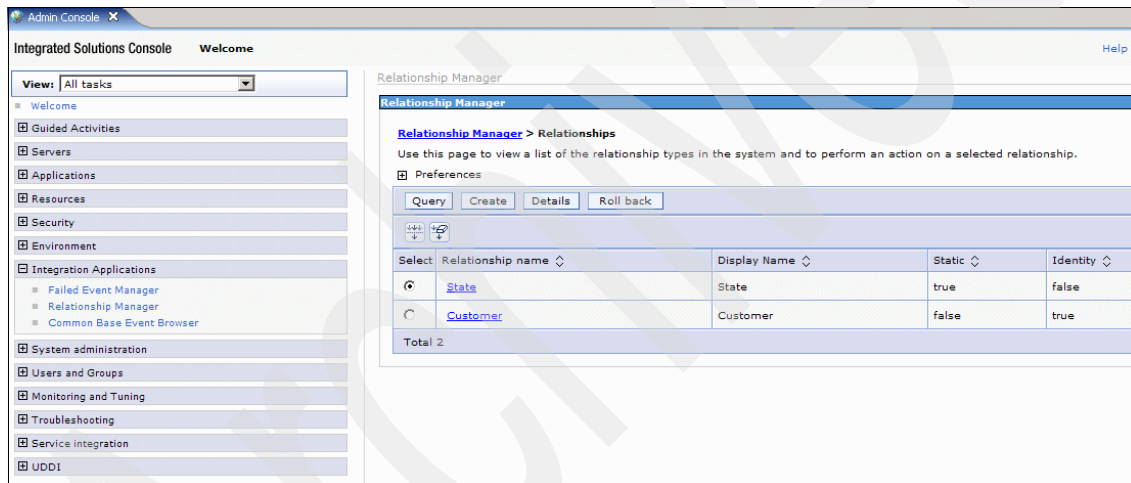


Figure 17-63 Verifying the migrated relationships

- f. In the next window, click **OK**. If state static relationship data was populated as a premigration step in Chapter 14, “Preparation for the technical solutions” on page 279, the query returns one instance as shown in Figure 17-64. If the query returns no rows, it is possible that the relationship data is not populated.

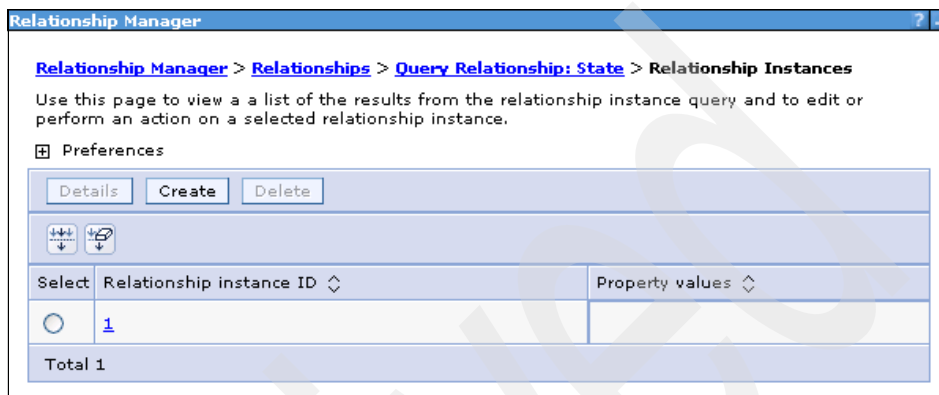


Figure 17-64 Static relationship instance data in Relationship Manager

7. If relationship data is not populated, download the scripts (STATE_STATECD_T_InsertScript.sql and STATE_STATENM_T_InsertScript.sql), as explained in Appendix D, “Additional material” on page 651, to populate this static relationship. Then, extract the scripts to the c:\setuptemp folder as explained in Chapter 14, “Preparation for the technical solutions” on page 279. Execute the scripts by copying and pasting the content of these files to the DB2 Command Center. Remember that these scripts only populate the tables. They do not create the tables. To create these tables, the WebSphere InterChange Server solution must be deployed to WebSphere InterChange Server successfully. Data can also be populated by using the version of Relationship Manager that comes with the WebSphere InterChange Server tools or that is packaged with the WebSphere Process Server administrative console. If any other errors occur while querying, you might have missed a migration step, and the migrated applications will not work.

Important: You must execute the Jython script prior to deploying the applications to the server for the first time. Also, after deploying the applications the first time and before testing the relationships as described previously, remember to restart the server.

17.3 Testing the end-to-end solution

To perform the end-to-end testing as explained in this section, you must have all of the required applications deployed and running on WebSphere Process Server.

Note: Ensure that you have one solution set (applications and adapters) running at a time and that your messages are not consumed by other adapters that are polling the same input queue.

To test the end-to-end solution:

1. Start SAP Front-End GUI Tool and log on (Figure 17-65).



Figure 17-65 SAP Front-End GUI Tool: Login

2. In the SAP Front-End GUI work panel, enter the transaction code /NWE02 in the command field, and press Enter (Figure 17-66 on page 498).

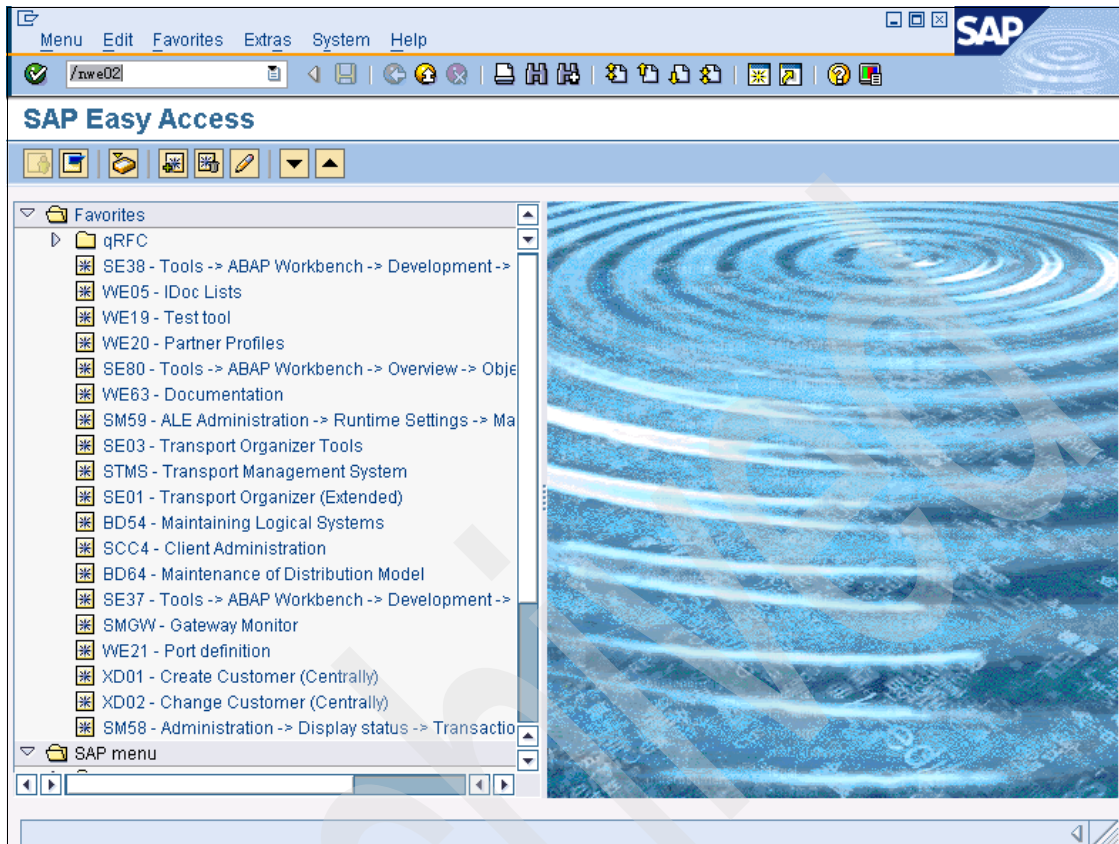


Figure 17-66 SAP Front-End GUI work panel

3. In NWE02, modify the IDoc Created date, and then, click **Execution** or press F8 to search the IDoc list (Figure 17-67).

The screenshot shows the SAP IDoc List interface. The window title is "IDoc List". The menu bar includes "Program", "Edit", "Goto", "System", and "Help". The toolbar contains various icons for file operations and search. Below the toolbar are three tabs: "Default", "Additional", and "EDI". The main area contains a form with various fields for filtering IDocs. The fields are arranged in a grid-like structure. The "Created on" field is highlighted with a yellow background. The "Created at" field is set to "00:00:00". The "Created on" field is set to "03/23/2009". The "Last Changed at" field is set to "00:00:00". The "Last Changed on" field is empty. The "Direction" field is empty. The "IDoc Number" field is empty. The "Current Status" field is empty. The "Basic Type" field is empty. The "Enhancement" field is empty. The "Logical Message" field is empty. The "Message Variant" field is empty. The "Message Function" field is empty. The "Partner Port" field is empty. The "Partner Number" field is empty. The "Partner Type" field is empty. Each field has a "to" field and a search icon.

Field	Value	to	Value	Search Icon
Created at	00:00:00	to	24:00:00	Yes
Created on	03/23/2009	to	03/26/2009	Yes
Last Changed at	00:00:00	to	24:00:00	Yes
Last Changed on		to		Yes
Direction				No
IDoc Number		to		Yes
Current Status		to		Yes
Basic Type		to		Yes
Enhancement		to		Yes
Logical Message		to		Yes
Message Variant		to		Yes
Message Function		to		Yes
Partner Port		to		Yes
Partner Number		to		Yes
Partner Type		to		Yes

Figure 17-67 IDoc List

4. Select one ORDERS IDoc from the IDoc List. Right-click the IDoc, and select **Copy Text** (Figure 17-68).

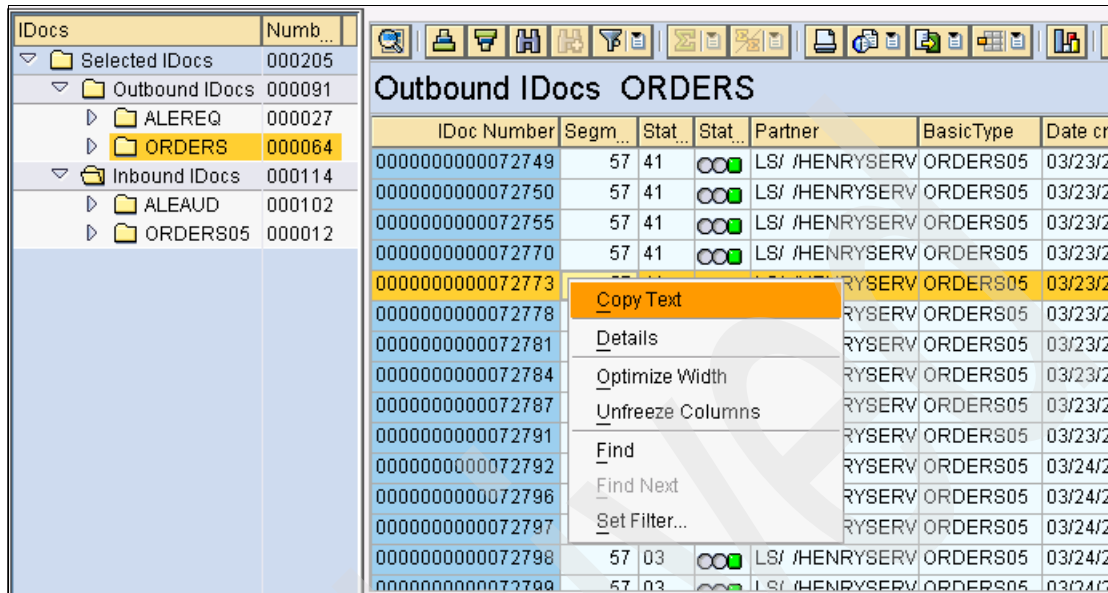


Figure 17-68 IDoc List: Search result

5. Enter the transaction code `/NWE19` in the command field, and press Enter (Figure 17-69).

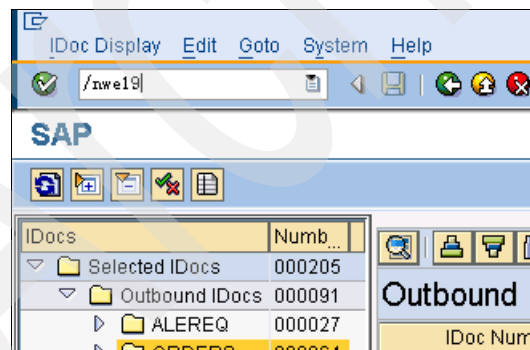


Figure 17-69 SAP Front-End GUI Tool work panel: Navigate

6. In NWE19, paste the IDoc into the Existing IDoc field, and click **Create** (Figure 17-70 on page 501).

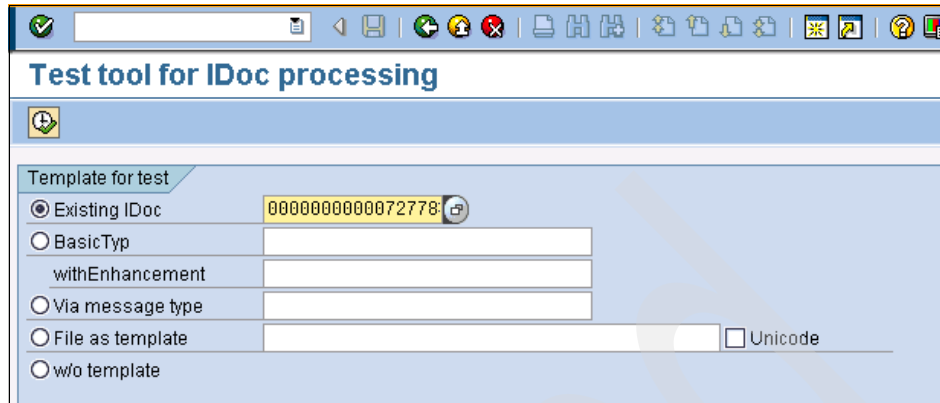


Figure 17-70 Search IDoc

7. Edit the IDoc, and enter the required value (Figure 17-71).

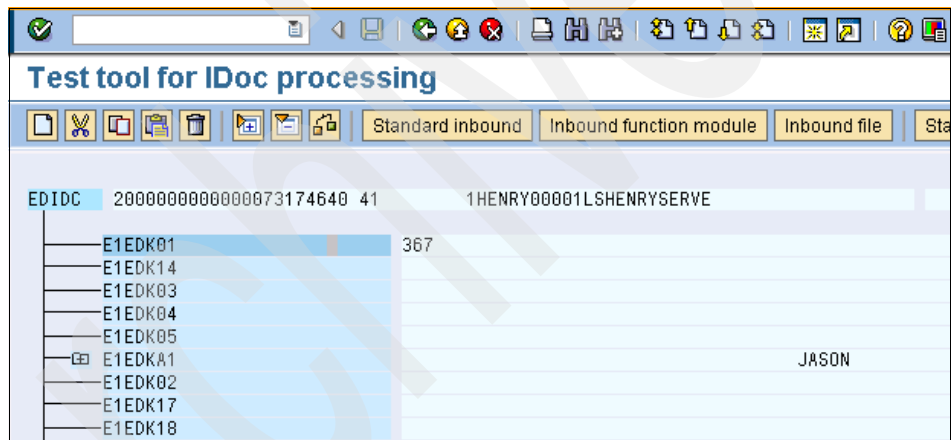


Figure 17-71 Edit the IDoc

8. After editing the IDoc, click **Standard Outbound Processing** to send the IDoc, and enter the IDoc quantity that you want sent (Figure 17-72 on page 502).

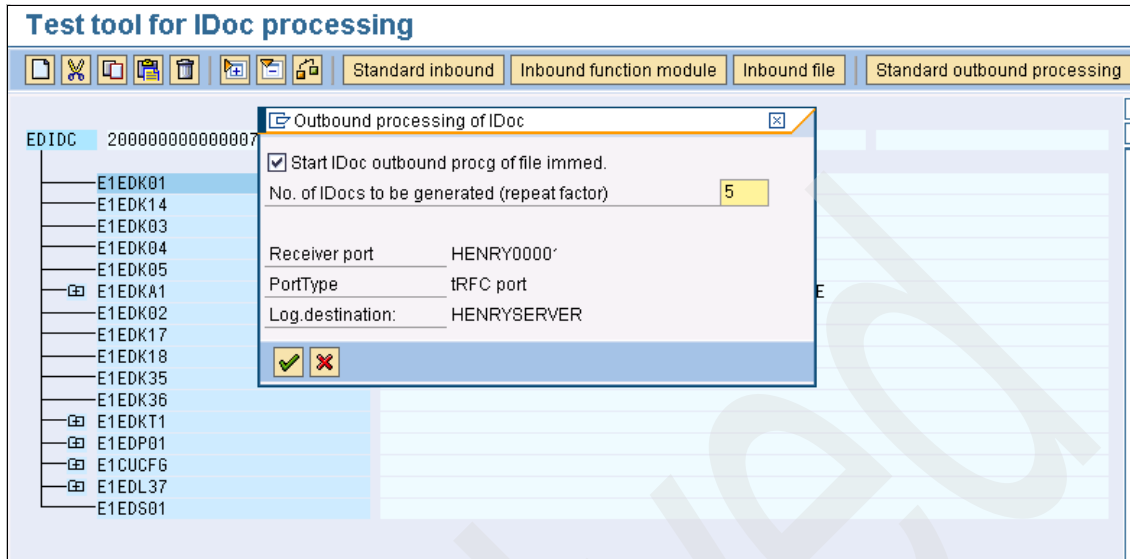


Figure 17-72 Send the IDoc

9. In the command field, enter the transaction code /NSM58, and press Enter (Figure 17-73).

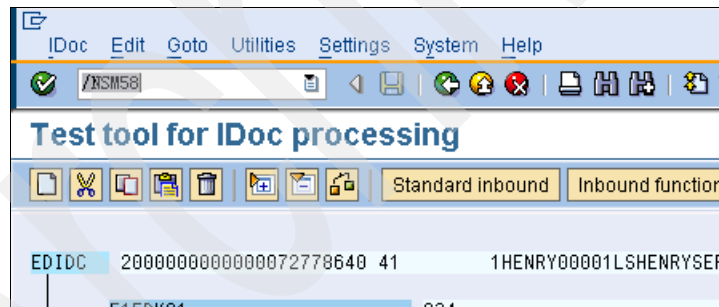


Figure 17-73 SAP Front-End GUI Tool work panel: Navigate

10. In NSM58, you can find all of the IDocs that you have sent. Right-click one and select **Execute LUW** (Figure 17-74 on page 503). Then, the test event will be sent out, and the WebSphere Adapter for SAP Software inbound listener will catch the event.

Transactional RFC				
Refresh				
Caller	Function Module	Target System	Date	Time
SONGHAN	TRFC_THROUGH_ASYNCRONOUS	HENRYSERVER	03/27/2009	00:04:12
SONGHAN	Help	F1	HENRYSERVER	03/27/2009
SONGHAN	Choose	F2	HENRYSERVER	03/27/2009
SONGHAN	Back	F3	HENRYSERVER	03/27/2009
SONGHAN	Possible Entries	F4	HENRYSERVER	03/27/2009
SONGHAN	Execute LUW	F6	HENRYSERVER	03/27/2009
SONGHAN	Refresh	F8	HENRYSERVER	03/27/2009
	Display Scheduled Jobs	F9		
	Cancel	F12		
	Delete Entry	Shift+F2		

Figure 17-74 Execute Test Event

11. To verify the success of the test, check the contents of the PeopleSoft database and the dynamic relationship participant tables:

Remember: Change the test data for every subsequent test that is run, especially the customer ID in the case of a create operation.

- a. Confirm that the row was created in the PeopleSoft database. Open **PS_WBI_CUSTOMER**, **PS_WBI_ADDRESS**, and **PS_WBI_PHONE** in the Oracle Enterprise Manager Console. Then, view the contents of the table (Figure 17-75).

The screenshot displays three overlapping Oracle Enterprise Manager Table Editor windows. The top window shows the PS_WBI_CUSTOMER table with one row containing customer ID 367 and name JASON. The middle window shows the PS_WBI_ADDRESS table with one row for customer 367, address line 1 'DONGBEIWANG', and address line 2 'BJ'. The bottom window shows the PS_WBI_PHONE table with one row for customer 367, address line 1 'DONGBEIWANG', address line 2 'BJ', and phone number 82450000.

WBI_CUSTOMER_ID	WBI_CUSTOMER_FNAME	WBI_CUSTOMER_LNAME	WBI_CUSTOMER_DOB
367	JASON		

WBI_CUSTOMER_ID	EFFDT	WBI_ADDRESS_ID	WBI_ADDRESS_LINE1	WBI_ADDRESS_LINE2	WBI_ADDRESS_CITY
367	02-Apr-2009 12:00:00 AM	367	DONGBEIWANG		BJ

WBI_CUSTOMER_ID	EFFDT	WBI_ADDRESS_ID	WBI_PHONE_ID	WBI_PHONE_NUMBER	WBI_PHONE_CITY
367	02-Apr-2009 12:00:00 AM	367	367	82450000	

Figure 17-75 Opening and viewing the PS_WBI_CUSTOMER, PS_WBI_ADDRESS, and PS_WBI_PHONE data

- b. Confirm that the identity relationship instances are recorded properly:
 - i. Run the administrative console and log in.
 - ii. Navigate to **Integration Applications** → **Relationship Manager**. In the right pane, select **Relationships**. See Figure 17-76 on page 505.

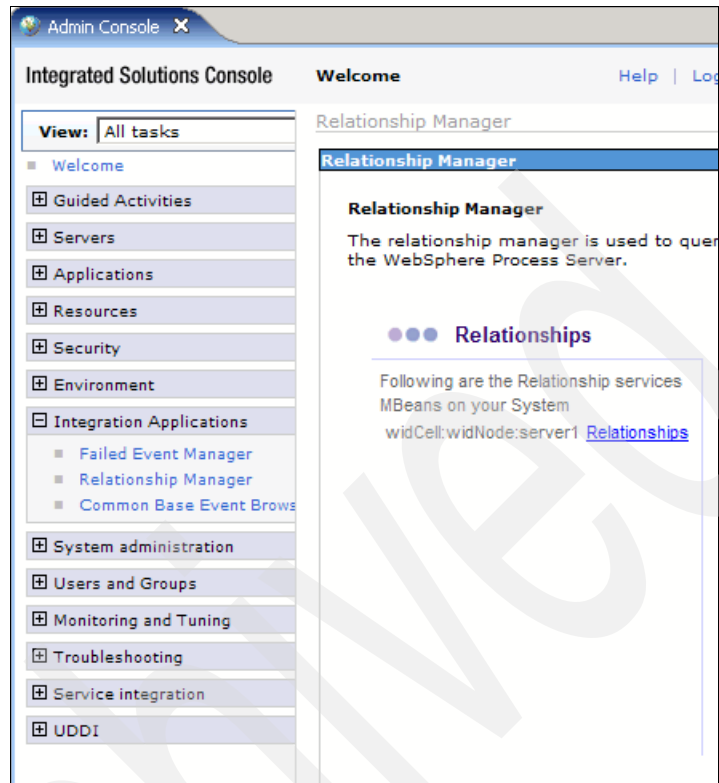


Figure 17-76 Selecting Relationship Manager in the administrative console

- iii. In the Relationships pane (Figure 17-77), select **Customer**, and click **Query**.



Figure 17-77 Querying the customer relationship

- iv. Confirm that the Logical state is set to **All**, and click **OK** (Figure 17-78).

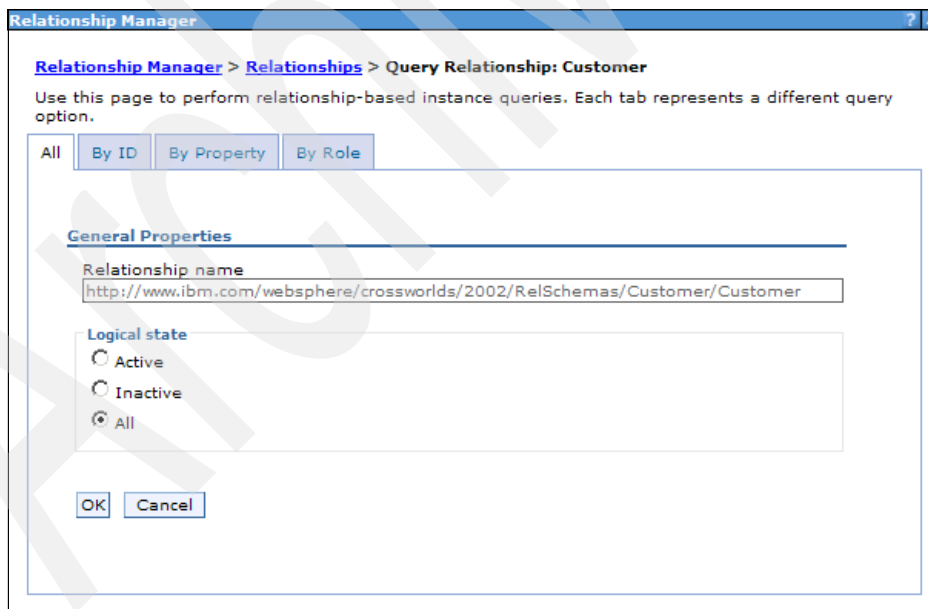


Figure 17-78 Querying all of the existing customer relationship instances

- v. Identify the generic CustomerID that is used in the console while the business process executed, and select the Relationship instance ID that corresponds to the generic CustomerID (Figure 17-79).

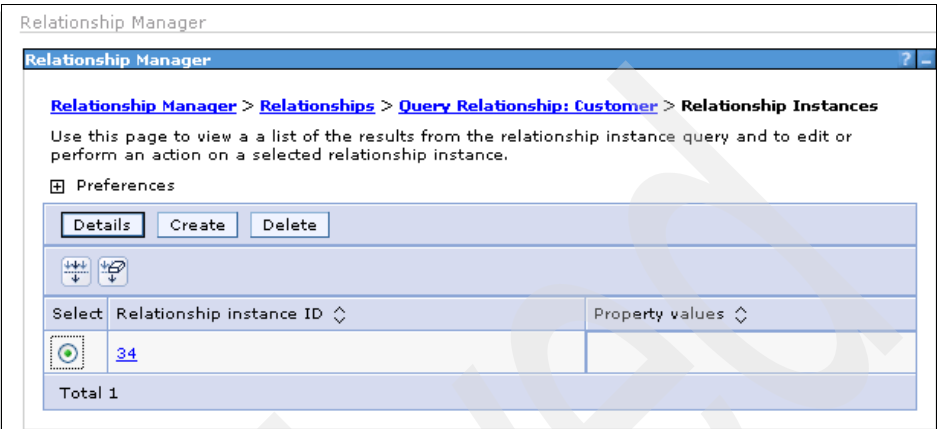


Figure 17-79 Selecting the Relationship instance ID created in the test run

- vi. Confirm that the SAPOrder role Key Attribute ID is set to the original test data value of 367, which is submitted by the SAP Front-End GUI Tool, as shown in Figure 17-80 on page 508. Also, confirm that the

PSFTCust role Key Attribute CUSTOMERID is set to the value that was created in the PS_WBI_CUSTOMER table. The table is shown in Figure 17-75 on page 504.

Relationship Manager

[Relationship Manager](#) > [Relationships](#) > [Query Relationship: Customer](#) > [Relationship Instances](#) > **Relationship Instance: 34**

Use this page to view detailed information for the selected relationship instance and to edit or perform an action on the relationship instance.

General Properties

Relationship name
http://www.ibm.com/websphere/crossworlds/2002/RelSchemas/Customer/Customer

Relationship instance ID
34

Property values
No properties defined

Role Instances

Cust

Select	ID	Logical State	Key Attributes	Property Values
<input type="button" value="Create"/>	<input type="button" value="Delete"/>			

SAPOrder

Select	ID	Logical State	Key Attributes	Property Values
<input checked="" type="radio"/>	<u>1</u>	Active	Dummy_key=367	
<input type="button" value="Create"/>	<input type="button" value="Delete"/>			

PSFTCust

Select	ID	Logical State	Key Attributes	Property Values
<input checked="" type="radio"/>	<u>1</u>	Active	WBI_CUSTOMER_ID=367.000000	

Figure 17-80 Dynamic relationship instance data in Relationship Manager

Note: A valid CustomerID value is required to create a customer in the data synchronization scenario. In addition, the value must be unique. So, subsequent tests must use different CustomerID values.

17.4 Conclusion

In this chapter, we illustrated a step-by-step migration of a data synchronization solution. The migrated WebSphere InterChange Server solution involved two adapters for back-end connectivity. The migration started by using the WebSphere Integration Developer Migration Wizard to get the project working as is in a WebSphere Process Server environment.

We also performed an end-to-end test to check the functionality of the migrated components after each step.

Customize and enhance the solutions

In this chapter, we present various ways to enhance a WebSphere Process Server solution coming from a WebSphere InterChange Server solution that was migrated with standard tools. We describe various options that add value to the migrated solution, either by providing new capabilities or raising the quality and the maintainability of the solution. We include the following sections:

- ▶ 18.1, “Enhanced error handling” on page 512
- ▶ 18.2, “Enhanced routing capability” on page 535
- ▶ 18.3, “Enhanced migration for artifacts designed visually” on page 551

Note: This chapter assumes the reader is already used to the standard migration tools and is acquainted with WebSphere InterChange Server, WebSphere Process Server concepts, and WebSphere Integration Developer tools. We recommend that you read Chapter 9, “Migration implementation approach” on page 151 in advance, because it provides useful context information for the present chapter.

18.1 Enhanced error handling

In this scenario, we demonstrate how to implement enhanced error handling in WebSphere Process Server, using Human Tasks. *Human Tasks* are an extremely efficient way to get the business users directly involved in the process execution. The purpose of this example is to enhance the migrated process so that the business users can take ownership of the functional error management. This way, the business can react more quickly than before to process issues, which adds value to the integration solution.

For the testing and validation, we assume that the WebSphere Process Server is your current working environment. We also assume that the corresponding databases, message queues, and adapters to run the scenario are installed and configured correctly.

For more information, see Part 2, “Migration implementation concepts” on page 71, for an overall discussion about the available options for migration implementation. In addition, see Part 3, “Migration tooling” on page 177, for detailed information about the standard migration tools, and see Part 4, “End-to-end technical solutions” on page 277, which illustrates the migration of individual artifacts.

We include the following sections:

- ▶ 18.1.1, “Target environment” on page 512
- ▶ 18.1.2, “Error handling example” on page 513

18.1.1 Target environment

In this section, we list the various software products that are involved in this example. Table 18-1 on page 513 lists the software products and minimal versions that are required to demonstrate the example in this chapter. It also indicates the products that are installed in the target environment.

Table 18-1 Software versions used

Software	Version
WebSphere Integration Developer	6.2
WebSphere Process Server	6.2
WebSphere MQ	6.0.2 Fix Pack 2
WebSphere InterChange Server	4.3.0 Fix Pack 5
DB2	8.1.14.292
WebSphere Business Integration Adapter Framework	2.6.0.11
WebSphere Business Integration Adapter JText	5.6.5
WebSphere Business Integration Adapter JDBC	2.6.9
WebSphere Business Integration Adapter WebSphere MQ	2.8.2
WebSphere Business Integration Adapter Web Services	3.4.7
XML DataHandler	2.7.0

18.1.2 Error handling example

WebSphere Process Server provides a richer set of functionalities to implement integration patterns, compared to WebSphere InterChange Server. WebSphere Process Server consists of various powerful components, such as WebSphere Application Server, Service Integration Bus, Service Component Architecture (SCA), and business process choreography engine. As a result, you can implement a more powerful error handling mechanism in WebSphere Process Server.

In WebSphere Process Server, if an error occurs in an asynchronous service invocation, the failed service request message is persisted. It can be persisted in three separate destinations, depending on where the error occurs:

- **Messaging-based dead letter queues**

A messaging-based dead letter queue can happen if a Java Message Service (JMS) binding is used. The failed service request message is then stored in the internal messaging layer (Service Integration Bus). A user, with a strong IT profile typically, can view the message in the Service Integration Bus and replay it using a dedicated tool.

► Failed event

A *failed event* can happen if an error occurs in an asynchronous service invocation between two SCA components. The failed service request message is then stored in specific tables, similar to the work-in-progress tables of WebSphere InterChange Server. The WebSphere Process Server component, Failed Event Manager, manages failed request messages. With this component, users can view, modify, and resubmit them. You can find Failed Event Manager in the administrative console of WebSphere Process Server. This console is still targeted for IT profiles.

► Business process instances in a failed state

Business process instances in a failed state can happen if an error occurs in the execution of a long running business process. In this case, the failed service request message is stored in the Business Process Choreographer component. It can be resubmitted at the process instance or process activity level. The Business Process Choreographer Explorer is a console that can be used by both IT and Business profiles.

These error management facilities are provided in WebSphere Process Server. You can use them to manage errors (service requests) after they have failed because errors occurred during their execution. In Business Process Choreographer, you can handle the errors just in time, so that the service request does not fail.

In addition to the functionality of persisting and resubmitting failed service request messages, Business Process Choreographer provides the functionality, the *fault handler*, to handle errors that occurred in business process execution. The fault handlers can be defined in a business process for service invoke activities, scopes, and even the whole process. They can catch faults that are thrown in the scope in which they are defined. If a fault is caught and handled by a fault handler, the process flow can proceed with its normal activities, starting from the return point of the fault handler.

In the example, we demonstrate how to use the fault handler in a business process to implement enhanced error handling after migrating from WebSphere InterChange Server to WebSphere Process Server. We start from the data access scenario that was described in Chapter 15, “Data access scenario with technology adapters” on page 293, and we illustrate how to enhance it with a functional error handling mechanism using Human Tasks.

Here is a reminder about the data access scenario. It is a functional implementation of the typical data access pattern that is implemented in WebSphere InterChange Server. This scenario is restricted to two collaborations: CreateCustomerCollab and RetrieveCustomerCollab:

- The *Create Customer Collaboration* (CreateCustomerCollab) exposes a single inbound port (FromCreate) and two outbound ports (ToMQXML and ToJDBC). The ports are bound to instances of the WebSphere Business Integration Adapter for Web services, WebSphere Business Integration Adapter for WebSphere MQ, and WebSphere Business Integration Adapter for JDBC, as illustrated in Figure 18-1.

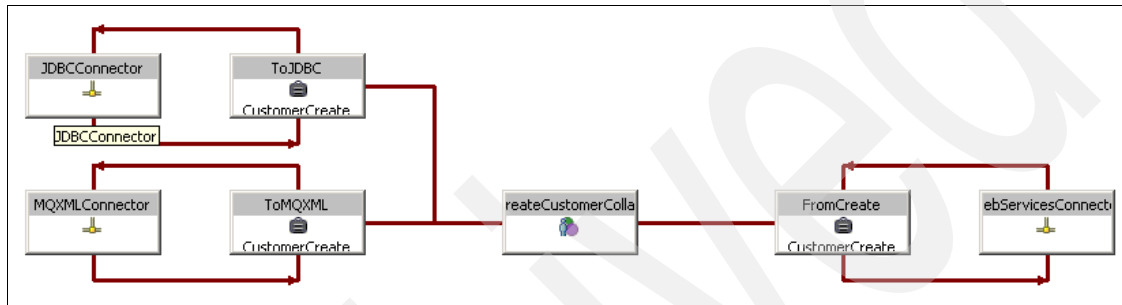


Figure 18-1 Create Customer Collaboration flow diagram

- The *Retrieve Customer Collaboration* (RetrieveCustomerCollab) exposes a single inbound port (FromRetrieve) and a single outbound port (ToJDBC). The ports are bound to instances of the WebSphere Business Integration Adapter for Web services and the WebSphere Business Integration Adapter for JDBC, as illustrated in Figure 18-2.



Figure 18-2 Retrieve Customer Collaboration flow diagram

When migrating the data access scenario from WebSphere InterChange Server to WebSphere Process Server and upgrading WebSphere Business Integration Adapter for JDBC to WebSphere Adapter for JDBC, the collaborations are converted in the following way to business processes by using the migration tools:

- CreateCustomerCollab to CreateCustomerCollab_FromCreate
- RetrieveCustomerCollab to RetrieveCustomerCollab_FromRetrieve

Our enhanced error handling mechanism example is going to be implemented in the RetrieveCustomerCollab_FromRetrieve business process (Figure 18-3 on page 517). In this process, the ToJDBC.Retrieve service invocation activity calls the WebSphere Adapter for JDBC. If the customerID attribute of the input business object equals the customerID column of a row in the DB2 database table, that row is retrieved. Otherwise, a fault is thrown out.

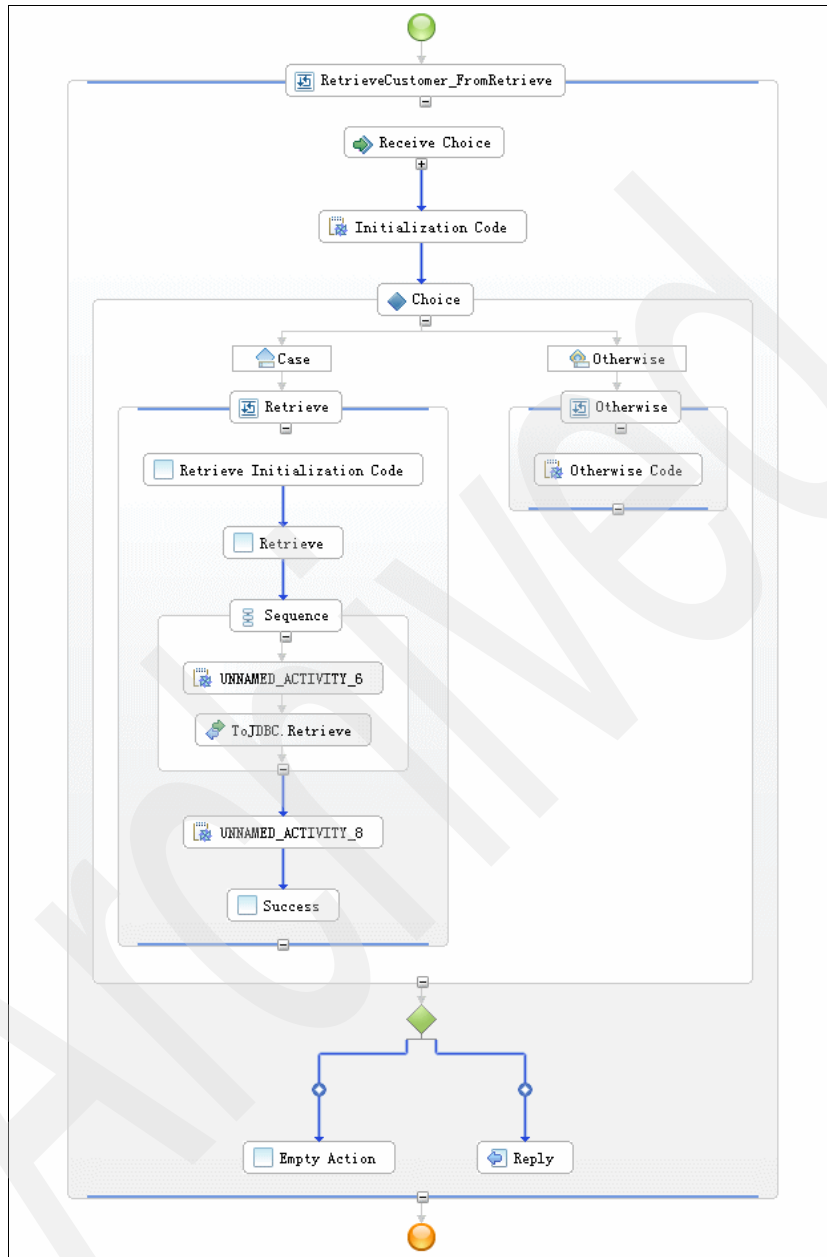


Figure 18-3 RetrieveCustomerCollab_FromRetrieve business process diagram

Enhanced error management

A fault handler is added to ToJDBC.Retrieve. If a fault is caught by this handler, it creates a new HumanTask. A business user with the correct credentials can claim this task in Business Process Choreographer Explorer. After claiming the task, the user can immediately check to see if the problem is caused by incorrect input data in the request. If it is, the business user can enter the correct data (customerID attribute in this example) and resubmit the request by completing the HumanTask. If the reason is more technical, the business user can request help from an IT user (to investigate the server's log file typically). After the root technical issue is solved (the database was temporarily unavailable, for example), the business user can resubmit the request by completing the HumanTask.

Note: Our example is obviously fictional, but its purpose is to show how you can add Human Tasks to an existing process to give control to business users. We chose the RetrieveCustomerCollab, because it was a simple collaboration to test because it is exposed as a Web service. We list the major reasons that this example is unrealistic:

- ▶ The process is triggered by a synchronous request-reply interaction (Web service). In the case of an error, the request is blocked until the business user completes the Human Task. This approach is unacceptable in a real scenario. To improve this example, you can add a time-out to provide better control; however, our example does not include a time-out.
- ▶ The business user is allowed to change the faulty customerID, which is not a functional error handling scenario. Again, our major purpose is to demonstrate that a business user can be alerted easily when incorrect or wrong data has been sent to the system (creation of a Human Task and a work item) and also that the business user can quickly and easily react to this problem (by completing the work item).

Figure 18-4 on page 519 illustrates our example.

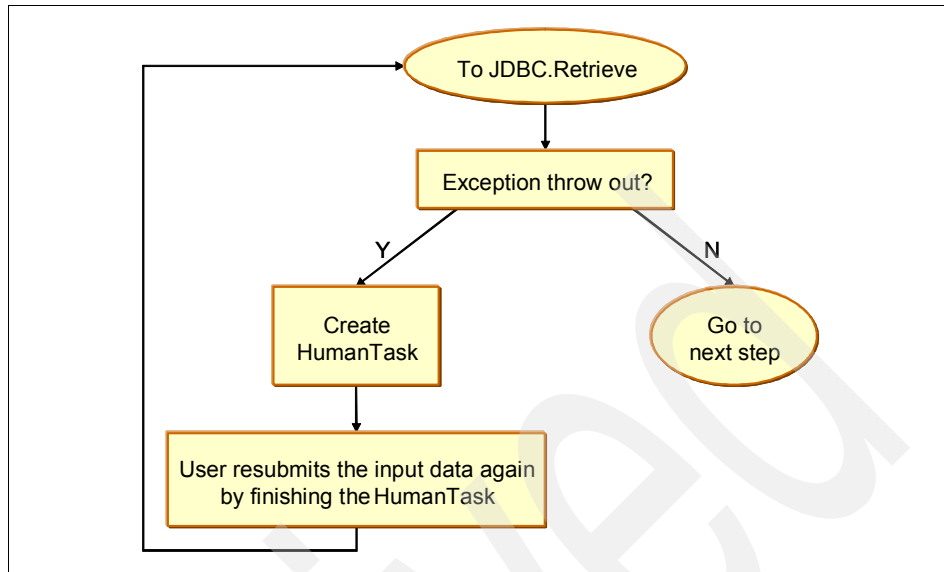


Figure 18-4 Fault handler on TOJDBC.Retrieve

Preparation

We assume that you have already migrated the data access scenario from WebSphere InterChange Server to WebSphere Process Server by following the instructions in 15.2, “Implementation” on page 295.

Development

Follow these steps to implement our enhanced error handling:

1. From the Business Integration view in WebSphere Integration Developer, expand the **RetrieveCustomerCollab** module, and open the associated **assembly diagram**.
2. In the RetrieveCustomerCollab assembly diagram, double-click the **RetrieveCustomerCollab_FromRetrieve** business process to open it as shown in Figure 18-3 on page 517.

Note: If the RetrieveCustomerCollab_FromRetrieve business process does not look as shown Figure 18-3, right-click the canvas and select **Expand All Activities**. If it still looks different with overlapping lines, right-click the canvas again, and select **Align Parallel Activities Contents Automatically**.

3. On the canvas of the RetrieveCustomerCollab_FromRetrieve Business Process Editor, select **Variables**, and click **Add Variable**.

4. In the Add Variable window (Figure 18-5), in the Name field, type `JDBCFault`, and for Matching data types, select **boolean**. Click **OK**.

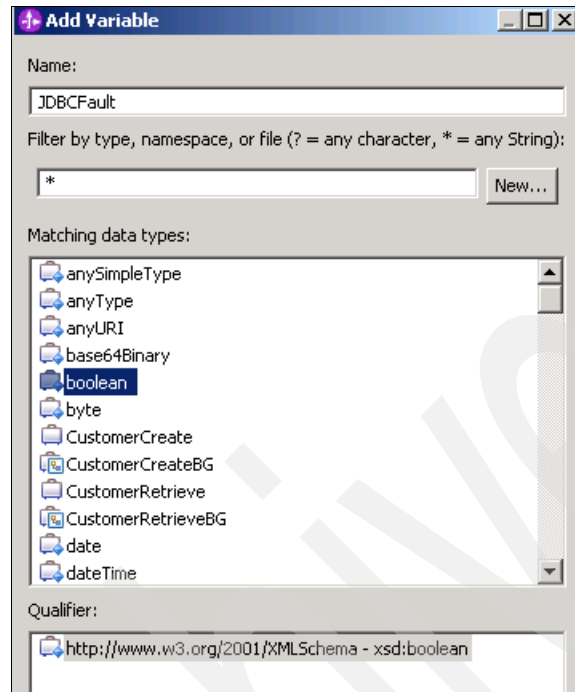


Figure 18-5 Adding the JDBCFault variable

5. Repeat step 3 on page 519. In the Name field, type `HTMRetrieveCustomerBG`, and for Matching data types, select **CustomerRetrieveBG**. Click **OK**.
6. Repeat step 3 on page 519. In the Name field, type `FaultDescription`, and for Matching data types, select **string**. Click **OK**.
7. In the Business Process Editor, click the **While Loop** icon on the Palette, and then, click the connection between **UNNAMED_ACTIVITY_6** and **ToJDBC.Retrieve**. The WhileLoop activity is added as shown in Figure 18-6 on page 521.

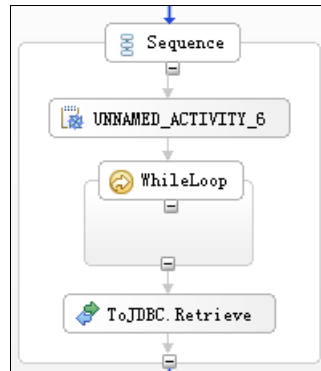


Figure 18-6 Adding the WhileLoop activity

8. In the Business Process Editor, click the **WhileLoop** activity. In the Properties view, click the **Details** tab. For Expression Language and Expression Type, select **Java**.
9. If the Question window (Figure 18-7) opens, click **Yes**.

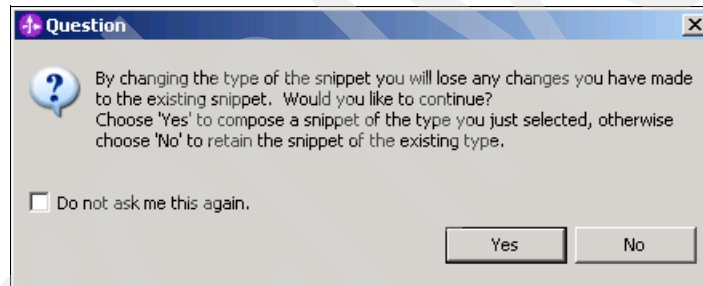


Figure 18-7 Question window prompting to change the type of snippet

10. In the Details window, enter the Java snippet as shown in Example 18-1.

Example 18-1 WhileLoop snippet

```
return JDBCFault;
```

11. In the Business Process Editor, drag the **ToJDBC.Retrieve** invoke activity to the **WhileLoop**.
12. In the Business Process Editor, click the **Snippet** icon on the Palette, and then, click the connection between **UNNAMED_ACTIVITY_6** and **WhileLoop**. The snippet activity is added.
13. In the Business Process Editor, click the snippet that you have added.

14. In the Properties view:

- a. Click the **Description** tab.
- b. Modify the snippet name and the display name to **setJDBCFault**.
- c. Click the **Details** tab.
- d. For the Implementation type, select **Java**.

15. If the Question window opens, click **Yes**.

16. In the text area, enter the Java code that is shown in Example 18-2.

Example 18-2 setJDBCFault snippet

```
JDBCFault = true;
```

17. In the Business Process Editor, click the **Snippet** icon on the Palette, and then click the connection between **ToJDBC.Retrieve** and **WhileLoop**. The snippet activity is added.

18. In the Business Process Editor, click the snippet that you have added.

19. In the Properties view:

- a. Click the **Description** tab.
- b. Modify the snippet name and display name to **ClearJDBCFault**.
- c. Click the **Details** tab.
- d. For the Implementation type, select **Java**.

20. If the Question window opens, click **Yes**.

21. In the text area, enter the Java code that is shown in Example 18-3.

Example 18-3 clearJDBCFault snippet

```
JDBCFault = false;
```

22. In the Business Process Editor, click the **ToJDBC.Retrieve** invoke activity, and click **Add Fault Handler** as shown in Figure 18-8. The default fault handler is **WICSFault**.

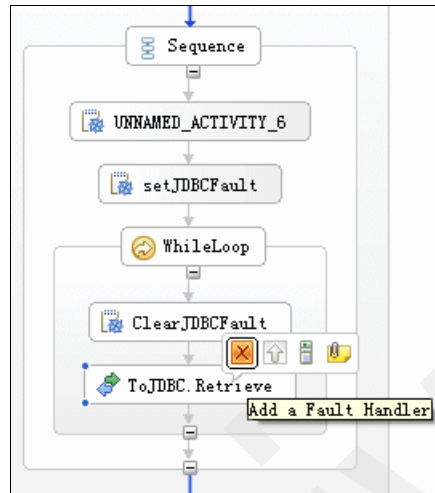


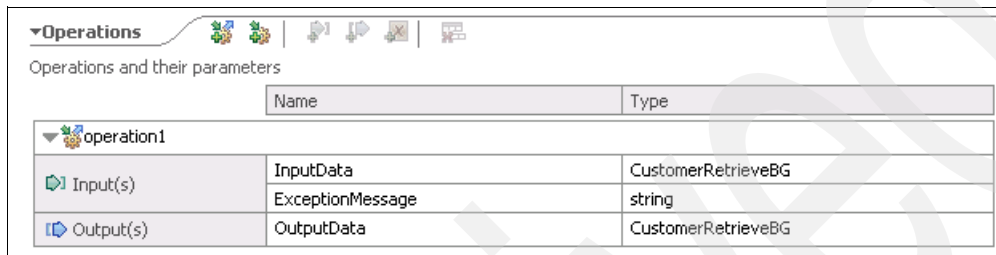
Figure 18-8 Adding a fault handler

23. In the Business Integration view, under the RetrieveCustomerCollab module, right-click **Interfaces**, and select **New** → **Interface**.
24. In the New Interface window, in the Name field, type **HTMInterface**, and then, click **Finish**.
25. In the Interface Editor, click **Add Request Response Operation**.
26. When you see **Operation1** with **input1** and **output1**, right-click **operation1**, and select **Add Input**.

27. When Input2 is displayed, make the following changes:

- Change the name of input1 to InputData and the message type of input1 to CustomerRetrieveBG.
- Change the name of input2 to ExceptionMessage and the message type of input2 to string.
- Change the name of output1 to OutputData and the message type of output1 to CustomerRetrieveBG.

Figure 18-9 shows the HTMInterface.



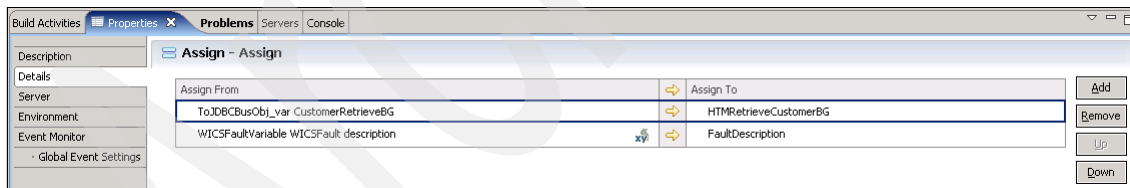
Operations and their parameters		
	Name	Type
operation1		
Input(s)	InputData	CustomerRetrieveBG
	ExceptionMessage	string
Output(s)	OutputData	CustomerRetrieveBG

Figure 18-9 HTMInterface

28. In the Business Process Editor, click the **Assign** icon on the Palette, and then, click **WICSFault** activity. The assign activity is added.

29. In the Business Process Editor, click the assign activity that you created.

30. In the Properties view, click the **Details** tab and change the Assign From and Assign To data objects as shown in Figure 18-10.



Assign - Assign		
Assign From	Assign To	
ToJDBCBusObj_var CustomerRetrieveBG	HTMRetrieveCustomerBG	Add
WICSFaultVariable WICSFault description	FaultDescription	Remove
		Up
		Down

Figure 18-10 Changing the Assign From and Assign To properties

31. In the Business Process Editor, click the **Human Task** activity icon on the Palette, and then, click the **Assign** activity that you created. The HumanTask activity is added.

32. In the Interface Selection window, select **HTMInterface**, and click **OK**. A Human Task called RetrieveCustomer_FromRetrieveTask1 is created, and the editor window opens automatically.

33. In the Business Process Editor, click the **HumanTask** activity that you have created.

34. In the Properties view, click the **Details** tab, and select **Use data type variables mapping**. Set the InputData and OutputData variables to **HTMRetrieveCustomerBG**. Set the ExceptionMessage variable to **FaultDescription**.
35. In the Business Process Editor, click the **Assign** icon on Palette, and then, click the **HumanTask** activity that you created previously. An assign activity called Assign1 is added.
36. In the Business Process Editor, click the **Assign1** activity that you created. In the Properties view, click the **Details** tab, and change the Assign From and Assign To data objects as shown in Figure 18-11.

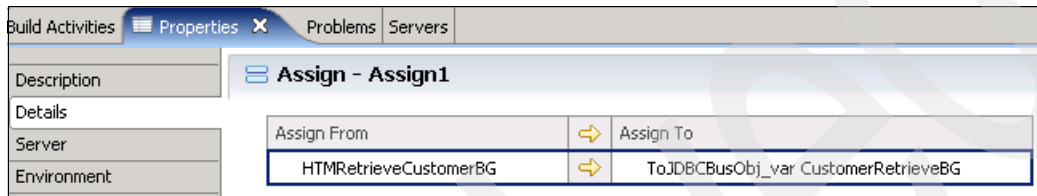


Figure 18-11 Assigning from HTMRetrieveCustomerBG to ToJDBCBusObj_var CustomerRetrieveBG

37. In the Business Process Editor, click the **Snippet** icon on the Palette, and then, click the **Assign1** activity. A snippet activity is added.
38. In the Business Process Editor, click the snippet that you have added.
39. In the Properties view:
 - a. Click the **Description** tab.
 - b. Modify the snippet name and the display name to **setJDBCFault1**.
 - c. Click the **Details** tab.
 - d. For the Implementation type, select **Java**.
40. If the Question window opens, click **Yes**.
41. In the text area, type the Java code that is shown in Example 18-4.

Example 18-4 setJDBCFault1 snippet

```
JDBCFault = true;
```

42. In the Business Process Editor, click the canvas outside of the process boundary.
43. On the **Details** tab of the Properties view, make sure that the **Process is long-running** check box is selected.

44. In the Assembly Diagram Editor, click the interface icon of the **RetrieveCustomerCollab_FromRetrieve** process (Figure 18-12).

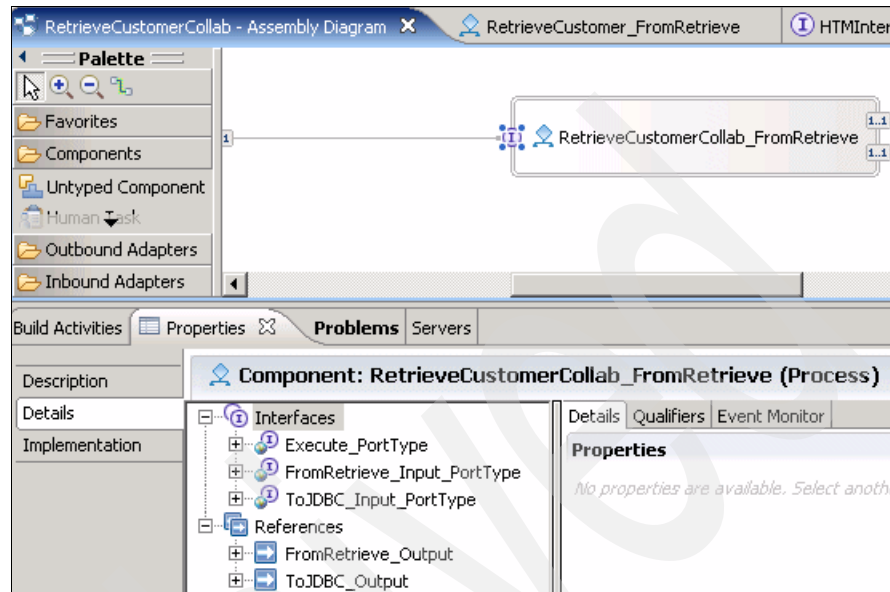


Figure 18-12 RetrieveCustomerCollab_FromRetrieve process interface

45. Click the **Execute_PortType** interface. Click the **Qualifiers** tab on the right and select **Join transaction**. Check the Value, which needs to be **False** (Figure 18-13).

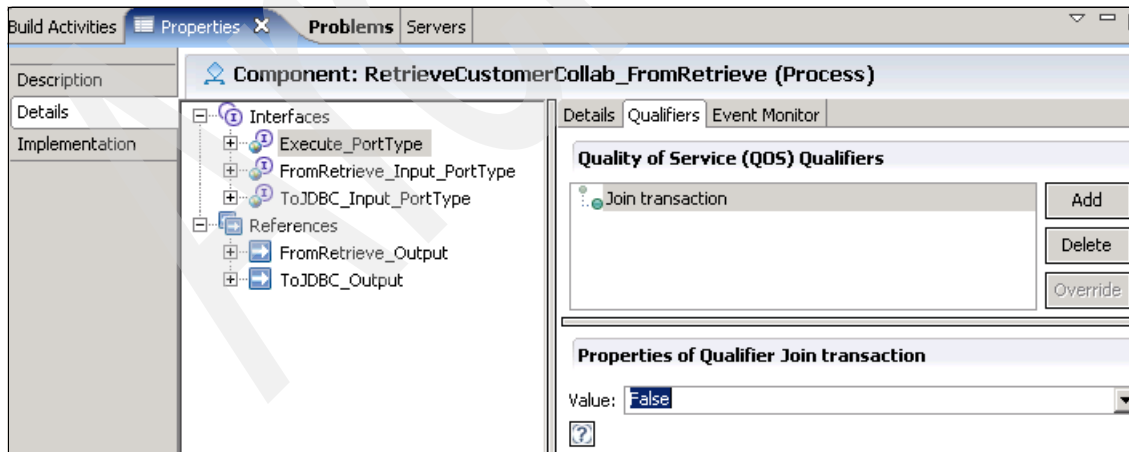


Figure 18-13 Execute_PortType interface

46. Repeat step 45 on page 526 to check the interfaces **FromRetrieve_Input_PortType** and **ToJDBC_Input_PortType**.
47. Make sure that the modified part of the RetrieveCustomer_FromRetrieve process looks as shown in Figure 18-14.

Note: If your process diagram looks different, drag the activities to match what is shown in Figure 18-14.

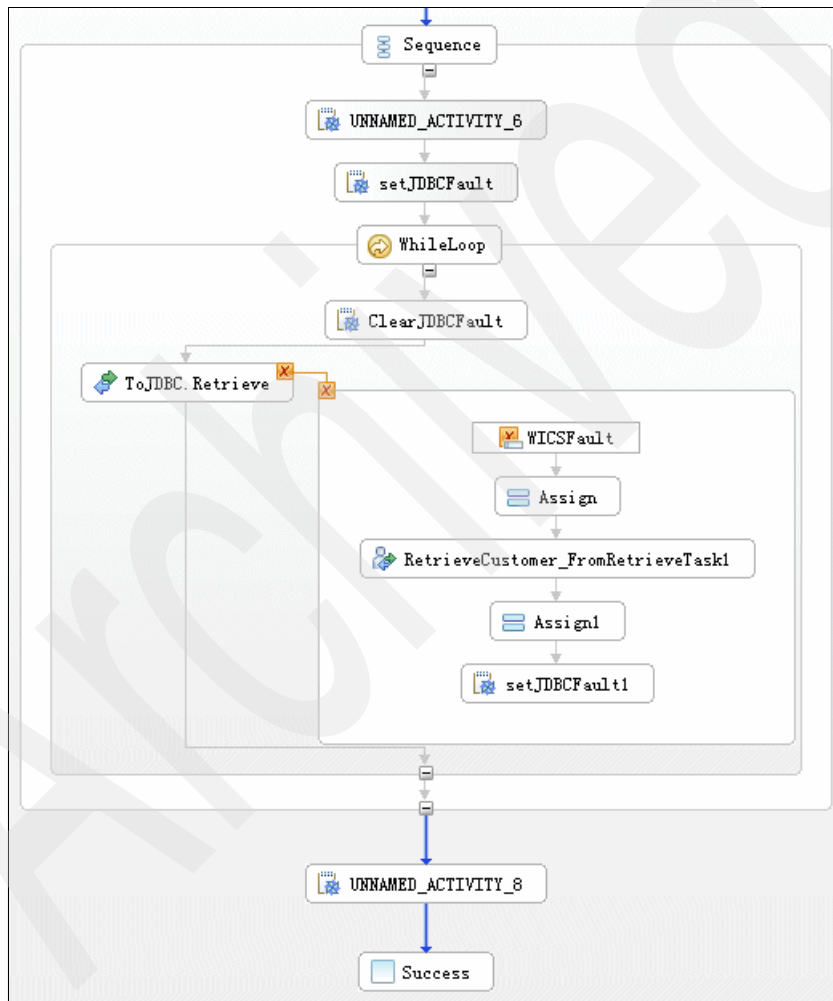


Figure 18-14 Modified part of the RetrieveCustomer_FromRetrieve process

48. In the Business Integration view, expand the **JDBCConnector** module and open the associated assembly diagram. Double-click the **Output_Processing** Java component to open it.
49. Add the code that is shown in Example 18-5 to the Java component. The code is under the line “public class Output_Processing {”

Example 18-5 Adding HashMap and BOFactory to a WBI2WSA Java component

```
private static final HashMap subTypes = new HashMap();
private static BOFactory boFactory = null;

static
{
    Output_Processing.boFactory =
        (BOFactory)ServiceManager.INSTANCE.locateService("com/ibm/websphere/
        bo/BOFactory"); //$NON-NLS-1$
    // -1 is the default "AppUnknown"
    subTypes.put(new Integer(-2), "AppTimeOut");
    subTypes.put(new Integer(-3), "AppLogOnFailure");
    subTypes.put(new Integer(-4), "AppRetrieveByContentFailed");
    subTypes.put(new Integer(-5), "AppMultipleHits");
    //There is no -6
    subTypes.put(new Integer(-7), "AppBusObjDoesNotExist");
    subTypes.put(new Integer(-8), "AppRequestNotYetSent");
}
```

50. Click the **red icon** on the left side of the editor pane. The red line under the Java class **HashMap** indicates that the reference is not resolved. From the list that opens below the class name, double-click **Import 'HashMap'**. Figure 18-15 shows the result.

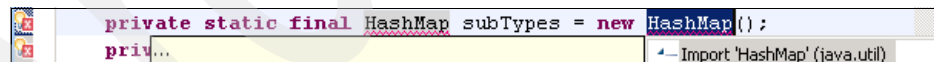


Figure 18-15 Importing the HashMap class

51. Click the **red icon** on the left side of the editor pane. The red line under the Java class **BOFactory** indicates that the reference has not been resolved. From the list that opens below the class name, double-click **Import 'BOFactory'**. Figure 18-16 shows the result.

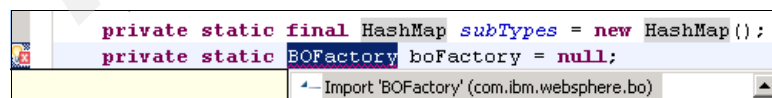


Figure 18-16 Importing the BOFactory class

52. Change the implementation of the `CustomerRetrieveBG_Sync()` method as shown in Example 18-6.

Example 18-6 CustomerRetrieveBG_Sync() method in the Output_Processing Java component

```
public DataObject CustomerRetrieveBG_Sync(DataObject
aCustomerRetrieveBG)
throws Exception {
    String verb = getVerb(aCustomerRetrieveBG);
    if (verb.equalsIgnoreCase("deltaupdate"))
        aCustomerRetrieveBG.setString("verb", "Update");
    String operation = getOperation(verb, aCustomerRetrieveBG);
    DataObject output=null;
    try{
        output = (DataObject) locateService_Output().invoke(
            operation, aCustomerRetrieveBG);

    }catch(Exception e)
    {
        return this.createWICSFault(null, e.getLocalizedMessage());
    }
    if (verb.equalsIgnoreCase("retrievebycontent"))
        output = getOutputType(output);
    return output;
}
```

53. Add the `createWICSFault` method to the Java component as shown in Example 18-7.

Example 18-7 Add createWICSFault() method in the Output_Processing Java component

```
private DataObject createWICSFault(String subType, String
description)
{
    DataObject fault =
    Output_Processing.boFactory.create("http://www.ibm.com/websphere/cro
ssworlds/2002/FaultSchema/WICSFault", "WICSFault");

    if (subType == null)
    {
        subType = "AppUnknown";
    }

    fault.set("type", "ServiceCallException");
    fault.set("subtype", subType);
}
```

```

if (description != null)
{
    fault.set("description", description);
}

throw new ServiceBusinessException(fault);
}

```

54. Click the **red icon** on the left side of the editor pane. The red line under the Java class `ServiceBusinessException` indicates that the reference has not been resolved. From the list that opens below the class name, double-click **Import 'ServiceBusinessException'**. Figure 18-17 shows the result.

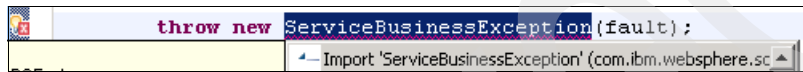


Figure 18-17 Importing the `ServiceBusinessException` class

55. In the WebSphere Integration Developer menu, click **File** → **Save All**.
56. In the WebSphere Integration Developer menu, select **Project** → **Clean** to rebuild the workspace.
57. In the Problems view, verify that there are no build errors.
58. Republish all of the modules to the server by right-clicking the server and selecting **Publish**.

Testing and verification

To test the new error handling capability, you must send a wrong data request, which starts a human task that we perform to get a valid response:

1. Set up the data access scenario testing environment by following the corresponding steps in Chapter 15, “Data access scenario with technology adapters” on page 293.
2. Run `CreateCustomer` by following the corresponding steps in Chapter 15, “Data access scenario with technology adapters” on page 293. Input the data as indicated in Example 18-8.

Example 18-8 Input data of `CreateCustomer`

```

CUSTOMERID: 123456
CUSTOMERNAME: Jim
CUSTOMERADDRESS: 4400 North 1st Street
CUSTOMERCITY: San Jose
CUSTOMERSTATE: CA

```

CUSTOMERZIP: 95134
CUSTOMERTELEPHONE: 4089561234

3. Make sure that CreateCustomer runs successfully.
4. Make sure that no row in the JDBCCustomer table has a CUSTOMERID of 123456. Figure 18-18 shows a record in the JDBCCustomer database table.

CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERZIP	CUSTOMERTELEPHONE
123456	Jim	4400 North 1st Street	San Jose	CA	95134	4089561234

Figure 18-18 JDBCCustomer table data

5. Run RetrieveCustomer by following the steps in 15.3, “Testing the end-to-end solution” on page 335. Then, type 12345 for the CUSTOMERID as shown in Figure 18-19. Click **Go**.

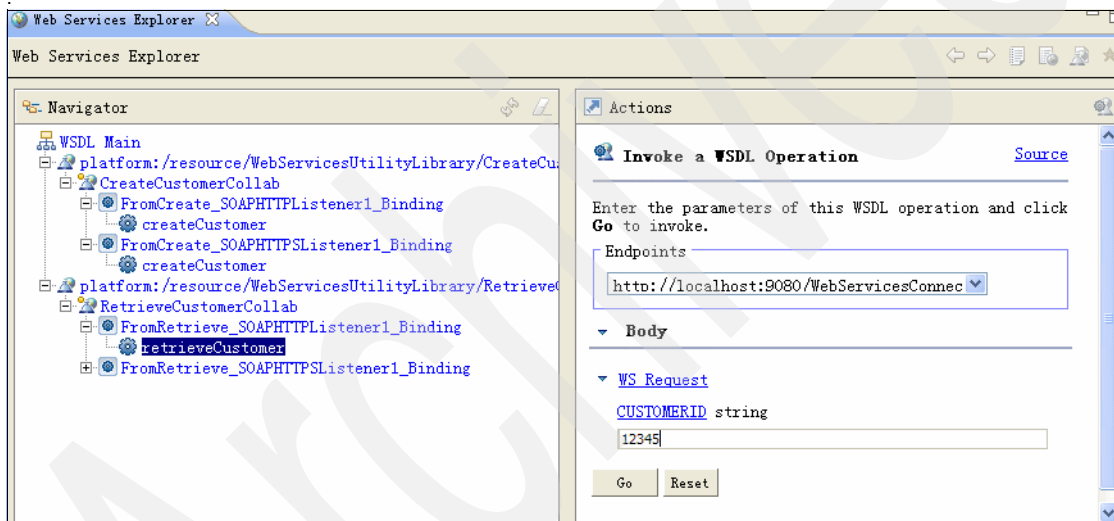


Figure 18-19 RetrieveCustomer test client

6. Open the Business Process Choreographer Explorer. Right-click the server and select **Launch** → **Business Process Choreographer Explorer**. Alternatively, you can also access the Explorer by going to the following URL:

`http://localhost:9080/bpc`

- Under Task Instances in the left pane, click **My To-dos**. In the right pane, you see a HumanTask as shown in Figure 18-20. Select **HumanTask**, and click **Work on**.

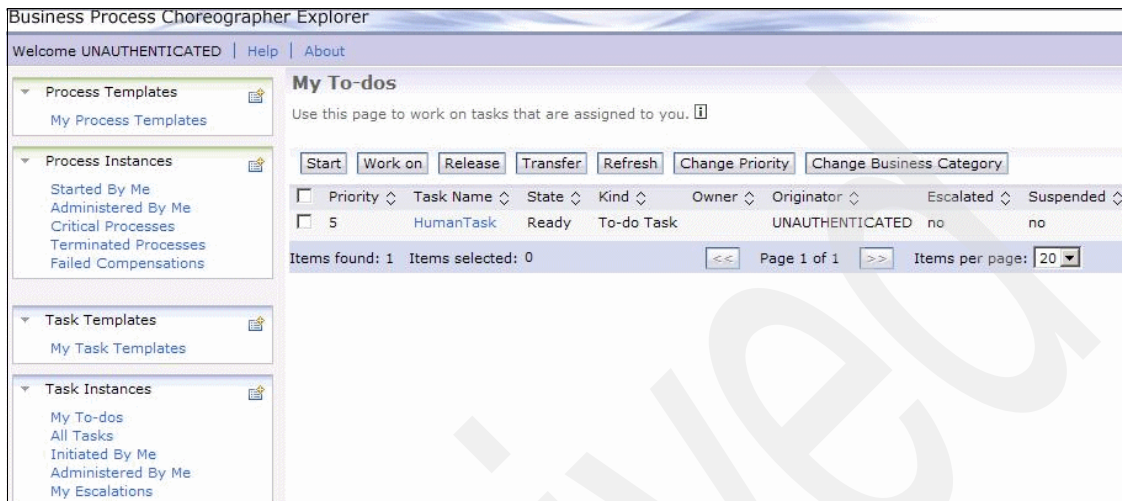


Figure 18-20 Exception handling HumanTask

8. Review the HumanTask input and output messages as shown in Figure 18-21. In the Form View, click **Add**. Type the correct customerID 123456, and click **Edit Source** → **Confirm**. Click **Complete** to complete the HumanTask.

Task Name

HumanTask

Task Input Message

Form View

InputData

CustomerRetrieve

customerID	12345
customerName	
customerAddress	
customerCity	
customerState	
customerZip	
customerTelephone	
@delta	false
@version	*,*,*

ExceptionMessage

BusinessObject: WICSFault@61306130 (type=ServiceCallException, subtype=AppUnknown, description=FaultException thrown in J2CMethodBindingImpl.invoke() com.ibm.j2ca.base.exceptions.RecordNotFoundException: No matching records found., error code:)

View Source

Task Output Message

Form View

OutputData

CustomerRetrieve

Add

Edit Source

Figure 18-21 HumanTask Input and Output message

9. Make sure that no new HumanTask is displayed under My To-dos.

10. Return to the RetrieveCustomer test client. Figure 18-22 shows the result.

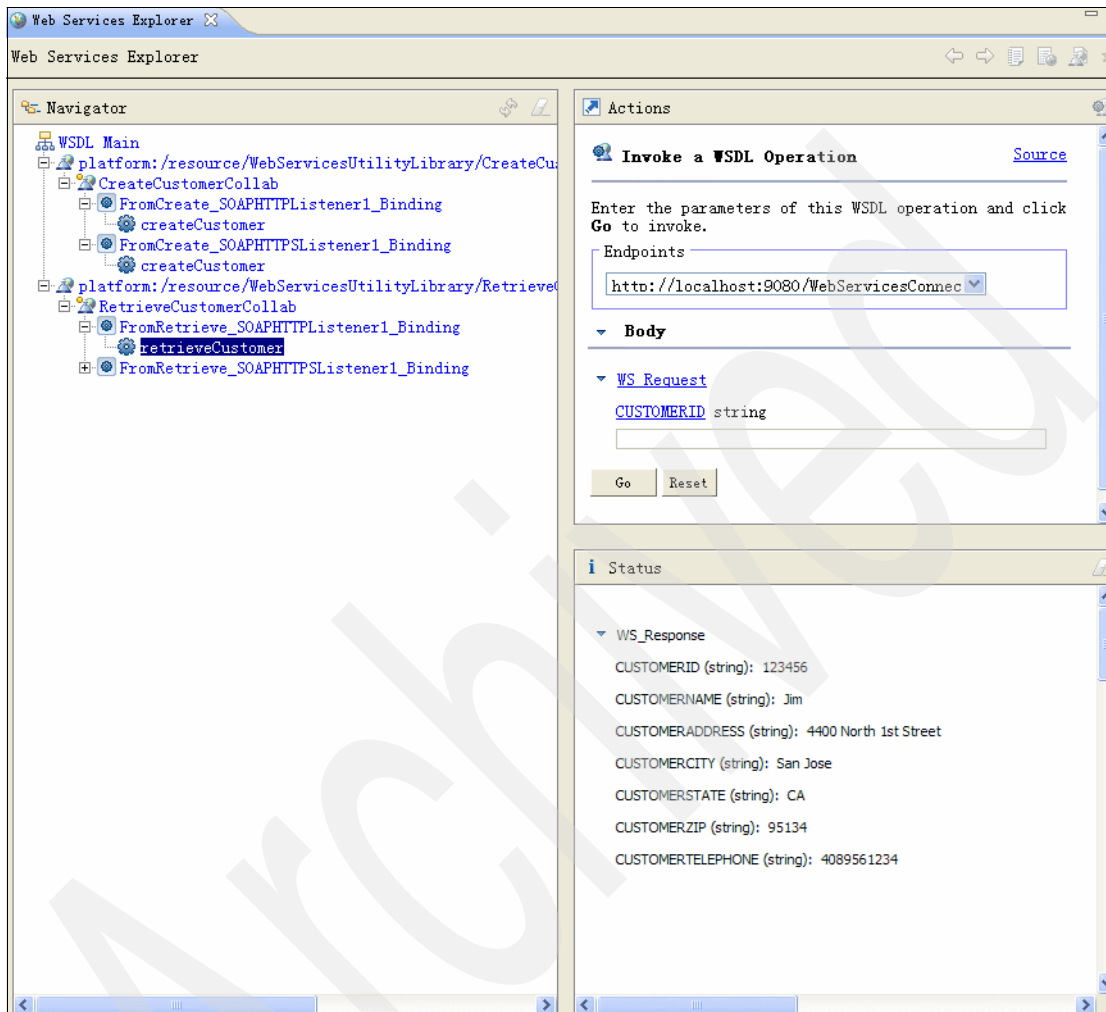


Figure 18-22 RetrieveCustomer result

Conclusion

In this example, we demonstrated a way to enhance error handling after migrating from WebSphere InterChange Server to WebSphere Process Server. For this example, we modified the process by adding a fault handler and a Human Task. We showed how to catch the exception that occurs in a business process and how to delegate control to the business users so that they can fix the problems themselves.

Remember that real applications are obviously more complex than this example. More preparation, design, and customization efforts are typically involved to implement error handling in real applications. For example, we have not shown how to secure the WebSphere Process Server to define business roles, so that only a restricted set of users can access to the relevant Human Tasks. This action typically implies that you have a well defined organization, which is stored in an user registry, such as a Lightweight Directory Access Protocol (LDAP). It is out of the scope of this book to discuss how to integrate user registries with WebSphere Process Server to provide role-based access control for human tasks, but it is certainly an extremely valuable feature of the product.

18.2 Enhanced routing capability

In this section, we explain how to optimize the routing capability after an IBM WebSphere InterChange Server project is migrated to a WebSphere Process Server integration solution.

Because routing is a diverse topic, which can be implemented in many ways in both WebSphere InterChange Server and WebSphere Process Server, we do not cover all of the possible scenarios in this section. Instead, we explain several important concepts, based on a typical example.

We discuss the following topics:

- ▶ 18.2.1, “Routing capability in the WebSphere InterChange Server” on page 535.
- ▶ 18.2.2, “Migrating the routing capability in WebSphere Process Server” on page 539.
- ▶ 18.2.3, “Optimization of the routing capability in WebSphere Process Server” on page 545.

18.2.1 Routing capability in the WebSphere InterChange Server

Figure 18-23 on page 536 shows an example of artifacts designed to provide a routing capability within the WebSphere InterChange Server.

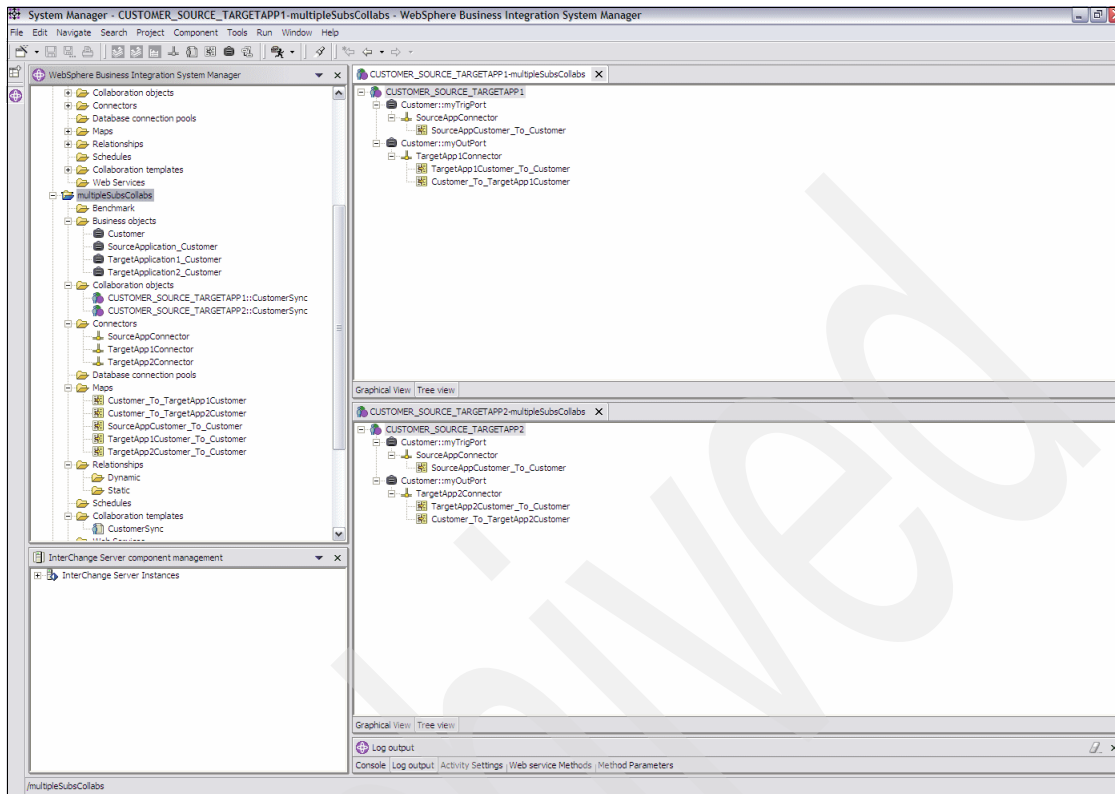


Figure 18-23 Example of a routing logic implementation in WebSphere InterChange Server

In our example, the routing capability relies on two collaboration objects with the following characteristics:

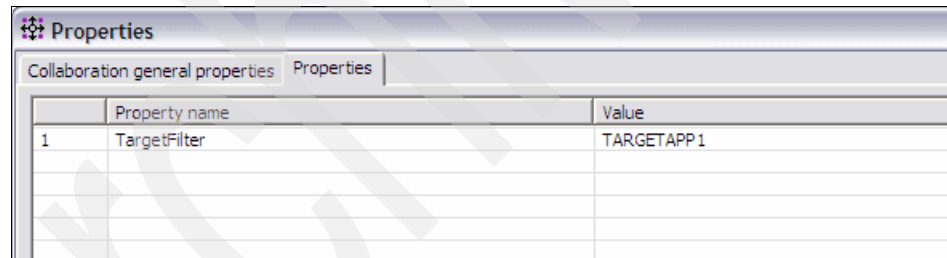
- ▶ The two collaboration objects are CUSTOMER_SOURCE_TARGETAPP1 and CUSTOMER_SOURCE_TARGETAPP2. They instantiate a common collaboration template called CustomerSync.
- ▶ The two collaboration objects subscribe to a common event, which is the SourceApplication_Customer business object, sent by a connector called SourceAppConnector.
- ▶ Each collaboration object has a separate target connector:
 - CUSTOMER_SOURCE_TARGETAPP1 is connected to the outbound connector called TargetApp1Connector. This connector interacts with an application called Application1.
 - CUSTOMER_SOURCE_TARGETAPP2 is connected to the outbound connector called TargetApp2Connector. This connector interacts with an application called Application2.

In summary, we have multiple subscribing collaborations to a common event, which allows sending the event to multiple applications (Application1 and Application2).

While this pattern does not yet provide true routing capability, it provides *multireceiver* or *multicast* capability, where a single event can be sent to multiple receivers. The multicast capability results from the internal publish/subscribe mechanism that is available in the WebSphere InterChange Server. This mechanism allows an inbound adapter (SourceAppConnector in our example) to duplicate the event so that it can be sent to each subscribing collaboration object.

You can achieve true routing capability in the WebSphere InterChange Server by adding filtering logic in the collaboration template by using collaboration properties. In this example, there are two collaboration objects: CUSTOMER_SOURCE_TARGETAPP1 and CUSTOMER_SOURCE_TARGETAPP2. The collaboration template CustomerSync defines a property called TargetFilter. The value of TargetFilter defines whether the collaboration object must filter the event.

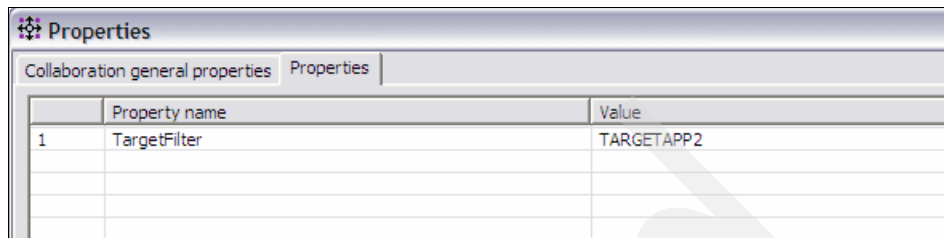
For example, the collaboration object CUSTOMER_SOURCE_TARGETAPP1 has this property set to the value of TARGETAPP1, as shown in Figure 18-24.



	Property name	Value
1	TargetFilter	TARGETAPP1

Figure 18-24 The first collaboration object has the TargetFilter property set to the value of TARGETAPP1

Similarly, the collaboration object CUSTOMER_SOURCE_TARGETAPP2 has this property set to the value of TARGETAPP2, as shown in Figure 18-25.



Properties		
Collaboration general properties		
Properties		
	Property name	Value
1	TargetFilter	TARGETAPP2

Figure 18-25 The second collaboration object has the TargetFilter property set to the value of TARGETAPP2

By setting these property values, the individual collaboration objects can determine if the event must be filtered. For example, if the event contains a value of TARGETAPP2 in one of its fields, CUSTOMER_SOURCE_TARGETAPP1 filters it out, while CUSTOMER_SOURCE_TARGETAPP2 processes it. It is easy to implement filtering logic within the collaboration template to get the collaboration object property value, compare it to the relevant fields in the incoming business object event, and then decide whether to filter the event.

Figure 18-26 on page 539 shows a typical filtering logic implementation in the collaboration template CustomerSync.

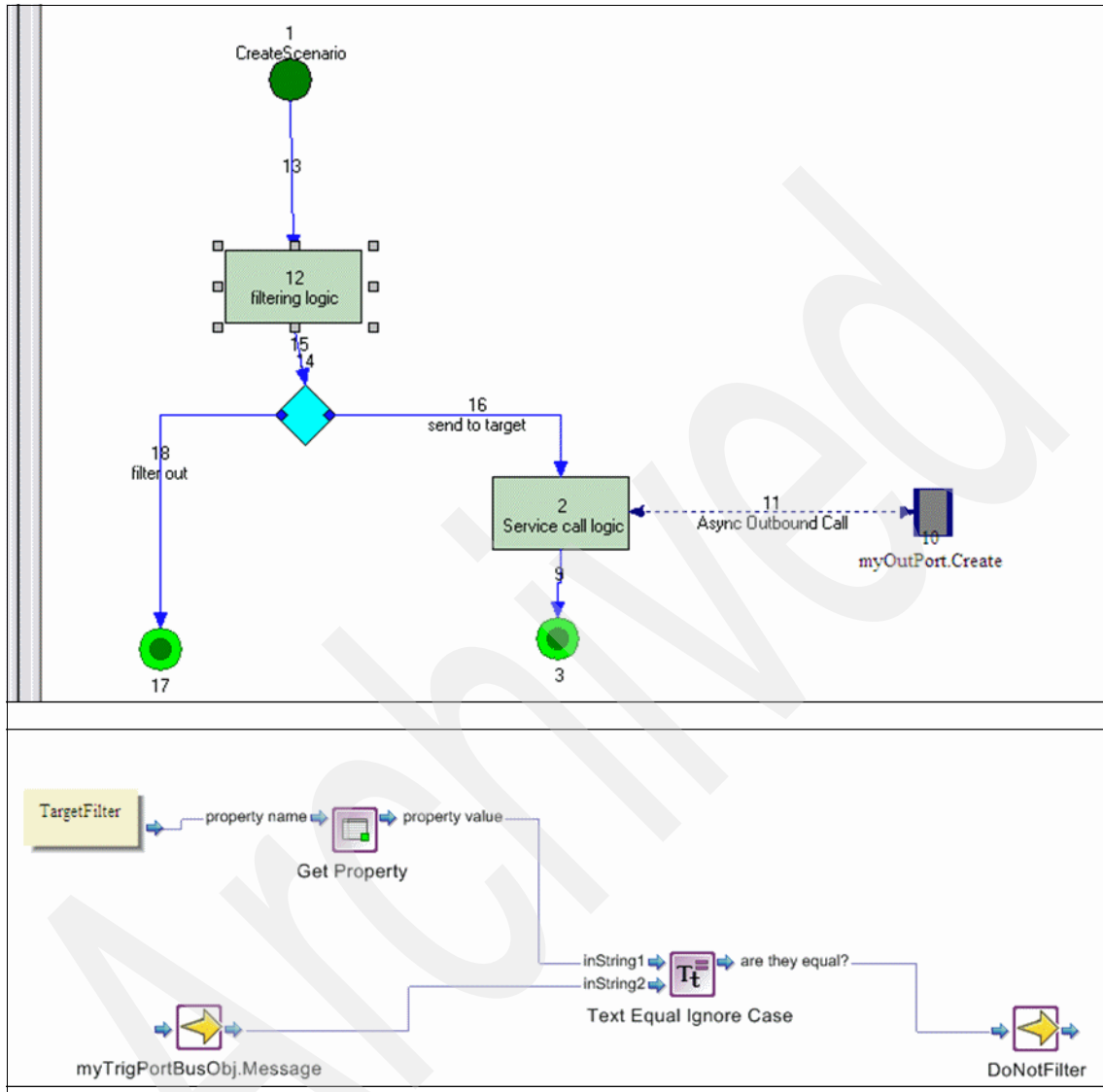


Figure 18-26 Filtering logic implemented in the collaboration template *CustomerSync*

18.2.2 Migrating the routing capability in WebSphere Process Server

In this section, we translate the routing capability example that was described in the previous section into WebSphere Process Server artifacts by using the standard migration tooling.

Figure 18-27 provides a global view of the previous example that was migrated by importing it into WebSphere Integration Developer. The figure shows the Projects (Modules) that are created in WebSphere Integration Developer and the Integration Solution diagram.

Note: The Integration Solution diagram is a new feature that is available in WebSphere Integration Developer 6.2.

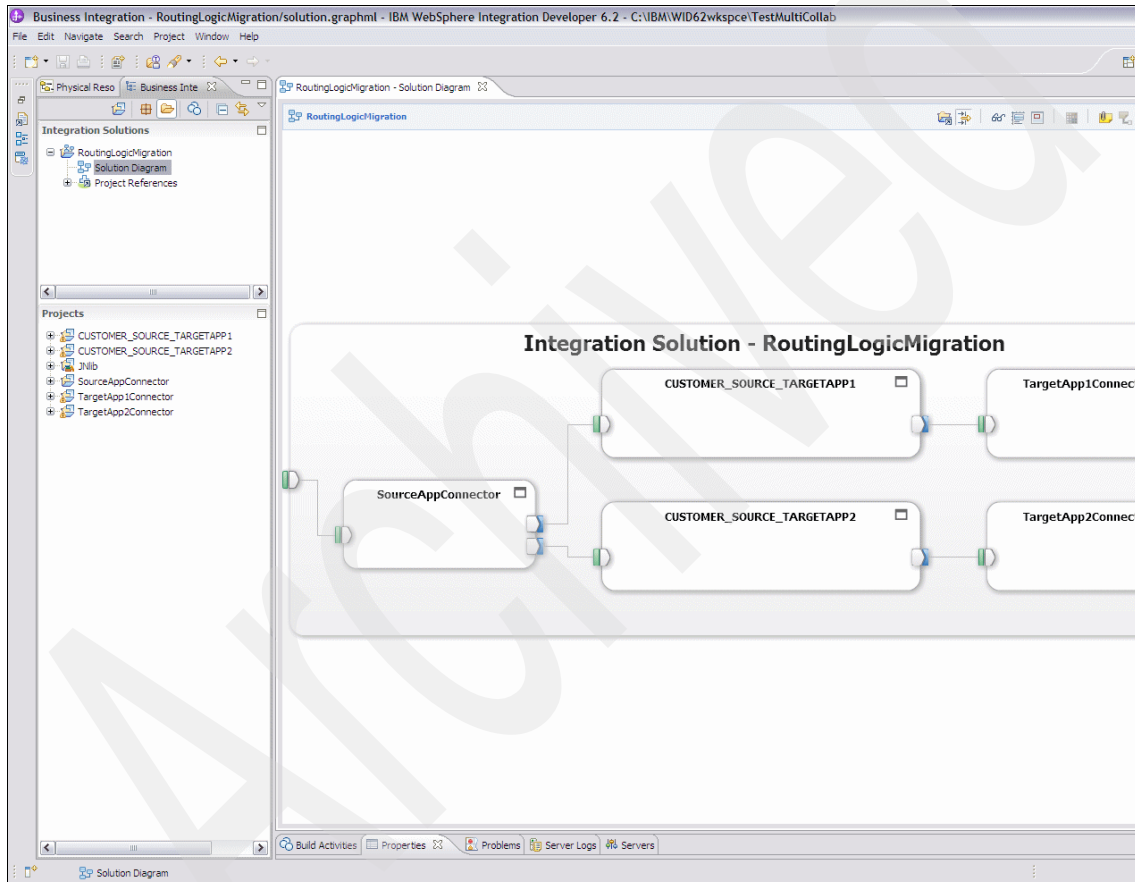


Figure 18-27 Global view of the migration of the WebSphere InterChange Server routing example within WebSphere Integration Developer

Notice these key points:

- ▶ The connectors are migrated into WebSphere Process Server modules:
 - The inbound connector SourceAppConnector is migrated into the module called SourceAppConnector.
 - The outbound connector TargetApp1Connector is migrated into the module called TargetApp1Connector.
 - The outbound connector TargetApp2Connector is migrated into the module called TargetApp2Connector.
- ▶ Each collaboration object is translated into a WebSphere Process Server module that contains a Business Process Execution Language (BPEL) component. It is important to notice that the BPEL component is not shared by the modules and is unique to each module (even if it is the result of a common artifact that was the collaboration template):
 - Collaboration object CUSTOMER_SOURCE_TARGETAPP1 is migrated into a module called CUSTOMER_SOURCE_TARGETAPP1. It contains a BPEL component called *CUSTOMER_SOURCE_TARGETAPP1_myTrigPort*.
 - Collaboration object CUSTOMER_SOURCE_TARGETAPP2 is migrated into a module called CUSTOMER_SOURCE_TARGETAPP2. It contains a BPEL component called *CUSTOMER_SOURCE_TARGETAPP2_myTrigPort*.
 - Figure 18-28 on page 542 shows in more detail how collaboration objects are migrated. Because the two collaborations are migrated in exactly the same way, we only show CUSTOMER_SOURCE_TARGETAPP1. The name of the BPEL component for this module is *CUSTOMER_SOURCE_TARGETAPP1_myTrigPort*.

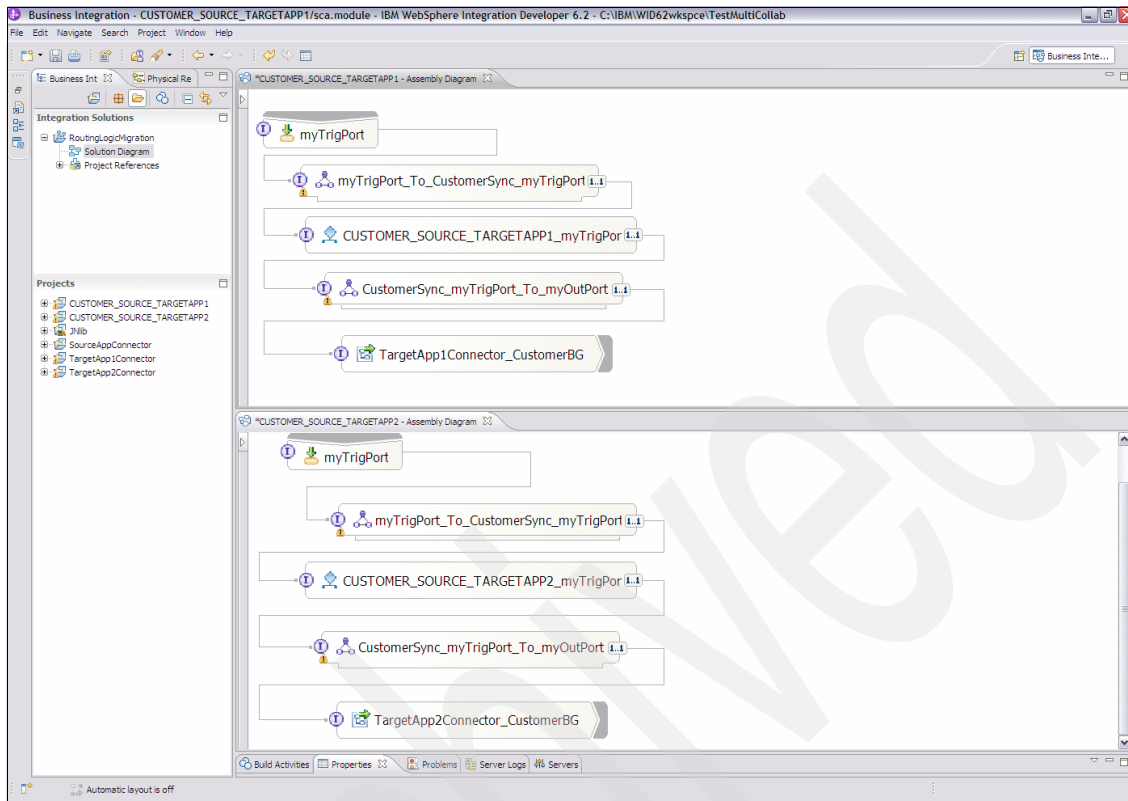


Figure 18-28 Migration of the collaboration objects with WebSphere Integration Developer

Now, we show how the standard migration tool specifically handles the routing logic. We first need to look at the inbound connector module, which is the SourceAppConnector module. Figure 18-29 shows this module.

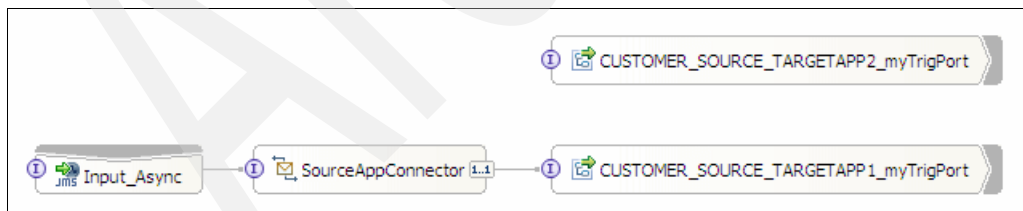


Figure 18-29 Migration of the inbound connector module

Notice these key points:

- ▶ The inbound connector module logic is implemented with a Mediation Flow Component (called `SourceAppConnector`).
- ▶ The module is bound to only one target module through the SCA import called `CUSTOMER_SOURCE_TARGETAPP1_myTrigPort`. The other SCA import, called `CUSTOMER_SOURCE_TARGETAPP2_myTrigPort`, is not bound. Therefore, as is, the event is sent to only one receiver module, that is, the `CUSTOMER_SOURCE_TARGETAPP1` module.

We no longer have the original multicast capability. We must make a change to be able to send the event to multiple receivers. The standard Migration Wizard leaves it up to you to implement this part of the routing capability, which is discussed in 18.2.3, “Optimization of the routing capability in WebSphere Process Server” on page 545.

We look at the BPELs in the collaboration modules now. BPELs are equivalent to the collaboration template and the collaboration objects, regarding the filtering logic:

- ▶ The collaboration property `TargetFilter` is translated into a BPEL custom property that is also called `TargetFilter`. The value of the property corresponds to the value that was set on each collaboration object.
- ▶ The filtering logic that was implemented in the collaboration template is reproduced in the BPEL, which is shown in Figure 18-30 on page 544.

18.2.3 Optimization of the routing capability in WebSphere Process Server

WebSphere Process Server provides more options to implement routing logic than the WebSphere InterChange Server. With WebSphere Process Server, you can use various components and capabilities, such as:

- ▶ Selector components
- ▶ Routing primitives in mediation flow components
- ▶ Dynamic endpoint selection capabilities and APIs
- ▶ Java Message Service (JMS) publish/subscribe capability with JMS binding

In this section, we do not review all of the capabilities of these components. We discuss improvements to the solution that was obtained with the standard migration tool. We focus on two improvements to our solution regarding routing capabilities. The first approach consists of enabling a multicast capability in the inbound adapter module. The second approach consists of merging the process modules and adding a new component that is in charge of the routing, with a focus on maintainability and dynamicity.

Enable a multicast capability at the inbound adapter level

We implement the missing multicast capability that we have outlined in the 18.2.2, “Migrating the routing capability in WebSphere Process Server” on page 539. Implementing this multicast capability is relatively easy:

1. We add a reference to our Mediation Flow Component SourceAppConnector to bind it to the CUSTOMER_SOURCE_TARGETAPP2_myTrigPort SCA import. The Mediation Flow Component is now able to duplicate the event to two receivers, which is shown in Figure 18-31 (to be compared with Figure 18-29 on page 542).

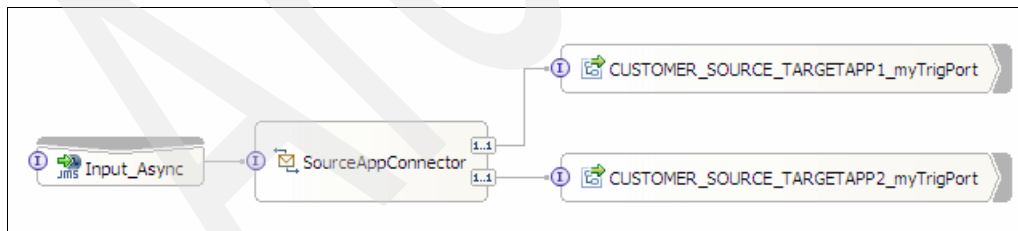


Figure 18-31 Custom multicast in the inbound adapter module: New reference

2. We modify the Mediation Flow Component implementation so that it takes care of the actual event duplication mechanism. Figure 18-32 on page 546 shows the original Mediation Flow Component (MFC) implementation. Figure 18-33 on page 547 shows the modification that we have applied to the

MFC, to duplicate the event and to send it to the two targets. We used a custom mediation primitive for this purpose, but other implementations are possible.

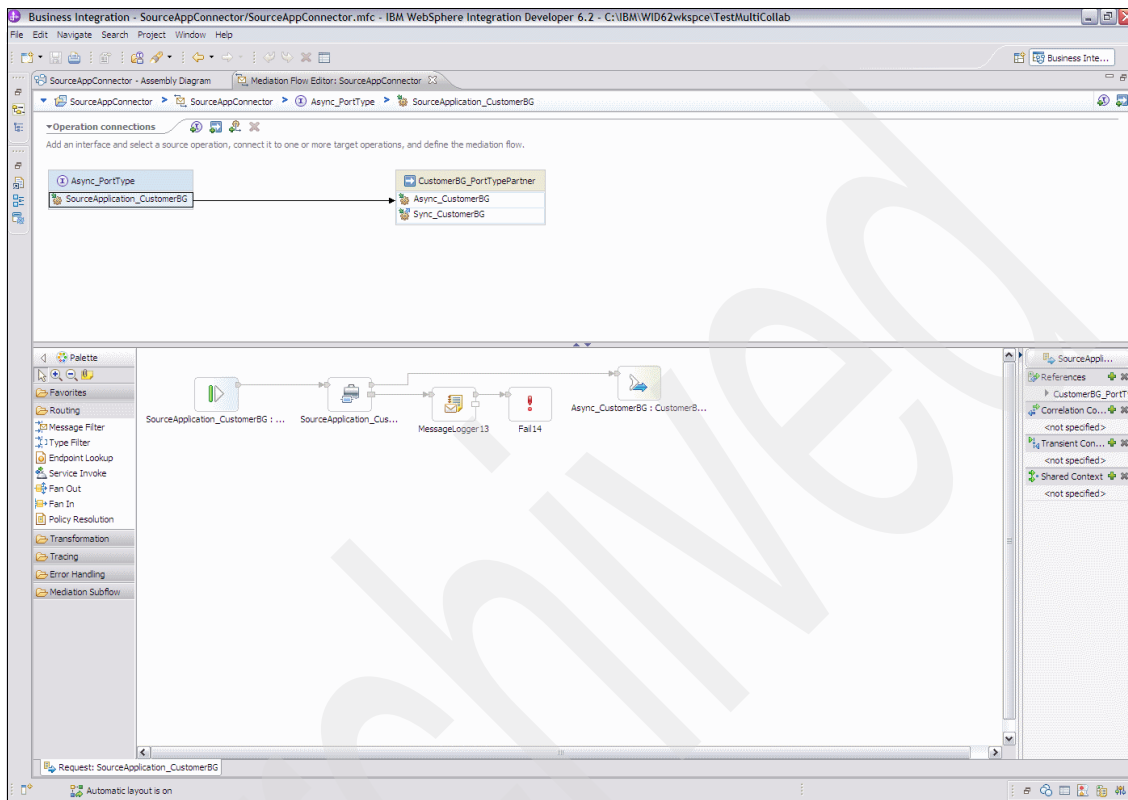


Figure 18-32 Custom multicast in the inbound adapter module: Original implementation in the MFC

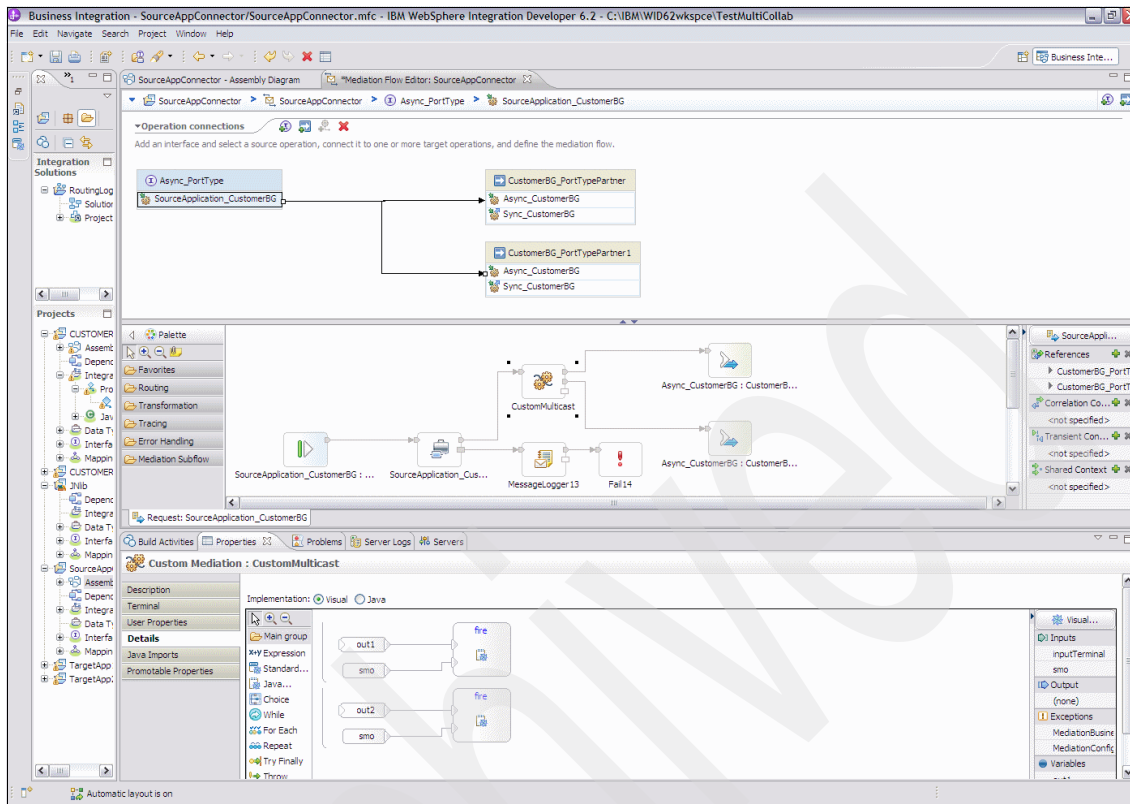


Figure 18-33 Custom multicast in the inbound adapter module: New implementation in the MFC

In summary, we were able to implement the multicast capability quite easily in the inbound adapter module. We now have an equivalent solution to the original WebSphere InterChange Server, because it is based on multicast and filtering logic. Compared to the WebSphere InterChange Server implementation, however, this solution lacks maintainability.

Optimizing the routing capability for maintainability and dynamicity

The Migration Wizard tool can migrate the filtering logic coming from the collaborations. But the way that it was done might be a problem. The integration logic (including the filtering logic) that was coded in the CustomerSync collaboration template was duplicated in two WebSphere Process Server modules (CUSTOMER_SOURCE_TARGETAPP1 and CUSTOMER_SOURCE_TARGETAPP2). This duplication occurs, because the concept of a template/object does not exist anymore with BPEL. So, the

migration tooling had to generate two similar BPELs, one in each module, which leads to a potential maintainability issue. Then, if we need to modify our integration logic (for example, to add more quality of service (QoS) around the service call logic snippet visible that is in Figure 18-30 on page 544), we must modify our integration logic in both BPELs, which is a duplication of effort.

Actually, the way that we have implemented the multicast capability in the inbound connector module also raises a maintainability issue. Assume that we must consider a new target application for the same event as our two existing applications. With our current implementation, we update our integration solution this way:

- ▶ Create a new BPEL module (probably by duplicating an existing one, such as CUSTOMER_SOURCE_TARGETAPP1).
- ▶ Impact the inbound connector module so that the multicast is done for three targets instead of two targets.
- ▶ Add a new module for the newly required outbound connector.

The two first items look like overhead. The first item worsens our maintainability issue with more duplication, and the second item shows another issue, which is lack of dynamicity. If we want to integrate a new application, we have to impact an existing module.

So, we have an overall design issue. Follow these steps to improve this situation:

1. Get rid of the duplication. We keep only one module between CUSTOMER_SOURCE_TARGETAPP1 and CUSTOMER_SOURCE_TARGETAPP2. Assume that we keep the CUSTOMER_SOURCE_TARGETAPP1 module.
2. If we have only a single BPEL module, we need to remove our filtering logic, which does not work anymore, from the BPEL module.
3. The other side effect of having a single BPEL module is that the multicast capability that we had implemented in the inbound connector module does not make sense anymore. So, for the inbound connector module, we revert back to the original migrated module.
4. We introduce a new dedicated component that we put after the BPEL component in the CUSTOMER_SOURCE_TARGETAPP1 module to implement the routing logic. Because we want to provide a dynamic solution, we do not explicitly bind this component to the two outbound connector modules. We need to find a routing component that allows true decoupling with the target modules. We discuss that component next.

Figure 18-34 shows an overview of our newly optimized solution. We have only one remaining BPEL module. It is not explicitly bound to the outbound connector modules any longer (to be compared with Figure 18-27 on page 540).

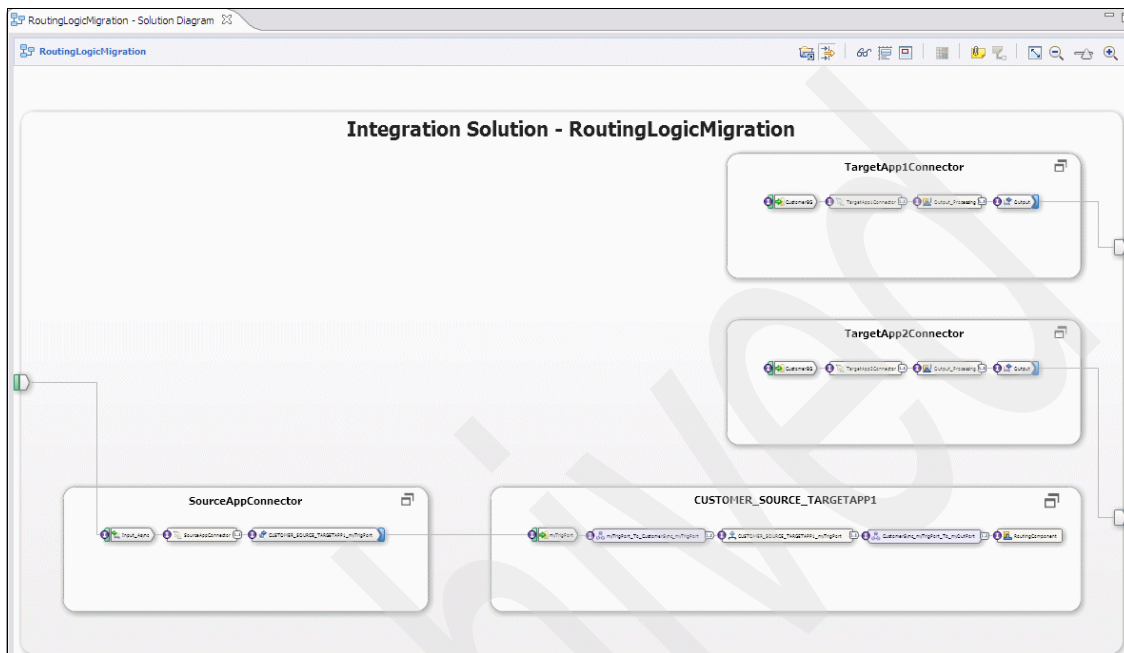


Figure 18-34 Optimized migrated solution for maintainability and dynamicity

In this paragraph, we discuss the type of component to provide routing capability but also *dynamicity* (no static binding with the target modules). We place this new component at the end of the chain of components composing the CUSTOMER_SOURCE_TARGETAPP1 module. We make sure that we do not use any SCA import in our module (static binding). In terms of component implementation, we have two options:

- ▶ We create a custom component that calls the target modules one at a time. The component must be designed to get the list of the target modules dynamically (no hard coding):
 - The component can be implemented using a Mediation Flow, a custom selector, or a custom plain old Java object (POJO).
 - The list of targets can be achieved by accessing an external registry, such as WebSphere Service Registry and Repository. In which case, the Mediation Flow is the preferred implementation, because it provides a standard mediation primitive to call WebSphere Service Registry and Repository. However, we can also use a database or any other means that is suitable for the project.

- With this type of an implementation, there is no impact on the outbound connector modules. The new component is able to call them in a loop using dynamic SCA APIs and capabilities.
- However, each target is called one at a time, so we need to make sure that we handle errors properly (for example, we need to decide if we continue sending the event to other targets in case one submission fails).
- ▶ We can use a JMS binding and rely on the JMS publish/subscribe capability:
 - With this type of a component, we do not need to implement the retrieval of the list of targets, and we do not need to implement the loop that sends the event to each target. It is natively provided through the publish/subscribe capability with additional setup work to define the JMS topics and resources.
 - However, we have an impact on the existing outbound connector modules, because we must add a JMS binding export in them, which is a minor impact.

We do not develop the implementation possibilities for the routing logic any further. Further development requires a detailed study appropriate for your context. Be aware that enhancements always present advantages but also drawbacks:

- ▶ Using dynamic invocation SCA APIs (EndpointReference) is probably the most flexible and powerful, because you can override the SCA endpoint easily and even pass dynamic information, such as queue names, with a JMS implementation. However, it requires Java skills and is less obvious to understand than a publish/subscribe implementation or a static implementation.
- ▶ Using JMS publish/subscribe is simpler in terms of implementation, but it is more sensitive to the deployment topology. JMS topics and subscribers must be carefully handled within a cluster topology to avoid message and task duplication.

You can obtain more details in the WebSphere Process Server documentation for each capability. You can obtain information about the dynamic invocation API at this address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/ref/rwesb_dynamicoverrideepr.html

Conclusion

We first enhanced the migrated solution to provide a routing capability based on the multicast and filtering pattern, which is similar to the original WebSphere InterChange Server solution. We have seen that this solution in WebSphere Process Server can create maintainability and dynamicity issues. We then

investigated other options to provide a solution with better maintainability and a more dynamic and decoupled routing capability.

The second solution might look more appealing than the first solution. It is better in terms of maintainability and dynamicity. However, depending on the context, it might not perform as well, because the dynamicity and the decoupling can involve overhead due to late binding. Your solution will have to be a compromise between flexibility and performance.

18.3 Enhanced migration for artifacts designed visually

In this section, we explain the capability of standard migration tools to migrate WebSphere InterChange Server artifacts that were designed with the Activity Editor. These artifacts can be maps or collaboration templates. Our purpose is to investigate the current limitations of the standard migration tools and to provide the best practices to optimize the migrated code. We use an example based on a simple map that is designed using the WebSphere InterChange Server toolset's Activity Editor. Refer to Chapter 11, "Artifacts migration" on page 215 to get general information about the migration of map artifacts.

This section is structured this way:

- ▶ 18.3.1, "Map designed with the Activity Editor" on page 551
- ▶ 18.3.2, "Migration of the map" on page 553
- ▶ 18.3.3, "Enhanced design for the migrated map" on page 557

18.3.1 Map designed with the Activity Editor

Figure 18-35 on page 552 shows the TestActEdBasic map in the WebSphere InterChange Server Map Designer. This map uses a custom transformation to treat the Message field, which is defined in both source (Clarify_Customer) and target (Customer) business objects.

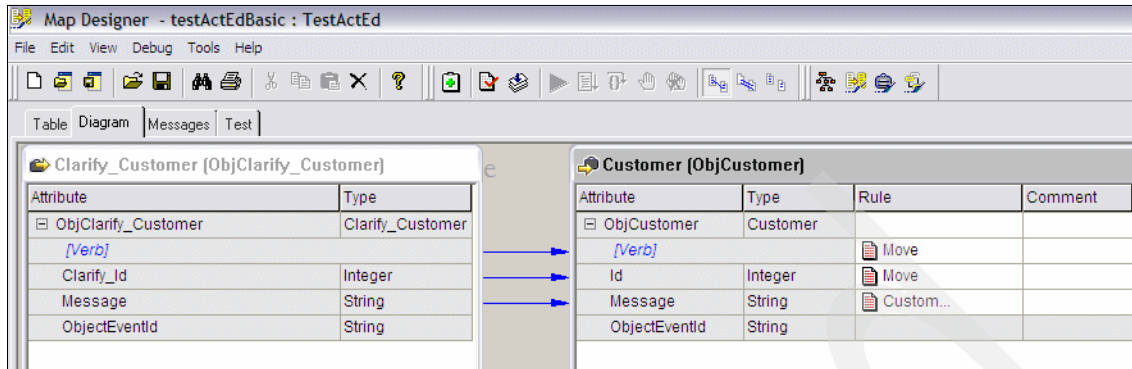


Figure 18-35 The TestActEdBasic map

The custom transformation is implemented with visual code by using the Activity Editor, as shown in Figure 18-36. It is a simple transformation that appends the hello String to the Message field.

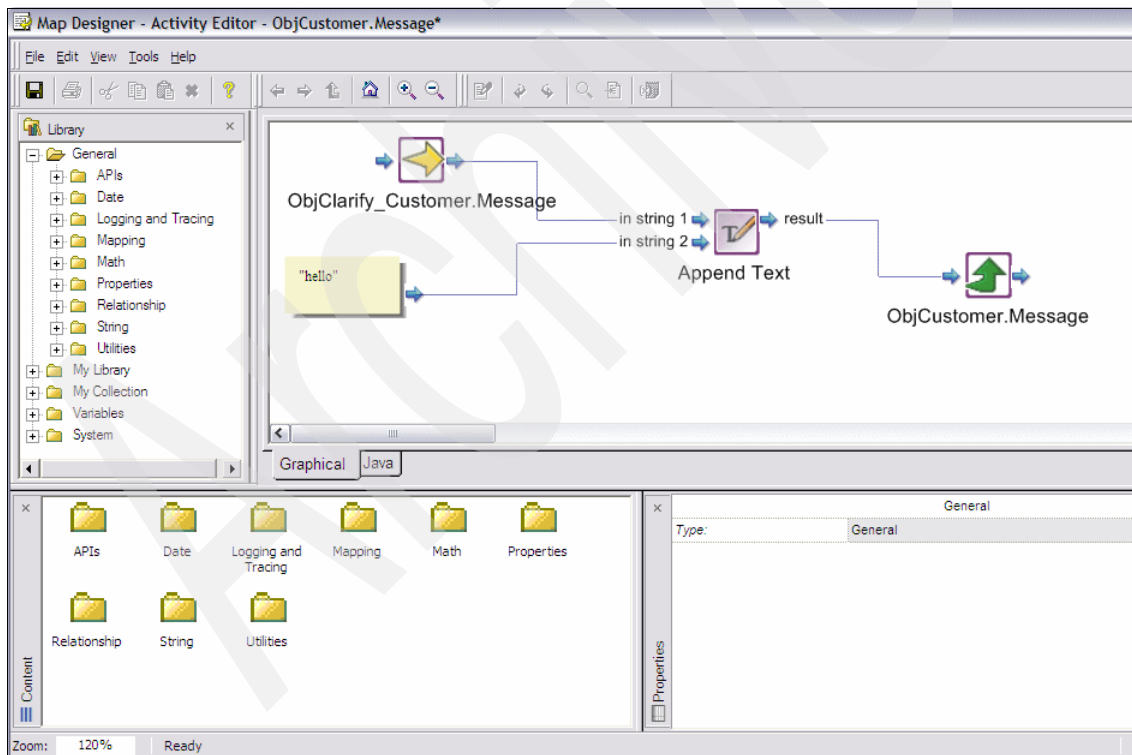


Figure 18-36 Custom transformation in a map that is implemented with Activity Editor

Figure 18-37 shows a part of the generated Java code that corresponds to the Activity Editor design. The Activity Editor has generated java code that relies on the StringBuffer class to append the hello string.

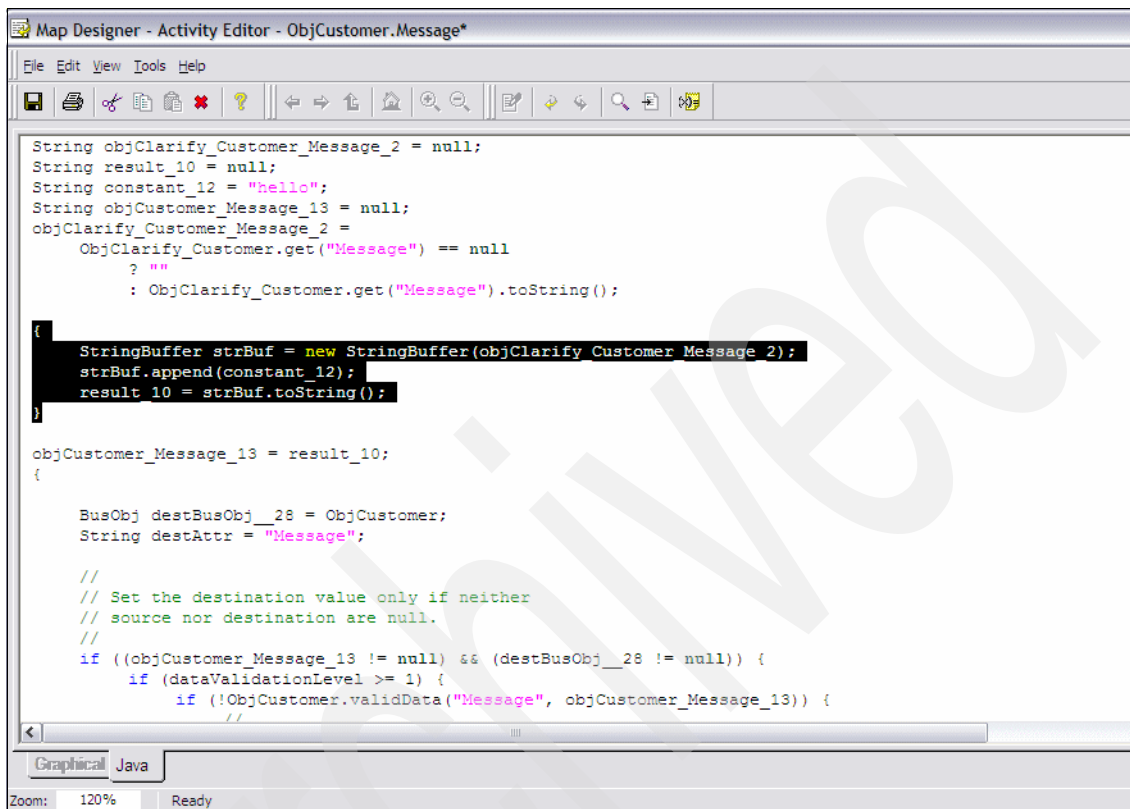


Figure 18-37 Generated Java code corresponding to the custom transformation designed with the Activity Editor

18.3.2 Migration of the map

The standard migration tools migrate this map and its business objects in the usual manner:

- ▶ Each business object becomes a set of two Service Data Objects (SDOs):
 - A business graph to contain the verb information
 - A business object to represent the actual business object payload
- ▶ Two maps are generated to handle the business graph and the business object.

Figure 18-38 shows the map that is related to the business graph.

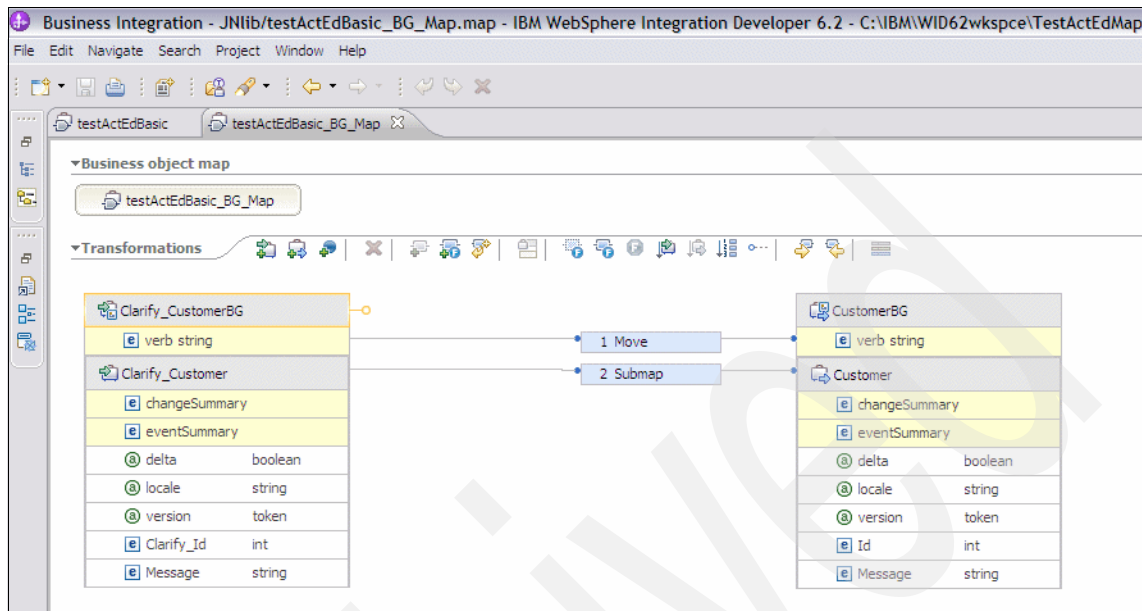


Figure 18-38 Business graph map that is generated by the migration tools

Figure 18-39 shows the business object map, which is the map of interest to us. The WebSphere InterChange Server custom transformation that was depicted in Figure 18-36 on page 552 and Figure 18-37 on page 553 is migrated into the WebSphere Process Server custom transformation called *2 custom*.

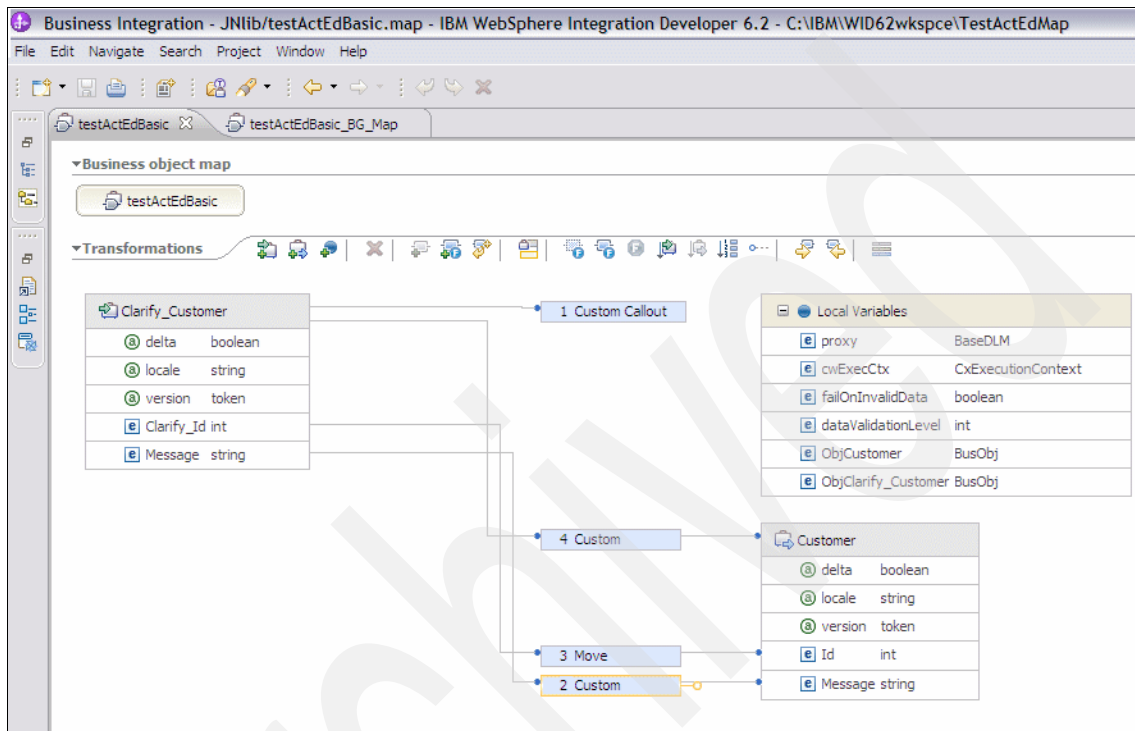


Figure 18-39 Business object map that is generated by the migration tools

We now look more specifically at this *2 custom* transform in the migrated map. Figure 18-40 on page 556 shows the details of the transform.

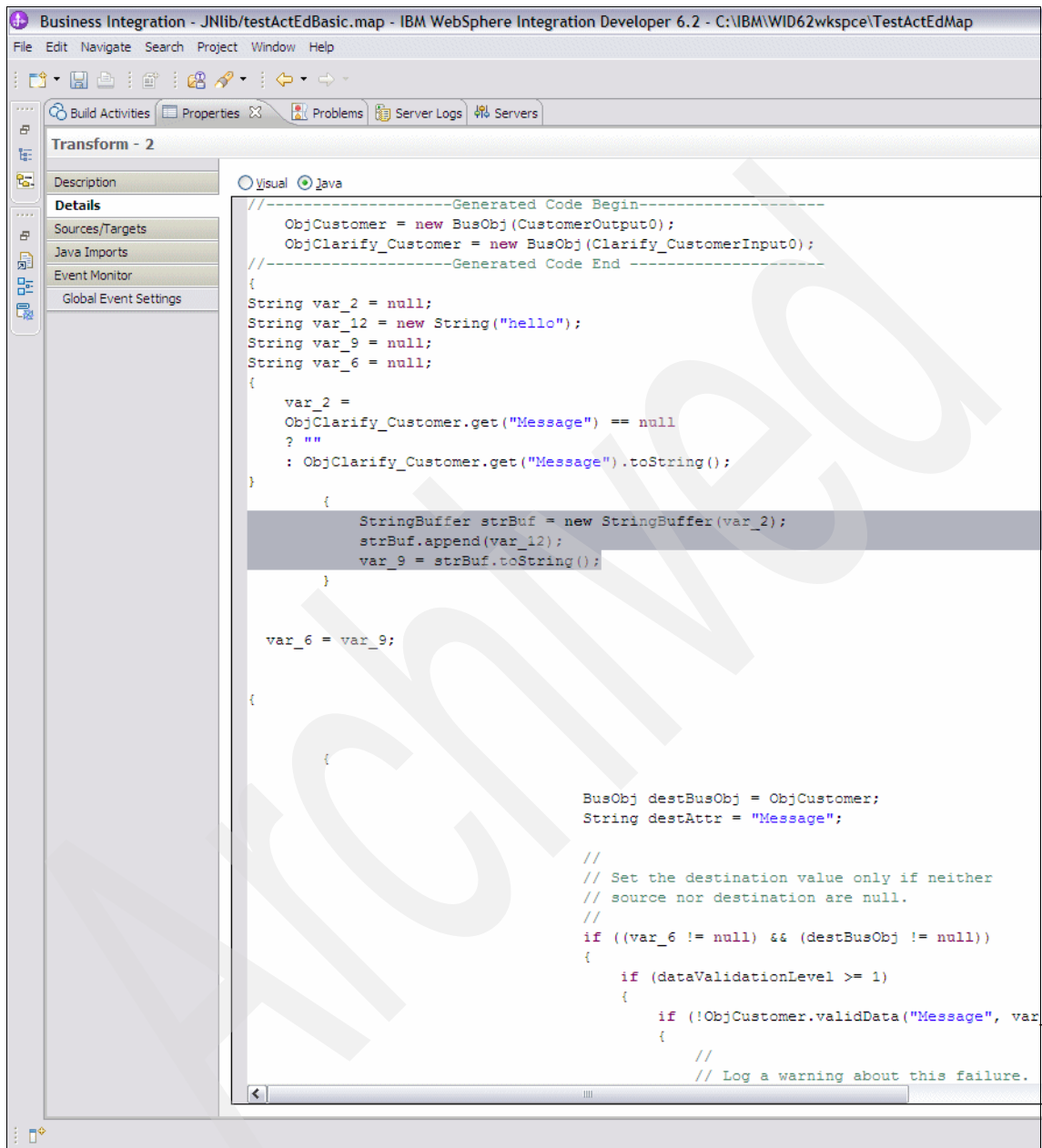


Figure 18-40 Java code is migrated but the visual design of the Activity Editor is lost

18.3.3 Enhanced design for the migrated map

The migration tool was *unable* to convert the original WebSphere InterChange Server visual design into a WebSphere Process Server visual design. As we can see in Figure 18-40 on page 556, the new transform is defined with Java code, but there is no visual code available anymore.

The standard migration tools support the migration of artifacts that are designed with the Activity Editor by migrating the resulting Java code (copy/paste of the original code). However, the standard migration tools are not able to convert the visual design. There is no one-to-one (1:1) mapping between the visual components available in WebSphere InterChange Server and the visual components available in WebSphere Integration Developer.

The inability to convert the visual design causes an issue in terms of maintainability:

- ▶ The visual code was lost, and only Java code is now available. Java code is typically harder to maintain.
- ▶ The migrated Java code relies on WebSphere InterChange Server APIs (Heritage APIs) through which the migration tools migrate specific Java code. Relying on WebSphere Process Server APIs is preferable.

In order to get the best solution, we recommend these steps:

- ▶ Consider the migrated artifacts that are obtained from the standard tools as a short-term solution.
- ▶ Spend the time to redesign parts of the migrated code with the WebSphere Integration Developer Visual Editor so that the code performs better and is easier to maintain.

In the case of our example, we need to redesign the 2 *custom* transform with the WebSphere Integration Developer Visual Editor. Figure 18-41 on page 558 (to be compared with Figure 18-39 on page 555 and Figure 18-40 on page 556) is an example.

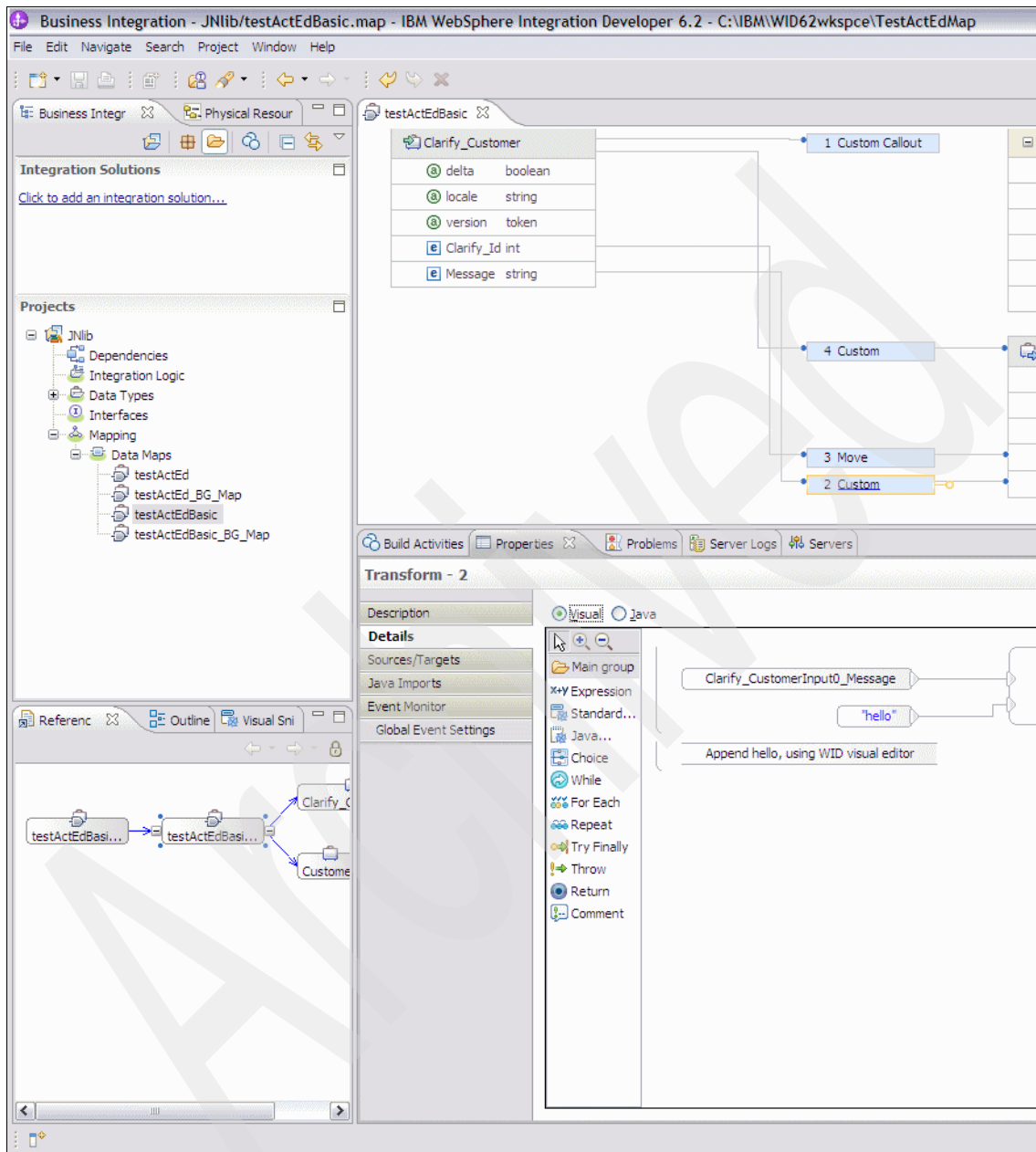


Figure 18-41 Transform has been rewritten using WebSphere Integration Developer Visual Editor

Technical solutions: Troubleshooting

In this chapter, we explain how to use troubleshooting to resolve potential issues that are related to the WebSphere InterChange Server to WebSphere Process Server migration.

Related scenarios: This chapter relates to issues that we experienced in the data access and data synchronization scenarios that were illustrated in Chapter 15, “Data access scenario with technology adapters” on page 293, Chapter 16, “Data synchronization scenario with technology adapters” on page 343, and Chapter 17, “Data synchronization scenario with application adapters” on page 431. We used WebSphere Integration Developer V6.2 and WebSphere Process Server V6.2.

We include the following sections in this chapter:

- ▶ 19.1, “General troubleshooting methods” on page 560
- ▶ 19.2, “Advanced troubleshooting methods” on page 568

19.1 General troubleshooting methods

In this section, we introduce the general troubleshooting methodology of the migration path.

19.1.1 Application module errors

After migrating by using the WebSphere Integration Developer Migration Wizard, the corresponding application module can throw errors in the build. In this type of a scenario, rebuild all of the relevant projects. If this action does not resolve the issue, delete the application module by switching to the Java 2 Platform, Enterprise Edition (J2EE) perspective and rebuilding the projects again. The error will go away.

19.1.2 Deploying and running the scenario in WebSphere Process Server

If you encounter problems while deploying or running the scenario on WebSphere Process Server, use the logs of WebSphere Process Server to help diagnose the problem. The logs are in the *WPS_install/profiles/profile_name/logs/server_name* directory, which contains the following files:

ServerName.pid	The Process ID (pid) of the Java process of WebSphere Process Server
SystemOut.log	The standard Java Virtual Machine (JVM) output log
SystemErr.log	The standard JVM error log
startServer.log	The log of start server events
stopServer.log	The log of stop server events
native_stdout.log	The Stdout of a Java process
native_stderr.log	The Stderr of a Java process
trace.log	The output from a diagnostic trace

The *SystemOut.log* and *SystemErr.log* files are useful for diagnosing the problem of deploying and executing the process.

19.1.3 WebSphere Process Server log and trace

WebSphere Process Server uses the same logging infrastructure as WebSphere Application Server. In this section, we explain how to view and modify the current log and trace levels. We also explain the meaning of the format of the message.

Viewing or changing the current logging and tracing levels

To view or change the current logging and tracing levels:

1. From the administrative console (Figure 19-1):
 - a. In the left pane, expand **Troubleshooting** and click **Logs and Trace**.
 - b. In the right pane, click the WebSphere Process Server, **server1**, to enable logging and tracing.

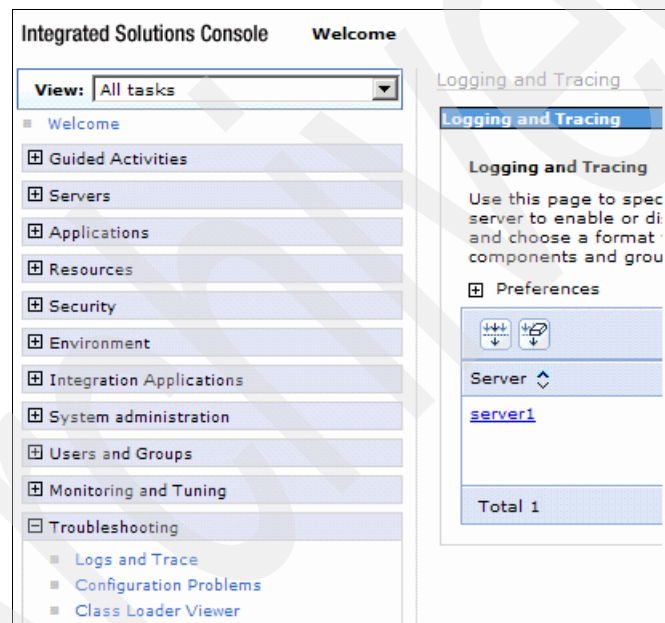


Figure 19-1 Logging and Tracing pane of the administrative console

2. Click **Change Log Detail Levels**.

- Choose and change the log details level as shown in Figure 19-2. The Map service log and trace level (in addition to other message and trace levels) is set to *all* (*=info:com.ibm.wbiserver.map.*=all).

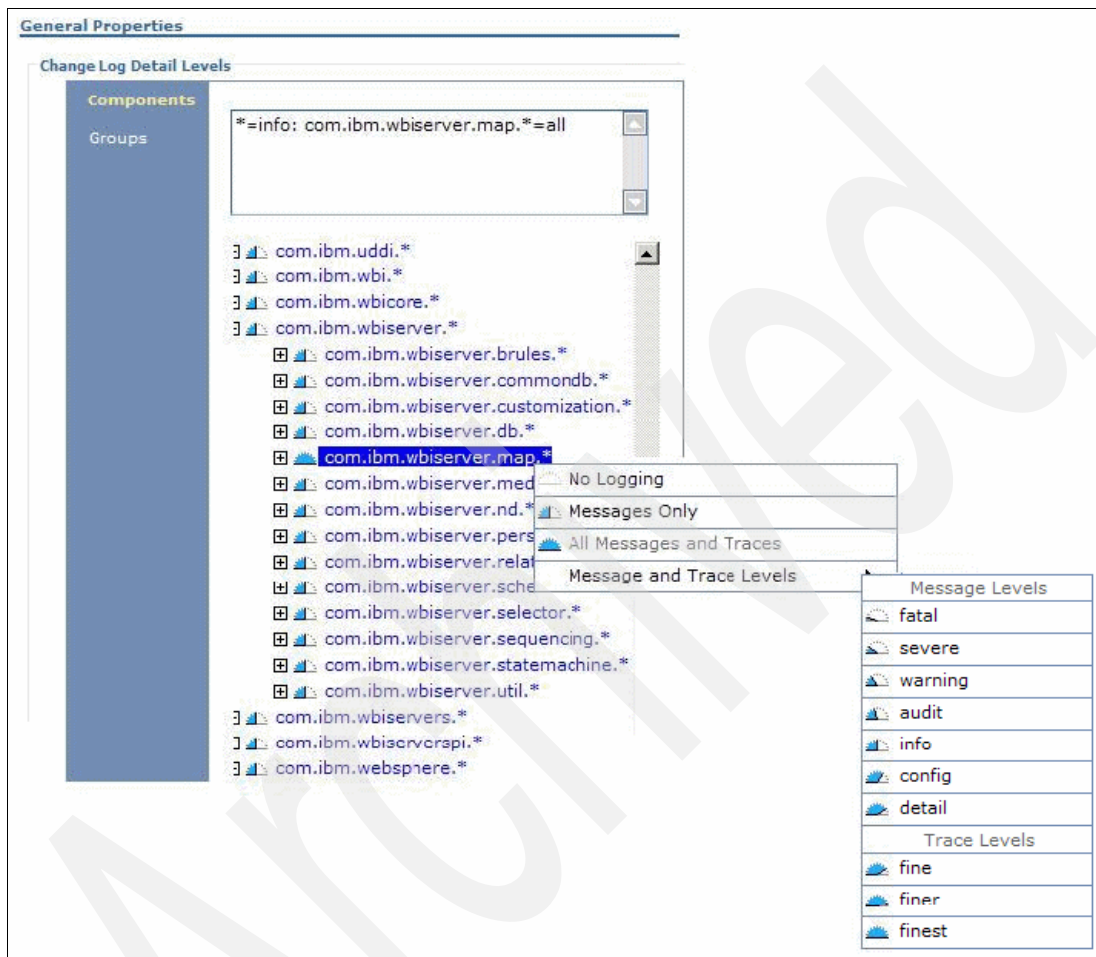


Figure 19-2 Changing the log detail levels

Any components that are not yet loaded are not initially displayed in the administrative console. For example, if the interface map is not invoked, it is not in the Log Level detail list, because interface maps are not loaded when the server is started. You can add any unlisted component manually in the administrative console by specifying the package or component name.

You can supply the level of detail either by component or group. A *group* is a set of related components for a functional area (for example, Web services).

Table 19-1 shows the recommended trace level components and their packages.

Table 19-1 Components and packages

Runtime component	Trace level	Package information
Adapter run time (EIS import/export)	Fine	j2c/jms tracing: com.ibm.ws.sca.*.eis com.ibm.ws.sca.internal.*.eis com.ibm.websphere.sca.*.j2c com.ibm.ws.sca.internal.*.j2c com.ibm.wsspi.sca.j2c com.ibm.websphere.sca.*.jms com.ibm.ws.sca.internal.*.jms com.ibm.wsspi.sca.jms
Artifact Loader	All	Dumps large amounts of output (can choose a lower setting if the size of the log is a concern)
ApplicationScheduler	All	com.ibm.wbiserver.sceduler.* (see also <i>profileName</i> /logs: appScheduler.log, configAppScheduler.ant.log, and configAppSchedulerDBTables.ant.log; all three logs are created during profile creation)
Business objects (BO) and CoreFramework	Finer or all	BOCore
WebSphere Business Process Choreographer, (BPC) is composed of Business Flow Manager and Human Task Manager	All	Business Flow Manager: com.ibm.bpe.*=all Human Task Manager: *=info:com.ibm.bpe.*=all:com.ibm.ws.staffsupport.*=all
Business Process Choreographer install/config	All	com.ibm.bpe.*=all:com.ibm.ws.staffsupport.*=all (staff support is part of the Human Task Manager)
Business rules and selectors	Finest or all	com.ibm.wbiserver.brules.* com.ibm.wbiserver.common.* com.ibm.wbiserver.selector.* com.ibm.wbiservers.brules.* com.ibm.wbiservers.common.* com.ibm.wbiservers.customization.* com.ibm.wbiservers.selector.*
Business state machines	Depends	com.ibm.wbiserver.statemachine.* Fine (high level events - message received, state entry exit), FINER to see source level events, FINEST/ALL for even more), add com.ibm.bpe.*=ALL if needed
Common Event Infrastructure (CEI)	All	com.ibm.wsspi.monitoring.*

Runtime component	Trace level	Package information
Data maps	All	com.ibm.wbiserver.map.*
Failed Event Manager	Fine	recoveryEJB, recoveryMgr (for CEI) WBIMonitor.cei.recovery, WBILocationMonitor.log.recovery
Interface maps	N/A	com.ibm.wbiserver.mediation.*
Human Task Manager	N/A	com.ibm.htm.*
Relationships	N/A	com.ibm.wbiserver.relationshipshipservice.* (finer provides information regarding processing)
Service Component Architecture (SCA)	N/A	Finest (most verbose, all entry/exit) and fine (high level information pertaining to component to component interactions and transaction context)
SCA security	See SCA	com.ibm.ws.sca.internal.security.handler*=all (security id, permission); com.ibm.ws.sca.internal.securitycontext.handler.*=all (security context for async), com.ibm.ws.sca.internal.managed.security.*=all; if more trace is needed, base WebSphere security: SASRas=all;com.ibm.ws.security.*=all
SCA unit of work	See SCA	com.ibm.ws.sca.internal.uow.handler.*
Session	Finest	com.ibm.ws.session.*

Log and trace configuration options

You can use the Configuration or Runtime tabs to specify the values for changes to take effect as shown in Figure 19-3:

- Configuration tab

Changes to the log and trace levels are effective the next time that the server is started.

- Runtime tab

Changes are effective immediately and are in effect until the server is stopped. When making changes to the Runtime tab, you can save the changes in the configuration, so that the changes are effective immediately and are applied in the future when the server is started. To do so, select the **Save runtime changes to configuration as well** check box.

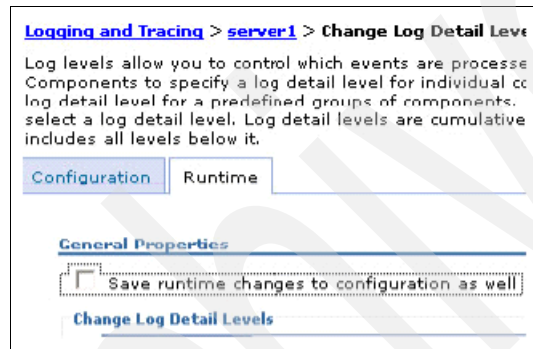


Figure 19-3 Configuration and Runtime tabs

Diagnostic trace

To configure a diagnostic trace, in the administrative console, in the left pane, select **Troubleshooting** → **Logs and Trace**. In the right pane, click the **WebSphere Process Server server name**, and click **Diagnostic Trace**.

To enable trace and modify the size and location of the trace file, specify the following values as shown in Figure 19-4 on page 566:

1. Select **Enable Log**.
2. For the Maximum Buffer Size, type 8.
3. Select **File**.
4. For the Maximum File Size, type 20.
5. For the Maximum Number of Historical Files, type 1.
6. For the File Name, type `${SERVER_LOG_ROOT}/trace.log`.

Logging and Tracing > server1 > Diagnostic Trace Service

Use this page to view and modify the properties of the diagnostic trace service. Diagnostic server components run within this managed process. Changes on the Configuration panel apply immediately.

Configuration Runtime

General Properties

☒ Enable Log

Trace Output

☐ Memory Buffer

* Maximum Buffer Size
8 thousand entries

☒ File

* Maximum File Size
20 MB

* Maximum Number of Historical Files
1

* File Name
\${SERVER_LOG_ROOT}/trace.log

Trace Output Format
Basic (Compatible)

Apply OK Reset Cancel

Figure 19-4 Diagnostic trace

Location of the JVM logs

The SystemOut.log and SystemErr.log files are in `WPS_install/profiles/profile_name/logs/server_name`. You can set the location and size of these files in the administrative console. In the left pane, select **Troubleshooting** → **Logs and Trace**. In the right pane, click the **WebSphere Process Server server name**, and select **JVM Logs**. Figure 19-5 on page 567 shows the Configuration tab of the JVM Logs section.

Logging and Tracing > server1 > JVM Logs

Use this page to view and modify the settings for the Java virtual machine (JVM) System.out and System.err logs for a managed process. The JVM logs are created by redirecting the System.out and System.err streams of the JVM to independent log files. The System.out log is used to monitor the health of the running application server. The System.err log contains exception stack trace information that is used to perform problem analysis. One set of JVM logs exists for each application server and all of its applications. JVM logs are also created for the deployment manager and each node manager. Changes on the Configuration panel apply when the server is restarted. Changes on the Runtime panel apply immediately.

Configuration **Runtime**

General Properties

System.out

* File Name:

File Formatting

Log File Rotation

<input checked="" type="checkbox"/> File Size	<input type="checkbox"/> Time
Maximum Size <input type="text" value="10"/> MB	Start Time <input type="text" value="24"/>
	Repeat Time <input type="text" value="24"/> hours

Maximum Number of Historical Log Files. Number in range 1 through 50.

Installed Application Output

☒ Show application print statements

☒ Format print statements

System.err

* File Name:

Log File Rotation

<input checked="" type="checkbox"/> File Size	<input type="checkbox"/> Time
Maximum Size <input type="text" value="1"/> MB	Start Time <input type="text" value="24"/>
	Repeat Time <input type="text" value="24"/> hours

Maximum Number of Historical Log Files. Number in range 1 through 50.

Installed Application Output

☒ Show application print statements

☒ Format print statements

Figure 19-5 SystemOut and SystemErr logs

Messages

The messages in WebSphere Process Server log and trace files are organized according to the identifier of the product feature that produces the message. Refer to Example 19-1 with the message CWTKE0036I.

Example 19-1 WebSphere Process Server SystemOut.log

```
[1/30/07 1:00:00:779 PST] 00000069 HTM          I    CWTKE0036I: The  
Staff Query Refresh Daemon has triggered 0 refresh operations.
```

Note the following explanations for Example 19-1:

- ▶ CWTKE is a five-character alphabetic component or application identifier.
- ▶ 0036 is a four-character numeric identifier that is used to identify the specific message for that component.
- ▶ I is an optional alphabetic severity indicator (I=Information, W=Warning, and E=Error).
- ▶ CWTKE0036I means that the people assignment refresh daemon (Staff Query Refresh Daemon) has triggered {0} refresh operations. This message indicates how many people assignments (staff query results) have been recalculated.

For detailed information about messages, see Messages in the WebSphere Process Server information center at the following address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wbpm.messages.620.doc/doc/welc_ref_msg_wbpm.html

19.2 Advanced troubleshooting methods

In this section, we explain known issues that you might encounter during migration that require advanced troubleshooting.

19.2.1 Adapter failed to start in a secured WebSphere Process Server

Deploying the migrated project into a secured WebSphere Process Server requires all of the necessary credentials for a Message-Driven Bean (MDB) and connection factories by using an authentication alias, SCA_Auth_Alias. Otherwise, the binding or connection-related exceptions can occur as shown in Example 19-2.

If you do not need to use security for your development environment, as a best practice, disable it as explained in 15.2.3, “Deployment” on page 319.

```
J2CA0138E: The MessageEndpoint activation failed for ActivationSpec
[9/7/06 1:20:22:328 PDT] 00000049 ActivationSpe E   J2CA0138E: The MessageEndpoint
activation failed for ActivationSpec
com.ibm.ws.sib.api.jmsra.impl.JmsJcaActivationSpecImpl and MDB Application
JTextConnectorApp#JTextConnectorEJB.jar#export.Input_SyncEIS, due to the following
exception: javax.resource.ResourceException: CWSIV0950E: An internal error occurred.
The exception com.ibm.wsspi.sib.core.exception.SINotAuthorizedException: CWSIP0303E:
No user specified when creating a connection to secure messaging engine
qaxs17Node01.server1-SCA.APPLICATION.qaxs17Node01Cell.Bus on bus
SCA.APPLICATION.qaxs17Node01Cell.Bus. was thrown while attempting to create a
connection on factory com.ibm.ws.sib.processor.impl.MessageProcessor@41a9e73b.
    at
com.ibm.ws.sib.ra.inbound.impl.SibRaMessagingEngineConnection.<init>(SibRaMessagingEn
gineConnection.java:228)
    at
com.ibm.ws.sib.ra.inbound.impl.SibRaEndpointActivation.getConnection(SibRaEndpointAct
ivation.java:362)
    at
com.ibm.ws.sib.ra.inbound.impl.SibRaStaticDestinationEndpointActivation.createListene
r(SibRaStaticDestinationEndpointActivation.java:669)
```

19.2.2 Failed to run the data synchronization scenario after migrating to WebSphere Adapter for JDBC

After migrating to WebSphere Adapter for Java Database Connectivity (JDBC) as explained in 16.2, “Implementation” on page 346 and running the data synchronization scenario end-to-end as explained in 16.3, “Testing the end-to-end solution” on page 419, you might encounter an error indicating that the new record is not inserted into the DB2 tables successfully. If you encounter this error, use the following steps to correct the problem:

1. Delete the records in the JDBCCustomer table.
2. Delete the relationship instance:
 - a. Select **Administrative Console** → **Integration Applications** → **Relationship Manager** → **Relationships**.
 - b. Select **Customer** relationship, click **Query**, and then, click **OK**.
 - c. Select an instance, and click **Delete**.
 - d. Repeat step c to delete all of the instances.
3. Run the data synchronization scenario again.

19.2.3 Failed to run the RetrieveCustomer on the data access scenario after migration

After migration when you run the RetrieveCustomer on the data access scenario, you might get the errors that are shown in Example 19-3. These errors are due to a problem in WebSphere Process Server V6.2 that will be fixed in WebSphere Process Server V6.2 Fix Pack.

Example 19-3 RetrieveCustomer errors

```
[4/7/09 20:39:41:296 PDT] 00000092 BaseMAP E
JDBCCustomerRetrieve_TO_CustomerRetrieve_Outbound_Response_SMO_Map
Transformation #2() CWLAS0015E: The Submap transform #2 in the
JDBCCustomerRetrieve_TO_CustomerRetrieve_Outbound_Response_SMO_Map map
did not complete because of the following exception:
com.ibm.ws.bo.bomodel.impl.DynamicBusinessObjectImpl incompatible with
java.util.List
```

WBI Adapters in a secured WebSphere Process Server environment

In this appendix, we explain how to reuse WebSphere Business Integration (WBI) Adapters with a secured WebSphere Process Server environment. We provide information about how to configure security credentials for the connector agent, as well as for the migrated artifacts and modules. We also describe the settings that are required for deploying migrated applications to secured WebSphere Process Server.

For background information, refer to Appendix C, “Migrate WBI Adapter for EJB Architecture” on page 615. This appendix describes how to configure a WebSphere Business Integration Adapter to work with a non-secured WebSphere Process Server environment.

You must be familiar with WebSphere Process Server deployment concepts to fully understand the following procedures.

This appendix contains the following sections:

- ▶ A.1, “Overview”
- ▶ A.2, “Configuring security”
- ▶ A.3, “Conclusion”

A.1 Overview

When reusing a WebSphere Business Integration Adapter with a secured WebSphere Process Server, follow up on several mandatory configuration steps. We describe the three major steps to consider:

- ▶ The WebSphere Business Integration Adapter connector agent must be able to connect to the secured WebSphere Process Server:
 - As a reminder, WebSphere Process Server is able to simulate an MQ manager because of the MQ client link. You must configure the connector agent to use Java Message Service (JMS), so that it is able to communicate properly with WebSphere Process Server through the MQ client link. You must configure the property `jms.MessageBrokerName` to use this syntax: `WBIA_QM:WBIA.JMS.SVRCONN:mqhost.mqport`. The variable `mqhost` is the IP address of the machine hosting WebSphere Process Server. The variable `mqport` is the value of the `SIB_MQ_ENDPOINT_ADDRESS` port that is defined in WebSphere Process Server.
 - You must set the `jms.UserName` and `jms.Password` properties to correspond to the `SCA_Auth_Alias` that is defined in WebSphere Process Server (by default, it is the `admin` user). This step is critical for a secured configuration. If the `UserName` and `Password` are set up incorrectly, the connector agent is unable to connect to WebSphere Process Server. A.2.2, “Setting up the connector agent with secured JMS” on page 574 provides an example.
- ▶ You must set up the WebSphere Process Server Service Integration Bus correctly:
 - The MQ client link communicates with the WebSphere Process Server internal messaging layer, which is called the *Service Integration Bus*.
 - Because the Service Integration Bus is also secured within a secured WebSphere Process Server environment, you must configure the Service Integration Bus to allow the use of an inbound MQ transport channel chain. A.2.4, “Allowing the MQ chain for the service integration bus” on page 582 provides an example.
- ▶ You must set up the WebSphere Process Server modules that are related to the adapters properly:
 - The WebSphere Process Server modules that correspond to the WBI Adapters contain JMS bindings. The JMS bindings communicate with the WebSphere Process Server internal messaging layer (called the Service Integration Bus), which is connected to the MQ client link.

- The JMS bindings rely on Message-Driven Beans (MDBs) to communicate with the Service Integration Bus queues. One MDB is generated for each JMS binding (import or export).
- Within a secured WebSphere Process Server environment, the Service Integration Bus is also secured. So, it is necessary to configure the MDBs to authenticate with the secured Service Integration Bus.
- The following sections provide a detailed example of the MDB configuration:
 - A.2.3, “Setting up a security identity qualifier in Message-Driven Beans” on page 576 for the setup in WebSphere Integration Developer
 - A.2.5, “Deployment of the connector modules” on page 584 for the deployment of the applications in the secured WebSphere Process Server environment

A.2 Configuring security

This section provides details about the major steps that are required to have a WebSphere Business Integration Adapter running with a WebSphere Process Server environment.

A.2.1 Initial assumptions

We assume that global security is enabled in the WebSphere Process Server environment.

We assume that the user name and password that are used correspond to the *admin* user in the WebSphere Process Server user registry.

To illustrate the security configuration steps, we use the customer synchronization scenario in this developerWorks article:

http://www.ibm.com/developerworks/websphere/library/techarticles/0511_mckinstry/0511_mckinstry.html

In this article, an inbound WBI Clarify connector sends events to be synchronized into the target SAP system, through the outbound WBI SAP connector. Figure A-1 on page 574 shows the flow of events for the customer synchronization scenario.

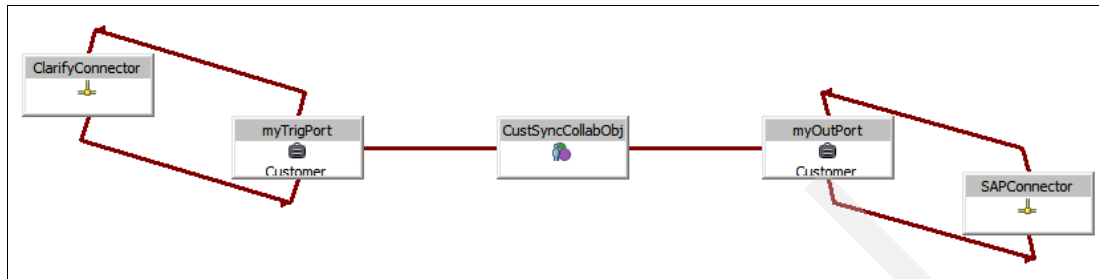


Figure A-1 Customer synchronization example from Clarify to SAP

A.2.2 Setting up the connector agent with secured JMS

The connector agent must be able to connect to the secured WebSphere Process Server through the MQ client link. For this purpose, it has to authenticate, and it is necessary to set up the `jms.UserName` and `jms.Password` properties so that they correspond to the `SCA_Auth_Alias` that is defined in WebSphere Process Server (by default, this alias is associated to the *admin* user, and the password is also *admin*). This step is critical for a successful secured configuration. If the `UserName` and `Password` are set up incorrectly, the connector agent is unable to connect to the secured WebSphere Process Server.

Figure A-2 on page 575 shows an example with the Clarify connector from our scenario. We use the Connector Configurator tool to edit the adapter properties. The major properties to consider are shown in blue. You must apply a similar setup to any WBI Adapter that is part of the secured WebSphere Process Server solution.

Connector Configurator - [WAS - ClarifyConnector*]

File Edit View Window Help

Target System: Windows

Standard Properties | Connector-Specific Properties | Supported Business Objects | Trace/Log Files | Data Handler

	Property	Value	Type	Subtype
1	AdapterHelpName		String	
2	AdminInQueue	CLARIFY/CONNECTOR/ADMININQUEUE	String	QueueName
3	AdminOutQueue	CLARIFY/CONNECTOR/ADMINOUTQUEUE	String	QueueName
4	AgentTraceLevel	0	Integer	
5	ApplicationName	ClarifyConnector	String	
6	BiDi.Transformation	false	Boolean	
7	BOTrace	None	String	
8	BrokerType	WAS	String	
9	CharacterEncoding	ascii7	String	
10	CommonEventInfrastructure	false	Boolean	
11	ContainerManagedEvents		String	
12	DeliveryQueue	CLARIFY/CONNECTOR/DELIVERYQUEUE	String	QueueName
13	DeliveryTransport	JMS	String	
14	DuplicateEventElimination	false	Boolean	
15	FaultQueue	CLARIFY/CONNECTOR/FAULTQUEUE	String	QueueName
16	jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory	String	AbsoluteClassName
17	jms.MessageBrokerName	WBIA_QM:WBIA.JMS.SVRCONN:localhost:5558	String	
18	jms.NumConcurrentRequests	10	Integer	
19	jms.Password	*****	String	
20	jms.UserName	admin	String	
21	Locale	en_US	String	
22	MessageFileName	InterchangeSystem.txt	String	FileName
23	PollEndTime	HH:MM	Time	
24	PollFrequency	10000	Integer	
25	PollStartTime	HH:MM	Time	
26	RepositoryDirectory	C:\IBM\reposDir	String	Directory
27	RequestQueue	CLARIFY/CONNECTOR/REQUESTQUEUE	String	QueueName
28	ResponseQueue	CLARIFY/CONNECTOR/RESPONSEQUEUE	String	QueueName
29	RestartRetryCount	3	Integer	
30	RestartRetryInterval	1	Integer	
31	RFH2MessageDomain	mrm	String	
32	SynchronousRequestQueue	CLARIFY/CONNECTOR/SYNCHRONOUSREQUESTQUEUE	String	QueueName
33	SynchronousRequestTimeout	0	Integer	
34	SynchronousResponseQueue	CLARIFY/CONNECTOR/SYNCHRONOUSRESPONSEQUEUE	String	QueueName
35	TivoliMonitorTransactionPerformance	false	Boolean	
36	WireFormat	CwXML	String	
37	WsifSynchronousRequestTimeout	0	Integer	
38	XMLNameSpaceFormat	short	String	

Figure A-2 Connector agent configuration for secured JMS communication

A.2.3 Setting up a security identity qualifier in Message-Driven Beans

The WebSphere Process Server modules that are related to the WBI adapters must be able to connect to the secured Service Integration Bus queues. For this purpose, the Message-Driven Bean (MDB) that is associated with each WBI adapter JMS binding (import or export) has to authenticate, and it must be set up with a security identity qualifier.

You must execute the following procedure for each WBI Adapter that is part of the solution, that is, for each module that is related to a WBI adapter. We illustrate the procedure with the customer synchronization scenario, in which we have two connector modules: ClarifyConnector and SAPConnector:

1. In WebSphere Integration Developer, within the workspace of your migrated solution, go to the Java Enterprise Edition (EE) perspective. Look at the available connector EJB modules. In our example, we have two connector EJB projects: ClarifyConnectorEJB and SAPConnectorEJB. Figure A-3 shows our two connector EJB modules.

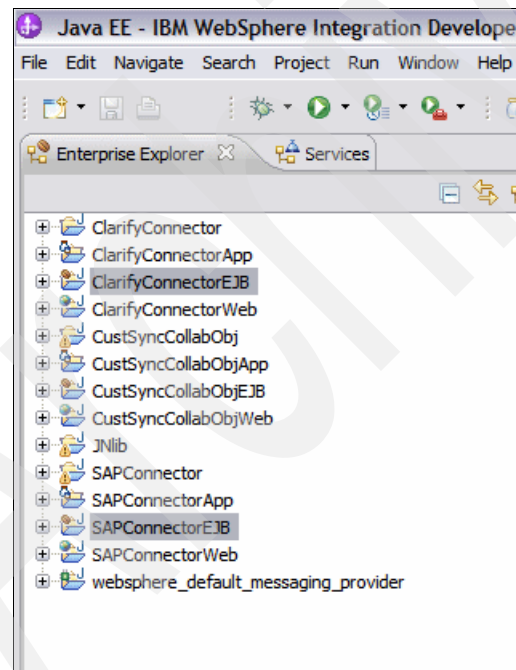


Figure A-3 Connector EJB modules in the Java EE perspective

2. Open the Deployment Descriptor for one connector EJB module. You must repeat the following steps for each relevant EJB module.
3. Click the **Assembly** tab.
4. In the **Security Roles** part, click **Add** (Figure A-4).

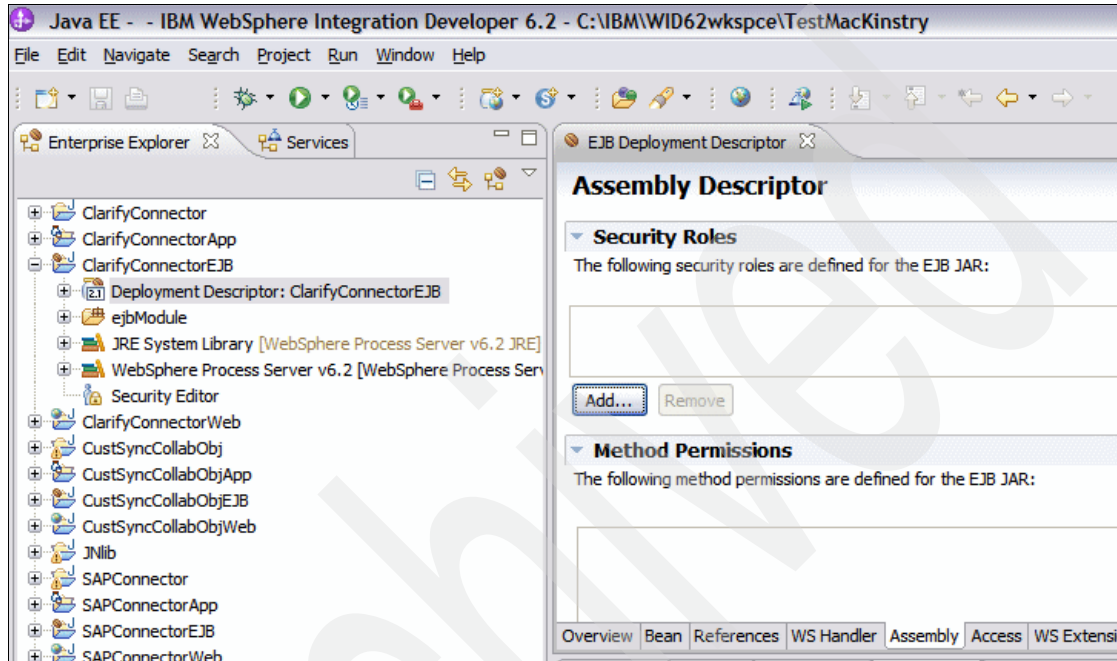


Figure A-4 Modifying the deployment descriptor and security roles

5. The Add Security Role window appears. Type Role4ICS in the Name field, and click **Finish** (Figure A-5). Note that you can set the Name to any value that you like. Document the name that you choose, because you will use it later in the security setup process.

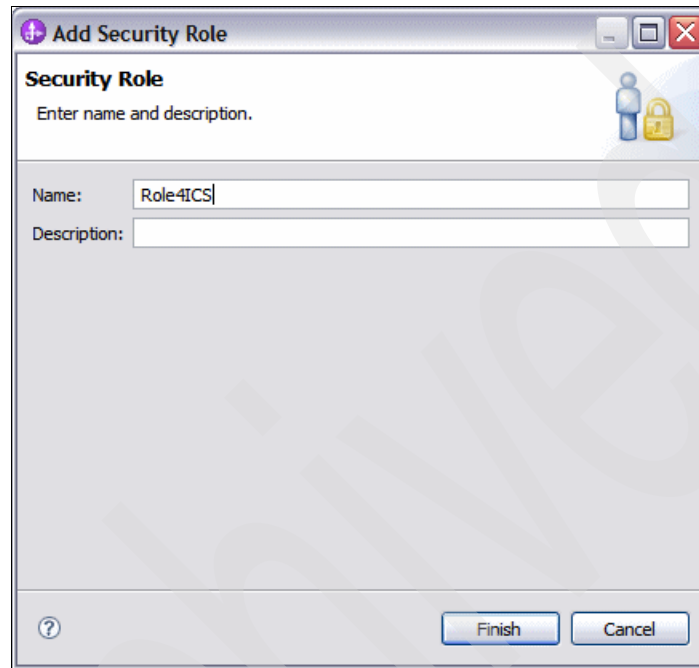


Figure A-5 Setting up a name for the Security Role

6. Click the **Access** tab. And then, click **Add** in the Security Identity (Bean Level) part (Figure A-6 on page 579).

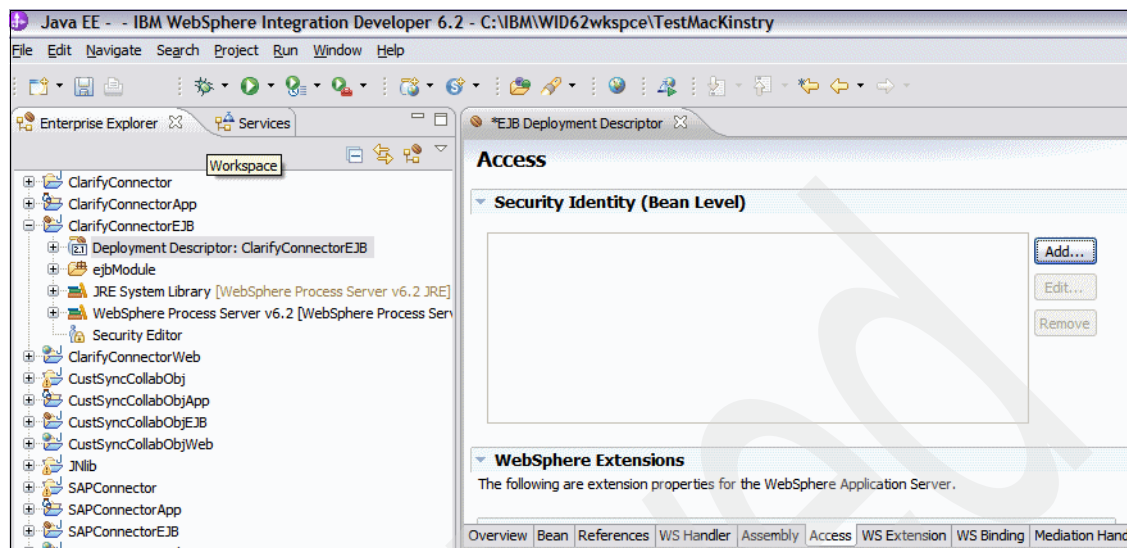
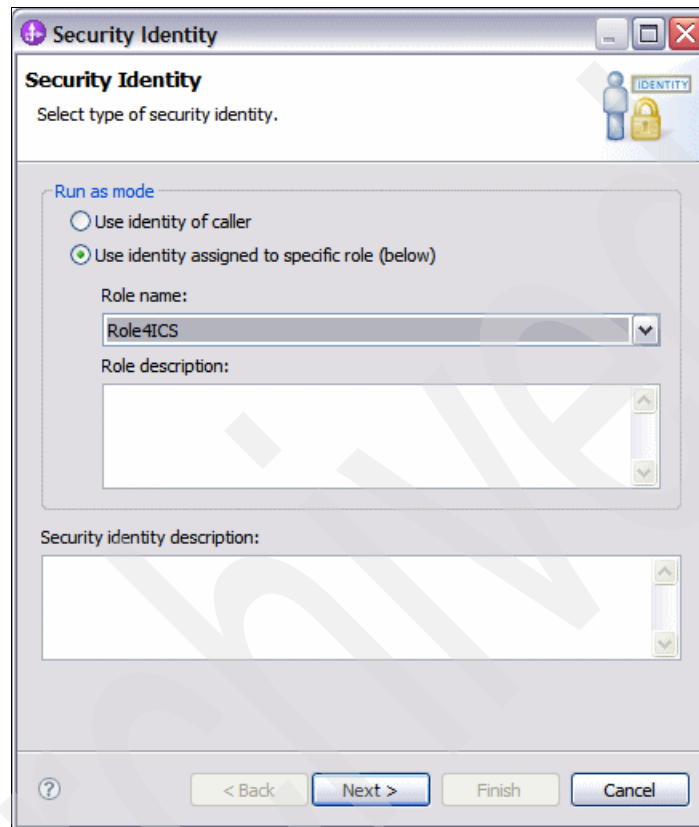


Figure A-6 Setting up the Security Identity

7. A new Security Identity window appears. Select **Use Identity assigned to specific role (below)**. For Role name as shown in Figure A-7, select **Role4ICS**, which you previously created. Click **Next**.



The image shows a 'Security Identity' dialog box. At the top, it says 'Security Identity' and 'Select type of security identity.' Below this, there are two radio buttons under the heading 'Run as mode'. The first is 'Use identity of caller' and the second is 'Use identity assigned to specific role (below)'. The second option is selected. Below the radio buttons, there is a 'Role name:' label and a dropdown menu showing 'Role4ICS'. Below that is a 'Role description:' label and a text area. At the bottom of the dialog, there is a 'Security identity description:' label and another text area. The bottom of the dialog has a question mark icon and four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Figure A-7 Setting up the Role name for Security Identity

8. In the Security Identity: Enterprise Bean Selection window, under **Beans found**, select the correct Message-driven beans. You can identify the relevant MDBs by their names. The name always starts with an underscore and is followed by JMS. After you have selected the correct MDBs, click **Finish**.

Figure A-8 and Figure A-9 on page 582 show two examples from our scenario. We have two EJB modules. The Clarify connector in the first module has one MDB associated with its export, and the SAP connector in the second module has one MDB associated with its import. Note that depending on your scenario, one WBI adapter can have several MDBs associated with it in one module.

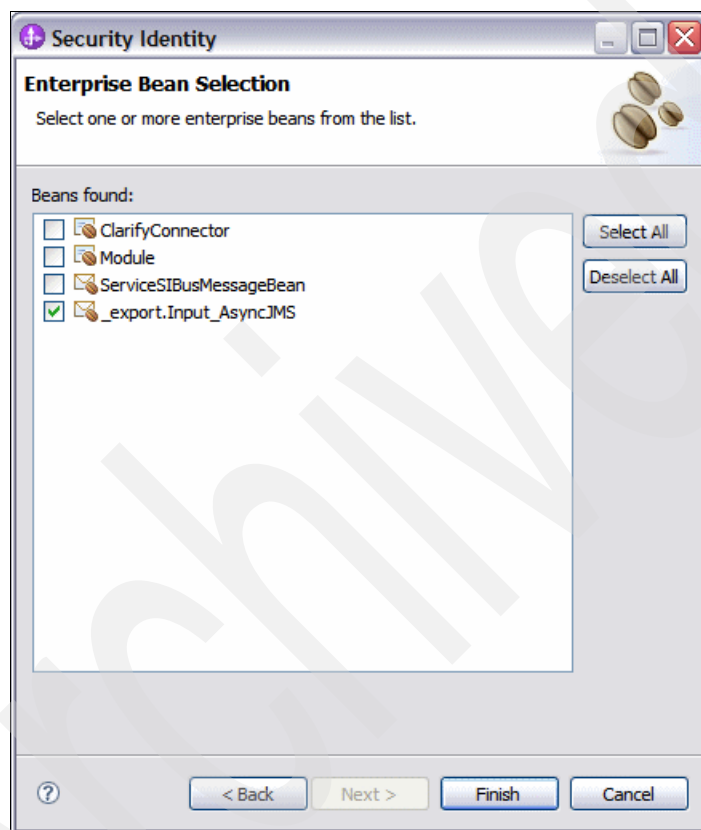


Figure A-8 Selecting the correct EJB components for the Clarify connector



Figure A-9 Selecting the correct EJB components for the SAP connector

9. Save all the changes in the deployment descriptor by typing Ctrl+s.

A.2.4 Allowing the MQ chain for the service integration bus

The MQ client link communicates with the WebSphere Process Server internal messaging layer, which is called the Service Integration Bus. Because the Service Integration Bus is also secured within a secured WebSphere Process Server environment, you must configure it to allow the use of an inbound MQ transport channel chain.

Follow these steps:

1. Open the administrative console of the WebSphere Process Server environment.
2. Go to the **Service integration** → **Buses** view (Figure A-10 on page 583).

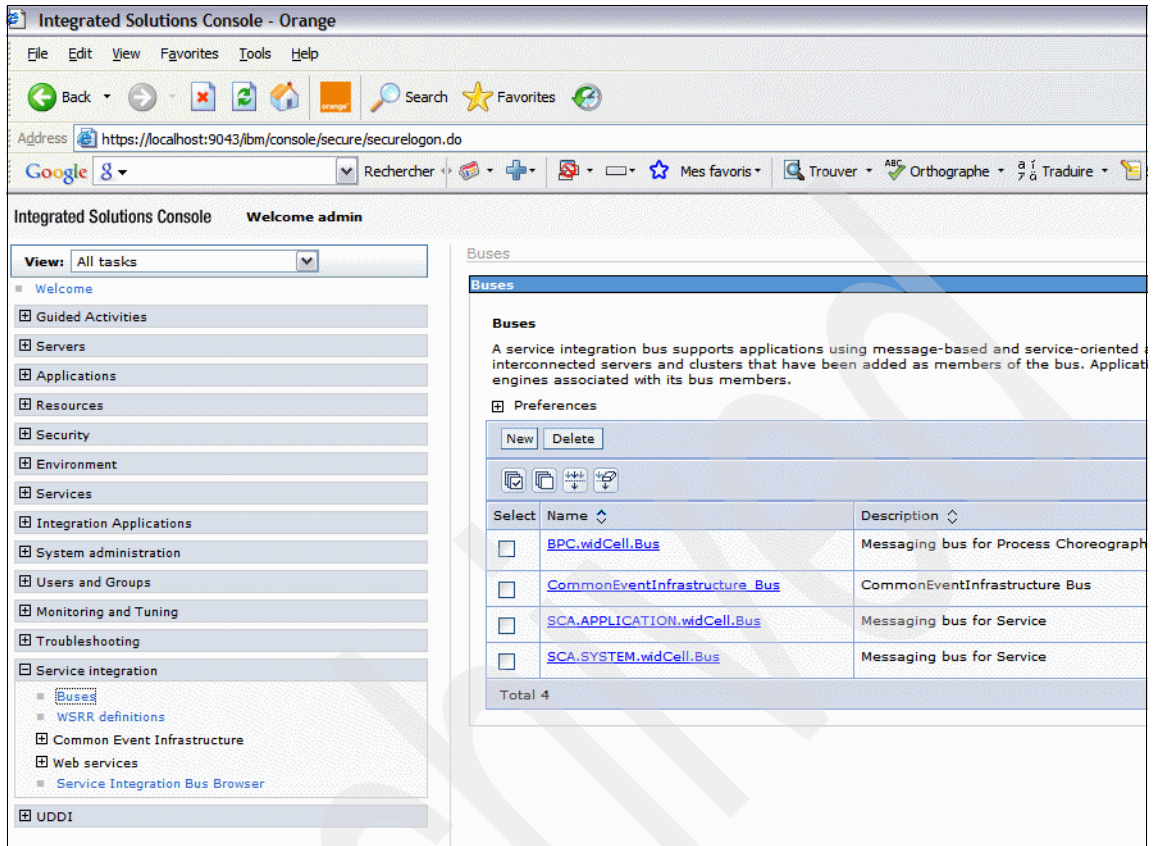


Figure A-10 Go to the Buses view in the administrative console

3. Click the **Enable bus security** check box as shown in Figure A-11. The WBI Adapters use only the SCA Application bus. Then, select **Allow the use of all defined transport channel chains**, and click **OK**. This action allows the MQ client link to communicate with the secured SCA Application bus (Figure A-11).

Buses

Buses > Security for bus SCA.APPLICATION.widCell.Bus

Configure the security settings for your service integration bus.

Configuration

General Properties

Security

☒ Enable bus security

Inter-engine authentication alias
SCA_Auth_Alias

Permitted transports

☒ Allow the use of all defined transport channel chains

☐ Restrict the use of defined transport channel chains to those protected by SSL

☐ Restrict the use of defined transport channel chains to the list of permitted transports

Mediations authentication alias
SCA_Auth_Alias

Additional Properties

- Users and groups in the bus connector role
- Permitted transports

Related Items

- JAAS - J2C authentication data
- Secure Administration and Applications

Apply OK Reset Cancel

Figure A-11 Selecting Allow the use of all defined transport channel chains

4. Then, click **Save** to save the configuration.

A.2.5 Deployment of the connector modules

In “Setting up a security identity qualifier in Message-Driven Beans” on page 576, we have defined a security identity role (Role4ICS) that must be used by the WebSphere Process Server modules related to the WebSphere Business Integration Adapters.

For each module, we now have to specify a real user to fulfill the role defined previously. You must perform this operation during the module deployment to the secured WebSphere Process Server environment.

The following description assumes that you deploy the module as an enterprise archive (EAR) file with the administrative console, using the “Show me all

installation options and parameters” option (so that all deployment steps are available).

Beware that the deployment step numbers that we provide might not be the same in your case. Therefore, we always put the complete name of the step beside the step number.

Follow this procedure. You must run this procedure for each connector module EAR file to deploy.

Start deploying your module EAR file by using the administrative console. Make sure that you select **Show me all installation options and parameters** before clicking **Next**, so that you can set up all of the required steps for security later (Figure A-12).

The screenshot shows the WebSphere Administrative Console interface. On the left is a navigation tree with a 'View:' dropdown set to 'All tasks'. The tree includes sections like 'Welcome', 'Guided Activities', 'Servers', 'Applications' (with sub-items 'Enterprise Applications', 'Install New Application', and 'SCA modules'), 'Resources', 'Security', 'Environment', 'Services', 'Integration Applications', 'System administration', 'Users and Groups', 'Monitoring and Tuning', 'Troubleshooting', 'Service integration', and 'UDDI'. The main content area is titled 'Enterprise Applications' and contains a dialog box 'Preparing for the application installation'. This dialog has a header 'Specify the EAR, WAR, JAR, or SAR module to upload and install.' and two main sections. The first section, 'Path to the new application', has two radio buttons: 'Local file system' (selected) and 'Remote file system'. Under 'Local file system', there is a 'Full path' text box containing 'C:\migration\directory\ju' and a 'Browse...' button. Under 'Remote file system', there is a 'Full path' text box and a 'Browse...' button. The second section, 'Context root', has a text box and a note 'Used only for standalone Web modules (.war files) a'. Below this is a section 'How do you want to install the application?' with two radio buttons: 'Prompt me only when additional information is required.' and 'Show me all installation options and parameters.' (selected). At the bottom of the dialog are 'Next' and 'Cancel' buttons.

Figure A-12 Selecting Show me all installation options and parameters

Complete these steps:

1. Accept all of the default values until you reach step 6 of the deployment page flow, which is “Bind listeners for Message-Driven Beans.”
2. In step 6, Bind listeners for Message-Driven Beans, of the deployment page flow, bind each relevant MDB to an activation specification authentication alias. You have to select the same MDBs as the MDBs that you selected in step 8 of “Setting up a security identity qualifier in Message-Driven Beans” on

page 576. You have to type `SCA_Auth_Alias` as the value for Activation spec authentication alias. Figure A-13 shows an example with the Clarify connector export from our customer synchronization scenario.

Object	EJB module	EJB	URI	Messaging type	Bindings
	ClarifyConnectorEJB	ServiceSIBusMessageBean	ClarifyConnectorEJB.jar,META-INF/ejb-jar.xml	com.ibm.wsspi.sib.ra.SibRaMessageListener	<div><input type="radio"/> Listener port Name <input type="text"/></div> <div><input checked="" type="radio"/> Activation Specification Target Resource JNDI Name <input type="text" value="sca/ClarifyConnector/Act"/> Destination JNDI name <input type="text"/> ActivationSpec authentication alias <input type="text"/></div>
	ClarifyConnectorEJB	_export.Input_AsyncJMS	ClarifyConnectorEJB.jar,META-INF/ejb-jar.xml	javax.jms.MessageListener	<div><input type="radio"/> Listener port Name <input type="text"/></div> <div><input checked="" type="radio"/> Activation Specification Target Resource JNDI Name <input type="text" value="ClarifyConnector/Input_A"/> Destination JNDI name <input type="text" value="CLARIFYCONNECTOR/DE"/> ActivationSpec authentication alias <input type="text" value="SCA_Auth_Alias"/></div>

Figure A-13 An example of setting up the authentication alias for the activation spec of the MDBs

3. Perform these steps in step 9, Map resource references to resources, of the page flow (Figure A-14 on page 588):
 - a. Under Specify authentication method, select **Use default method (many-to-one-mapping)**. Then, select **SCA_Auth_Alias** for the Authentication data entry.
 - b. Select the check box of the line corresponding to the connection factory (it is the line where the Resource Reference name ends with _CF).
 - c. Click **Apply**.

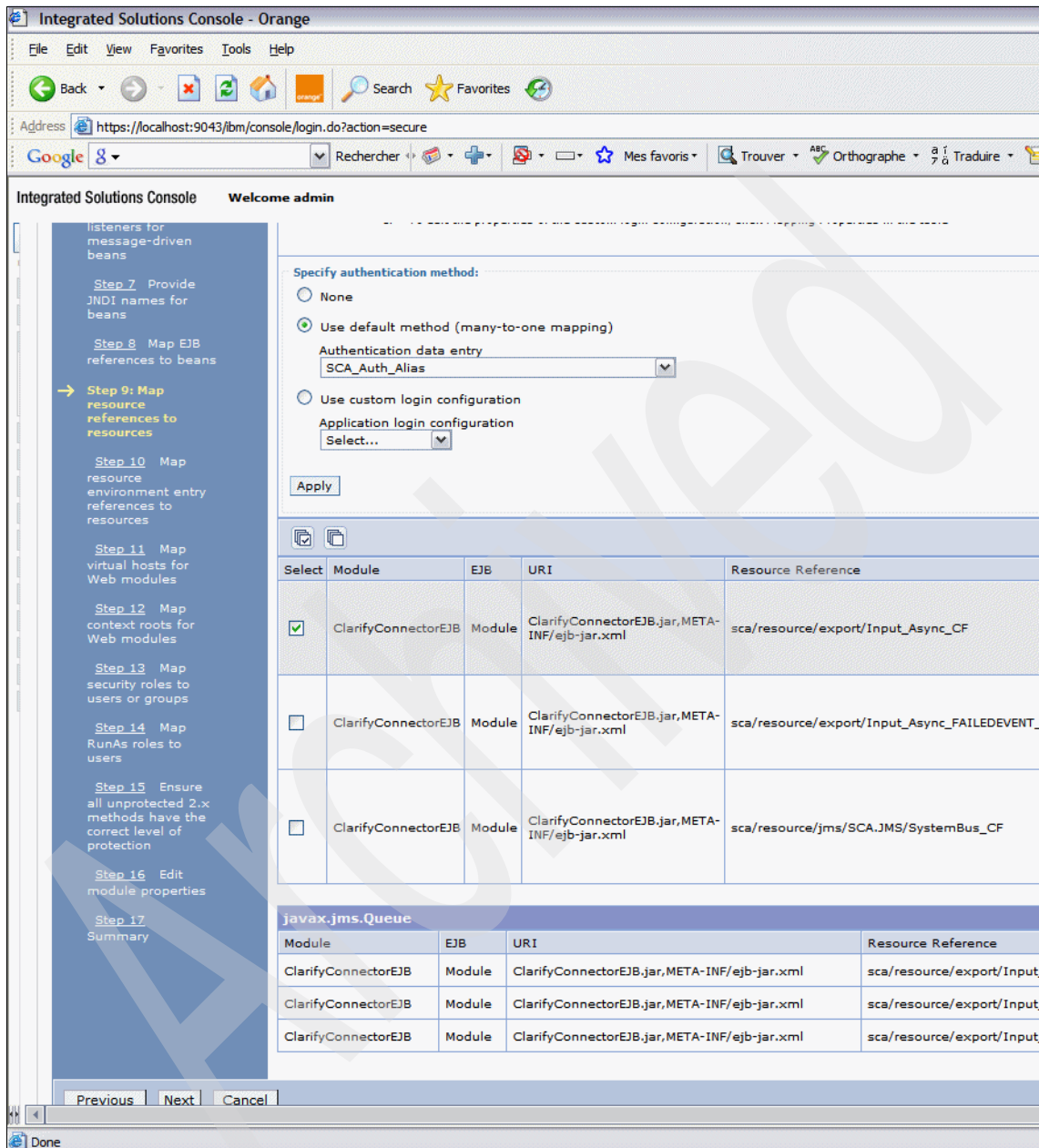


Figure A-14 Setting up the authentication alias for the connection factory

4. In step 13, Map security roles to users or groups, of the page flow, in the Map security roles to users or groups pane (Figure A-15), click **Look up users** to map the **Role4ICS** role to the admin mapped user. Then, click **Next**.

Integrated Solutions Console Welcome admin

Enterprise Applications

Install New Application

Specify options for installing enterprise applications and modules.

Step 13: Map security roles to users or groups

Map security roles to users or groups

Each role that is defined in the application or module must map to a user or group from the domain user.

Select	Role	Everyone?	All authenticated?	Mapped users
<input type="checkbox"/>	Role4ICS	<input type="checkbox"/>	<input type="checkbox"/>	admin

Done

Figure A-15 Mapping the security role with an actual user

5. In step 14, Map RunAs roles to users, in the Map RunAs roles to users pane (Figure A-16 on page 590), specify the user from the user registry (admin user and the same password, in our example), select the listed role (**Role4ICS** role, in our example), and click **Apply**.

Map RunAs roles to users

The enterprise beans or servlet that you are installing contain predefined RunAs roles. Some enterprise beans or servlet use roles to run as a particular role that is recognized when interacting with another enterprise bean.

username

password

Remove the RunAsUser user name and password from the selected roles.

Select	Role	User name
<input type="checkbox"/>	Role4ICS	admin

Figure A-16 Mapping the RunAs role to an actual user

6. Finish the application installation and save the changes.

A.3 Conclusion

The procedures given in this appendix described the major steps that are required to use an existing WebSphere Business Integration Adapter within a secured WebSphere Process Server environment.

Access EJB migration

In this appendix, we show an example of the standard support to migrate WebSphere InterChange Server applications that use the WebSphere Access Enterprise JavaBean (EJB), or Access EJB, feature. We also show testing and verification of the migrated sample application. Remember that you can call a WebSphere InterChange Server collaboration synchronously from any Java 2 Platform, Enterprise Edition (J2EE) client through the use of an EJB session that is provided as part of the WebSphere InterChange Server product. This EJB is the main component of the WebSphere Access EJB.

The standard migration tools are one option among others to migrate WebSphere InterChange Server applications to WebSphere Process Server. See Part 2, “Migration implementation concepts” on page 71, for an overall discussion about the possible options for migration implementation. Part 3, “Migration tooling” on page 177 provides detailed information about the standard migration tools.

We present a simple example in this chapter to illustrate a complete upgrade that uses the standard migration tools. In reality, complex WebSphere InterChange Server solutions generally require changes before or after migrating. You must manually make changes either to the WebSphere InterChange Server artifacts before the migration or to the generated WebSphere Process Server artifacts after the migration.

We include the following sections in this appendix:

- ▶ B.1, “Target environment” on page 592
- ▶ B.2, “Access EJB example” on page 593
- ▶ B.3, “Conclusion” on page 614

B.1 Target environment

Table B-1 identifies the software that we used in the target environment.

Important: We executed this example with the product versions that are shown in Table B-1. With newer product versions, there might be slight differences in windows and wizards.

Table B-1 Products installed in the software environment

Software	Version
WebSphere Process Server	6.2
WebSphere InterChange Server and toolkit	4.3.0.5
WebSphere Integration Developer	6.2
WebSphere Adapter Framework	2.6.0.11
DB2 Universal Database	8.1.16 (or 8.2.9)
WebSphere MQ Client	6.0.2.4
Microsoft Windows XP Professional	SP 2
WebSphere Business Integration Data Handler for XML	2.7.3

B.2 Access EJB example

In the Access EJB example, a J2EE client called *AccessServlet* calls the AccessEJB that is hosted in WebSphere Process Server. The session bean, consisting of the `executeCollaborationExtended()` remote method, invokes the `access_collab` collaboration through its port `XmlPort` as shown in Figure B-1.

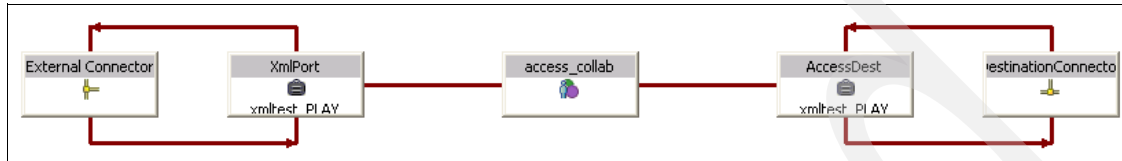


Figure B-1 Access EJB example overview

The collaboration, in turn, calls `DestinationConnector` to retrieve an event. The response received from the connector is sent to the `AccessServlet` through the collaboration.

`DestinationConnector` is a simple test connector that simulates an Enterprise Information System (EIS)-specific application connector. Upon receiving an event from the access collaboration with `Retrieve` as the operation verb, it returns the same business object as a response.

B.2.1 Preparation

In this section, we discuss the initial steps that are required to migrate the Access EJB example to the WebSphere Process server target environment.

Examining the WebSphere InterChange Server repository file

The WebSphere InterChange Server repository file, `AccesCollab_ICL.jar`, is in the compressed files that are available for download as explained in Appendix D, “Additional material” on page 651. You can see the artifacts by using the WebSphere InterChange Server System Manager tool.

The AccesCollab_ICL.jar file has the following contents, grouped by component names:

Java files: Any .java files that are generated by WebSphere InterChange Server when compiling the artifacts are not used by the migration support. The files are ignored.

- ▶ **BusinessObject:**
 - xmltest_PLAY. The xmltest_PLAY business object
 - Employee. The employee business object
 - Company. The company business object
 - Department. The department business object
 - MO_Datahandler.*. The metadata business objects for data handlers
 - MO_Server_DataHandler. The metadata business objects that are used by the server
- ▶ **Collaboration**
access_collab.cwc. The collaboration object that shows the wiring of an external EJB call and destination connector with an access collaboration template
- ▶ **CollaborationTemplate**
access.cwt. The collaboration template that defines the business process for the Access EJB example
- ▶ **Connector**
DestinationConnector.con. The connector definition for the destination adapter

Modifying the business objects

The business objects that are part of the AccesCollab_ICL.jar file require minor modifications in order to work with WebSphere Process Server. To modify the business objects:

1. Extract the BusinessObject and Connector folders from the AccesCollab_ICL.jar file to a directory. In this example, we use C:\ServerAccessInterface_EJBexample.
2. Rename the XML files in the BusinessObject folder to have a file extension of XSD.

Modifying the adapter configuration files

The connector configuration file requires several changes. To modify the file:

1. By using the connector configurator tool, open the `DestinationConnector.con` file from the `C:\ServerAccessInterface_EJBexample\repository\Connector` folder.
2. Change the following properties:
 - a. For `BrokerType`, type `WAS`.
 - b. For `jms.MessageBrokerName`, change the value to `WBIA_QM:WBIA.JMS.SVRCONN:localhost:5560`.

The number 5560 should match the port number for the `SIB_MQ_ENDPOINT_ADDRESS` port in WebSphere Process Server. You can look up this value for your installation in the administrative console by selecting **Application servers** → **server1** → **Ports**.
 - c. In the `jms.Username` field, type any value.

This sample runs without security enabled, and the value in this field does not matter, as long as the field is not empty.
 - d. Change `RepositoryDirectory` to the location of your business objects, for example `C:\ServerAccessInterface_EJBexample\repository\BusinessObject`.
3. Save the changes.

Importing the AccessEJB project

To have the AccessEJB project work with WebSphere Process Server:

1. Copy the `CwDataHandler.jar` and `CwXmlDataHandler.jar` data handler Java archive (JAR) files into the WebSphere Process Server library, which is `WPS_HOME\lib`.
2. Under the Business Integration perspective, select **File** → **Import**. Select **Project Interchange**, and click **Next**.
3. Browse to locate the `WPS_HOME\HeritageAPI\ AccessEJB.zip` file as an input compressed file, and click **Next**.

4. Click **Select All**, and then, click **Finish** (Figure B-2).

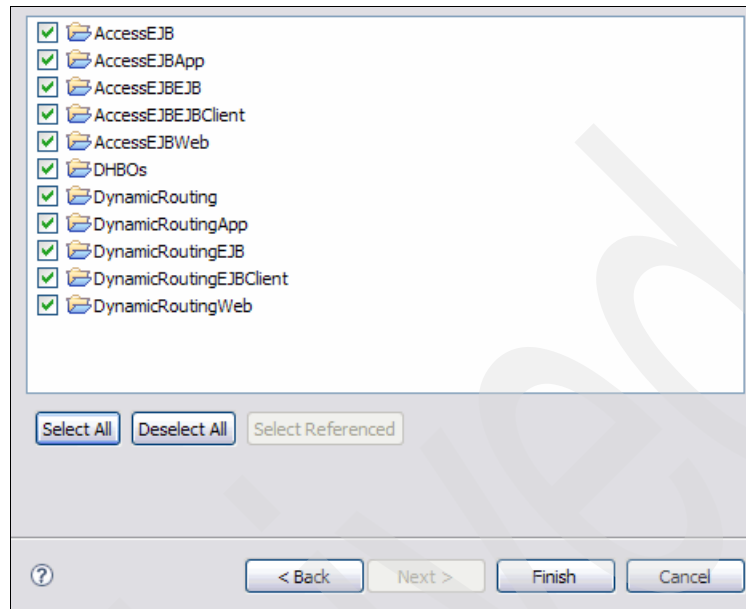


Figure B-2 Importing the AccessEJB project interchange

5. Right-click the **DynamicRouting** module, and select **Properties**.
6. In the Properties for DynamicRouting window, click **Java Build Path**, and go to the **Library** tab.

7. Click **Add External JARS** to add the data handler JAR files to the Java build path for the DynamicRouting module as shown in Figure B-3.

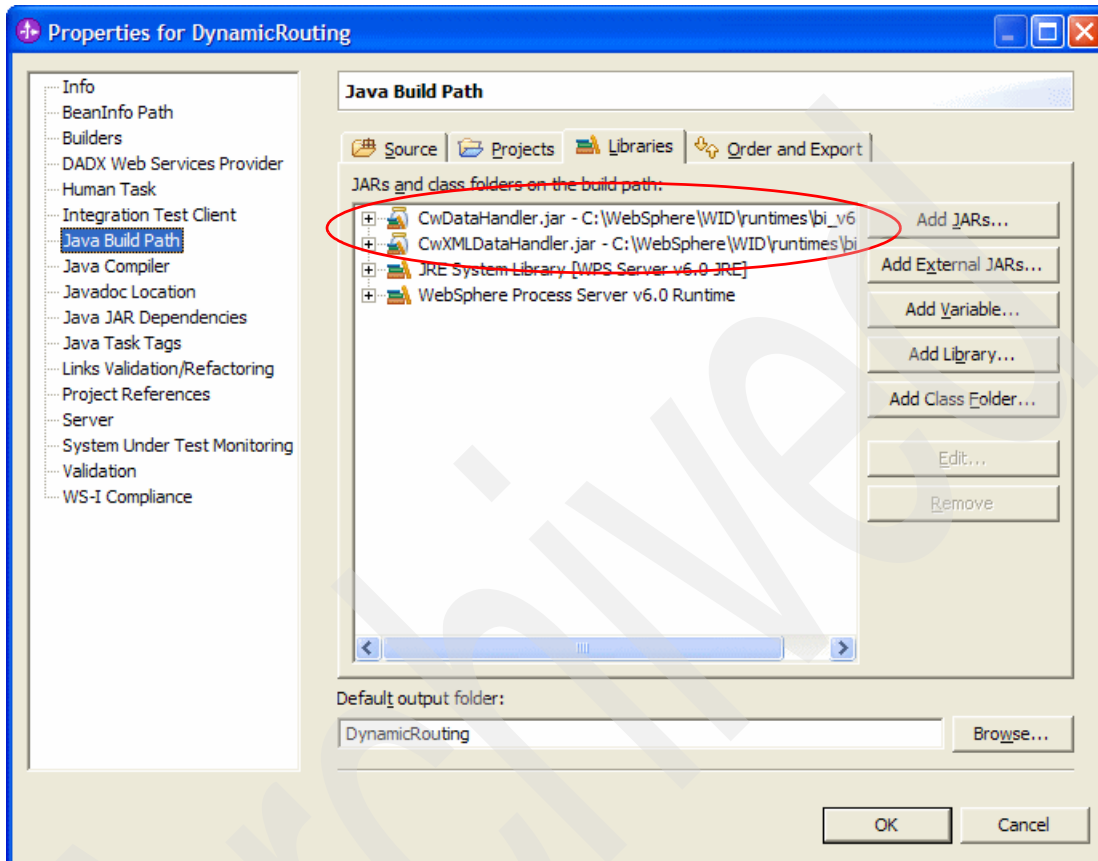


Figure B-3 Adding the data handler JAR files to the build path

Contents of the AccessEJB project interchange file

In this section, we discuss the contents of the AccessEJB project that comes with the WebSphere Process Server product.

AccessEJB assembly diagram

Figure B-4 shows the assembly diagram of the AccessEJB project.

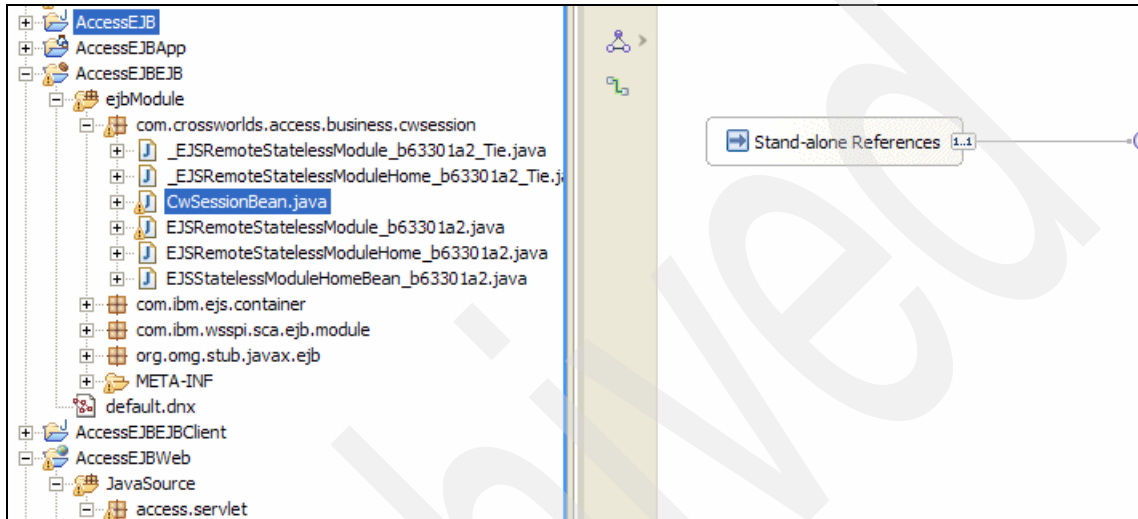


Figure B-4 AccessEJB assembly diagram

The EJB project contains CwSessionBean and its relevant CwSessionHome home and CwSession remote interfaces. The Web project contains the J2EE client AccessServlet.

The servlet calls the executeCollaborationExtended() remote method in the CwSession bean to invoke the access_collab Business Process Execution Language (BPEL) process through the selector in the DynamicRouting module.

DHBOs library

DHBOs is a common library that holds all of the shared artifacts, such as business objects, maps, and interfaces that are used by the AccessEJB modules. DHBOs includes the data handler metadata business objects that are required to handle various Multipurpose Internet Mail Extensions (MIME) types.

DynamicRouting selector module

DynamicRouting is a selector module that is used to dynamically select Service Component Architecture (SCA) targets (BPELs) based on the collaboration name and port name that are passed to the AccessEJB. Figure B-5 shows the assembly diagram.

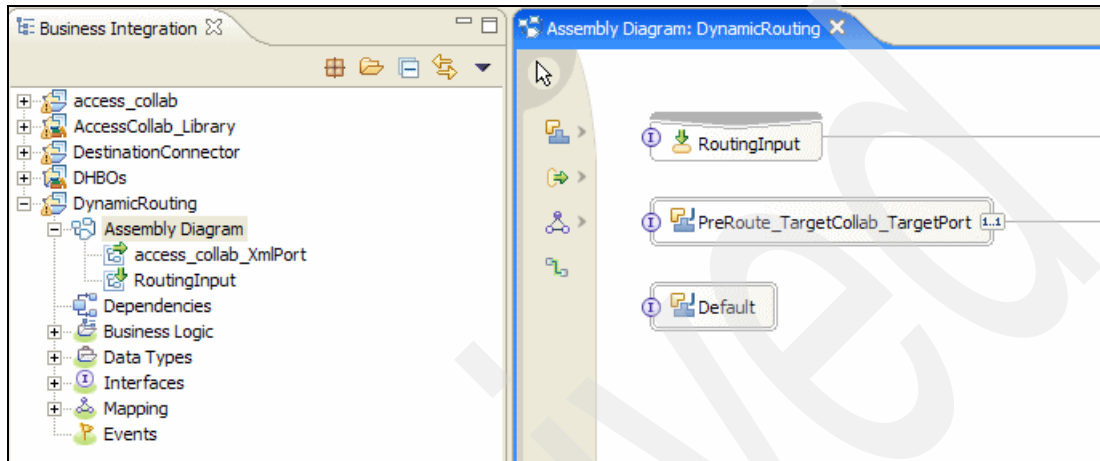


Figure B-5 Assembly diagram of the DynamicRouting module

Inputs specified in the Access Servlet J2EE client

In this example, *access_collab* is passed as the collaboration name and *XmlPort* is passed as Portname in the remote method call to the session bean:

```
executeCollaborationExtended("", access_collab, XmlPort, data,  
text_xml, Retrieve);
```

In this example, the parameter data is the string representation of the XML file that is passed as the triggering business object into the collaboration.

Copy and paste the AccesEJB_InputFile.xml file that is in C:\. The file name and the path to the file are specified in the AccessServlet.java file. Example B-1 shows the contents of the XML file.

Example B-1 Input XML file

```
<?xml version="1.0" ?>  
<!DOCTYPE PLAY>  
<PLAY>  
<CNAME>IBM</CNAME>  
<CPROD>Software</CPROD>  
<CNUM>1000</CNUM>  
<CLOC>Miami</CLOC>
```

```
<CTURN>12345789.00</CTURN>
<DEPT>
<DID>Adapters</DID>
<DNAME>120</DNAME>
<DGOOD>true</DGOOD>
<EMP>
<EID>420</EID>
<ENAME>JoeSomebody</ENAME>
<EAGE>36</EAGE>
<EADDRESS>577 Airport Blvd Burlingame CA</EADDRESS>
</EMP>
</DEPT>
</PLAY>
```

B.2.2 Implementation

In this section, we explain how to migrate the AccessEJB technical example.

Migrating the artifacts

To migrate the artifacts:

1. Open WebSphere Integration Developer in the Business Integration Perspective.
2. Click **File** → **Import**.
3. Select **WebSphere InterChange Server Repository**, and click **Next**.
4. In the next window, specify the AccessCollab_ICL.jar repository JAR file to be migrated. Enter AccessCollab_Library as the new library name, and click **Next**.
5. In the Configure Connector Migration window, accept the default value for DestinationConnector migrating to JMS to Undetermined WBI Adapter, and click **Next**.
6. In the Conversion Options window, choose all of the recommended options, and click **Finish**.

The Migration Wizard successfully migrates the WebSphere InterChange Server artifacts to the WebSphere Process Server artifacts that reside in two projects and a library. The artifacts that are used commonly across all the projects are stored in the library as shown in Figure B-6 on page 601.

Building the projects might take a while. After all of the the projects are built, warning messages are displayed on the Problems tab. Although these APIs are all deprecated, they are still supported in WebSphere Process Server.

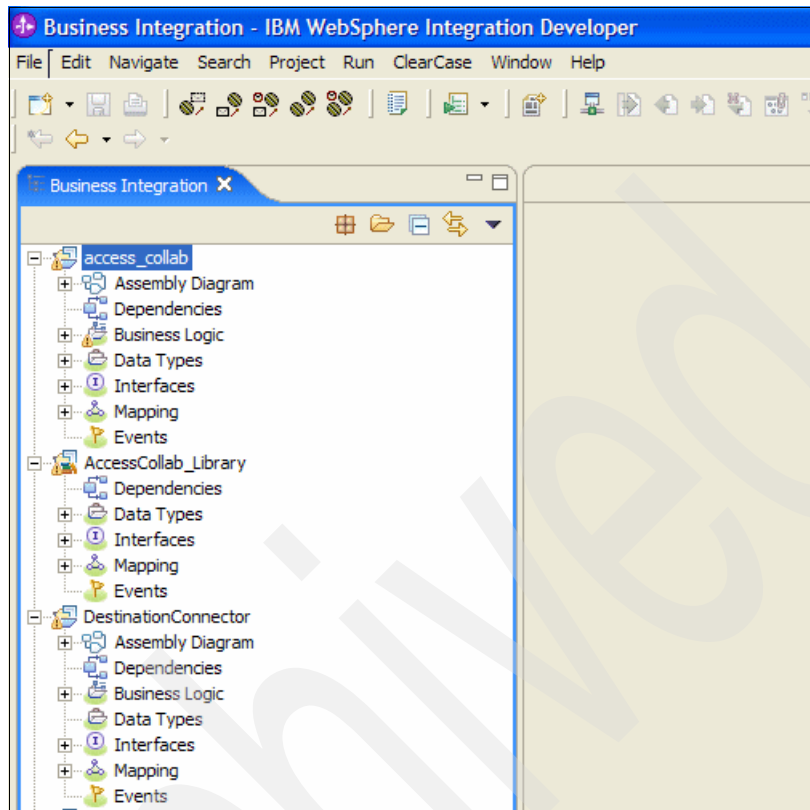


Figure B-6 Generated projects after migration

Event flow through the migrated projects

Figure B-7 shows the flow of events, end to end, through the various applications and the components that are involved in the AccessEJB example.

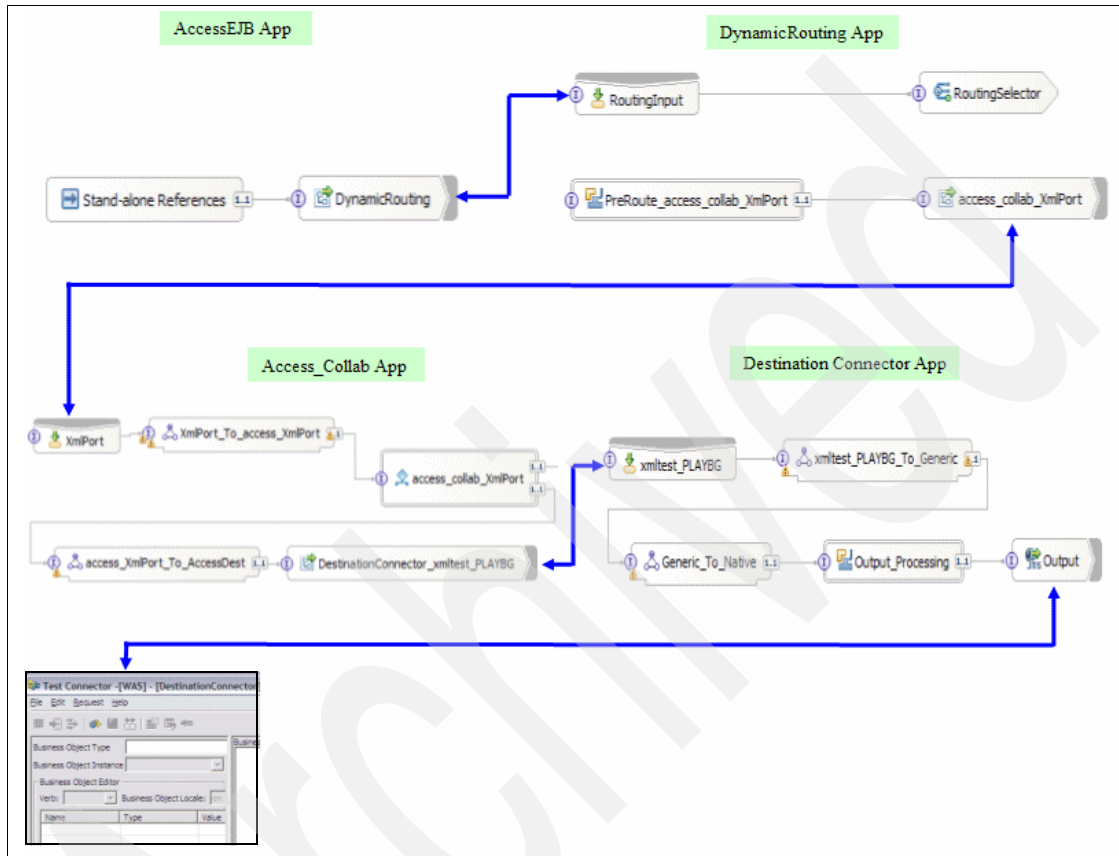


Figure B-7 End-to-end flow of events through the examples

Customizing the contents of the AccessEJB project

In the following sections, we explain how to customize the contents of the AccessEJB project.

Adding the AccessCollab_Library to the DynamicRouting module

AccessCollab_Library is a required library, because the selector invokes the access_collab module and needs references to the dependent files. To add the library:

1. Select the **DynamicRouting** module, right-click, and select **Open Dependencies**.
2. Click **Add**.
3. From the Library Selection Window, select **AccessCollab_Library**. Then, click **OK**.
4. Save and close the Dependencies window.

AccessCollab_Library is added to the module along with DHBOs, as shown in Figure B-8.

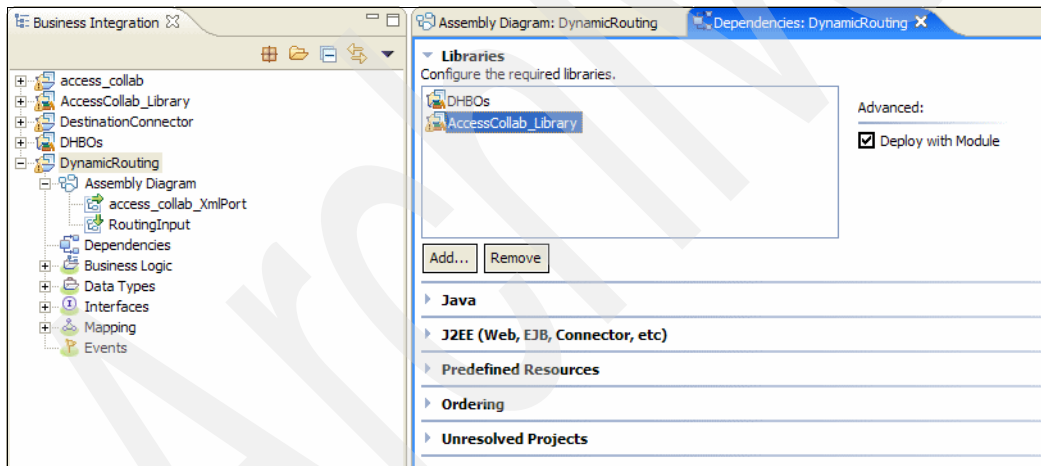


Figure B-8 Adding a dependent library

Creating references to the migrated process module

To create the references:

1. Open the assembly diagram of the DynamicRouting module.
1. Drag the **XmlPort** export component of the access_collab module to the assembly diagram of the DynamicRouting module as an import component.
2. In the window that opens, select **Import with SCA Binding**, and click **OK**.

3. Rename the newly created import to `access_collab_XmlPort`.
4. Copy and paste the existing **PreRoute_TargetCollab_TargetPort** component in the assembly diagram of the DynamicRouting module. Rename it to `PreRoute_access_collab_XmlPort`.
5. On the `PreRoute_access_collab_XmlPort` component, remove the existing reference. Click the reference (small box on the right side of the component containing "1..1"), right-click, and select **delete**.
6. Wire the `PreRoute_access_collab_XmlPort` to `access_collab_XmlPort`. When you see a prompt regarding a new reference that will be created on the source, click **OK**.
A new reference called `xmltest_PLAYBG_PortTypePartner` is added to invoke the `access_collab` module as shown in the assembly diagram in Figure B-9.
7. Save the changes.

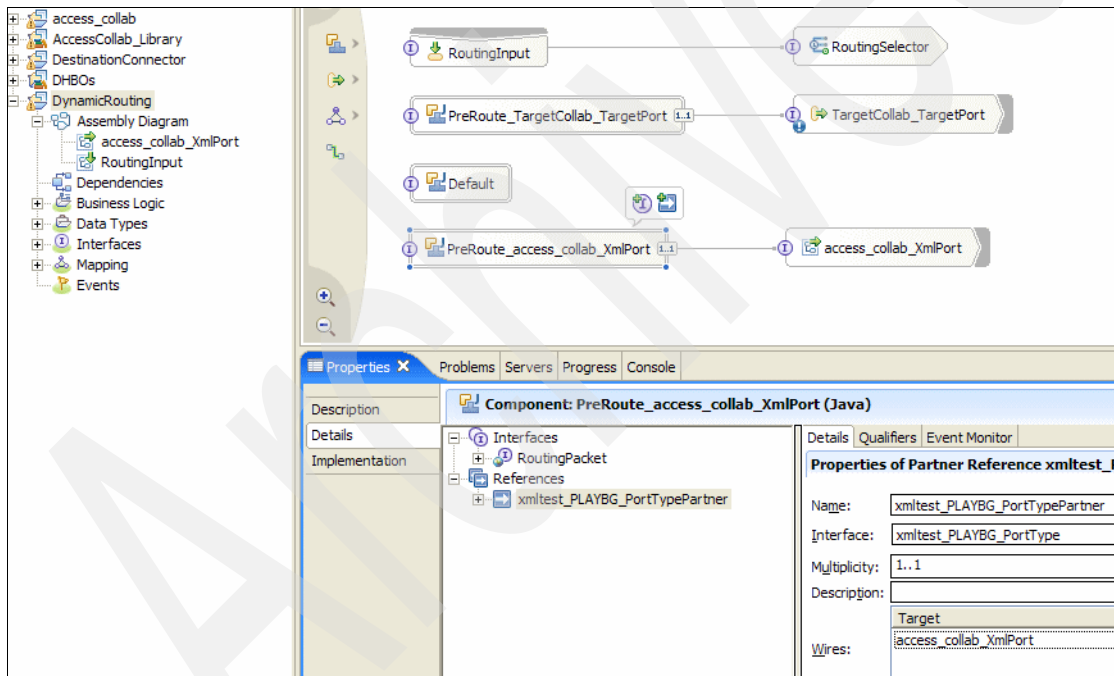


Figure B-9 Modified assembly diagram of the DynamicRouting module

Updating the selector file

Update the existing selector component in the DynamicRouting module with the appropriate collaboration and port names, in order to invoke the `access_collab` BPEL process:

1. Change the perspective to **Java**.

2. Expand **DynamicRouting** → **com.ibm** and select **RoutingSelector.selt** in the Text Editor as shown in Figure B-10.

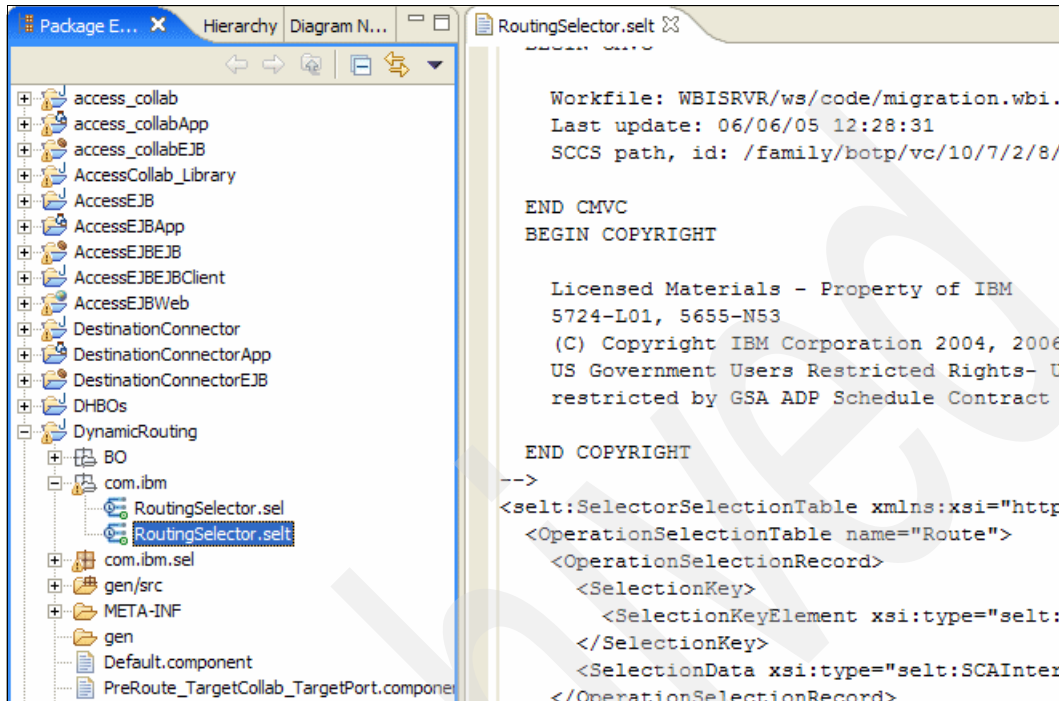


Figure B-10 Selector file in the Java perspective

3. Copy and paste the **OperationSelectionRecord** block.

4. In the new block, change the value from TargetCollab_TargetPort to access_collab_XmlPort. In the same block, change componentName from PreRoute_TargetCollab_TargetPort to PreRoute_ access_collab_XmlPort as shown in Figure B-11.

```
<OperationSelectionTable name="Route">
  <OperationSelectionRecord>
    <SelectionKey>
      <SelectionKeyElement xsi:type="selt:StringSingletonKey" value="TargetCollab_TargetPort"/>
    </SelectionKey>
    <SelectionData xsi:type="selt:SCAInternalComponent" componentName="PreRoute_TargetCollab_TargetPort"/>
  </OperationSelectionRecord>
  <OperationSelectionRecord>
    <SelectionKey>
      <SelectionKeyElement xsi:type="selt:StringSingletonKey" value="access_collab_XmlPort"/>
    </SelectionKey>
    <SelectionData xsi:type="selt:SCAInternalComponent" componentName="PreRoute access_collab_XmlPort"/>
  </OperationSelectionRecord>
  <DefaultSelectionData xsi:type="selt:SCAInternalComponent" componentName="Default"/>
</OperationSelectionTable>
</selt:SelectorSelectionTable>
```

Figure B-11 Additions to the selector file

5. Save the changes.

Specifying the Java implementation file

To specify the Java implementation file:

1. Expand the **com.ibm.sel** folder.
2. Copy and paste **PreRoute_TargetCollab_TargetPortImpl.java** to the same folder.

3. As shown in Figure B-12, rename the Java file to `PreRoute_access_collab_XmlPortImpl`.

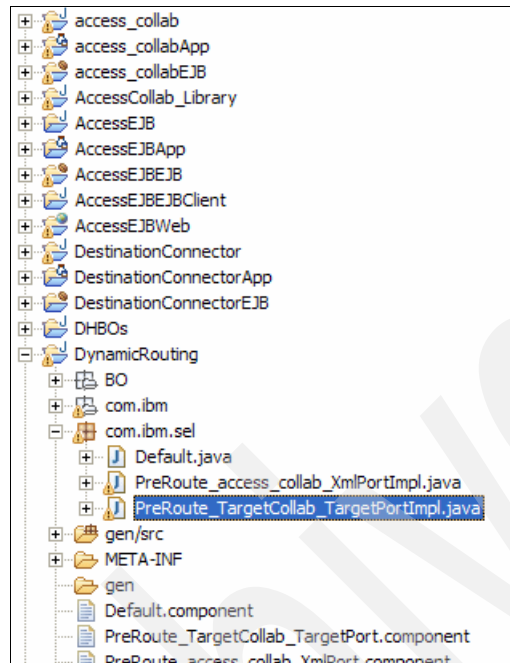


Figure B-12 Creating a new Java implementation file

4. Change to the **BusinessIntegration** perspective, and open the assembly diagram for the `DynamicRouting` module.
5. Select **PreRoute_access_collab_XmlPort** component.
6. In the Properties view, click the **Implementation** tab.

7. As shown in Figure B-13, on the **Details** subtab, in the Class field, enter the newly created implementation file name:
`com.ibm.sel.PreRoute_access_collab_XmlPortImpl`

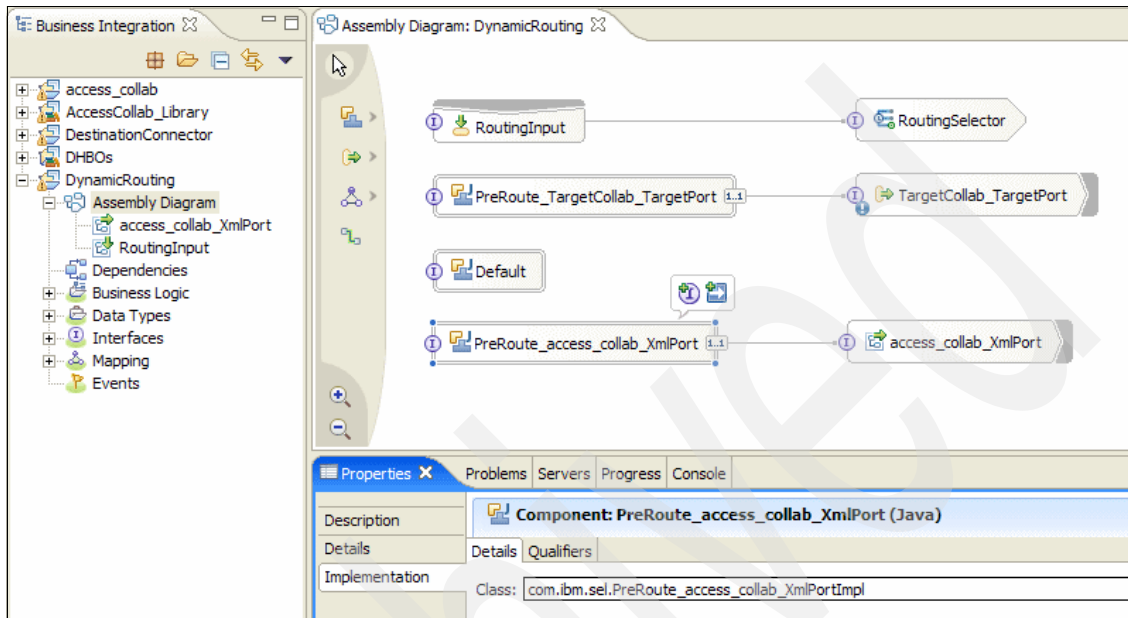


Figure B-13 Specifying the Java implementation file in the component

Modifying the implementation files

To modify the implementation files:

1. Open the **PreRoute_access_collab_XmlPortImpl.java** file.
2. Change the method name of `locateService.TestBOInterfacePartner` to `locateService_xmltest_PLAYBG_PortTypePartner`.

3. Change all instances of TestBOInterfacePartner in the file to xmltest_PLAYBG_PortTypePartner as shown in Figure B-14.

```
Assembly Diagram: DynamicRouting | PreRoute_access_collab_XmlPortImpl.java X
* @return Service
*/
public Service locateService xmltest_PLAYBG_PortTypePartner()
{
    System.out.println(" >>> In PreRoute_access_collab_XmlPortImpl file in DynamicRoutingApp >>>");

    String METHODNAME = "locateService xmltest_PLAYBG_PortTypePartner";
    if (logger.isLoggable(Level.FINER))
    {
        logger.entering(CLASSNAME, METHODNAME);
    }
    Service result = (Service)ServiceManager.INSTANCE.locateService("xmltest_PLAYBG_PortTypePartner");

    if (logger.isLoggable(Level.FINER))
    {
        logger.exiting(CLASSNAME, METHODNAME, result);
    }
    return result;
}

DataObject result = (DataObject)this locateService_xmltest_PLAYBG_PortTypePartner().invoke("Sync", tmpres);
logger.logp(Level.FINER, CLASSNAME, METHODNAME, "After result = " + result);

// Convert DataObject result to string
logger.logp(Level.FINER, CLASSNAME, METHODNAME, "Before BusObj resultBO = new BusObj(result)");

Assembly Diagram: DynamicRouting | PreRoute_access_collab_XmlPortImpl.java X
* @return Service
*/
public Service locateService_xmltest_PLAYBG_PortTypePartner()
{
    System.out.println(" >>> In PreRoute_access_collab_XmlPortImpl file in DynamicRoutingApp >>>");

    String METHODNAME = "locateService_xmltest_PLAYBG_PortTypePartner";
    if (logger.isLoggable(Level.FINER))
    {
        logger.entering(CLASSNAME, METHODNAME);
    }
    Service result = (Service)ServiceManager.INSTANCE.locateService("xmltest_PLAYBG_PortTypePartner");

    if (logger.isLoggable(Level.FINER))
    {
        logger.exiting(CLASSNAME, METHODNAME, result);
    }
    return result;
}
```

Figure B-14 Changes in the implementation file

4. In the method "Route", you need to construct the invoked operation name as the following modification:

```
DataObject result =  
(DataObject)this.locateService_xmltest_PLAYBG_PortTypePartner().invo  
ke("Sync", tmpres);
```

Change it to:

```
DataObject result =  
(DataObject)this.locateService_xmltest_PLAYBG_PortTypePartner().invo  
ke("Sync_"+tmpres.getType().getName(), tmpres);
```

5. Modify the BOPrefix in BO MQ_Datahandler_DefaultXMLConfig. Find following code:

```
mtB0.setDefaultAttrValues();
```

Add the following code after it:

```
mtB0.setAttrValue("BOPrefix", "xmltest");
```

6. Open the AccessServlet.java file and ensure that the right IIOP port number is specified. The number 2811 needs to match the port number for the BOOTSTRAP_ADDRESS port in WebSphere Process Server.

You can look up this value for your installation in the administrative console by selecting **Application servers** → **server1** → **Ports**.

7. Specify the correct location to the XML input file for the access servlet as shown in Figure B-15.

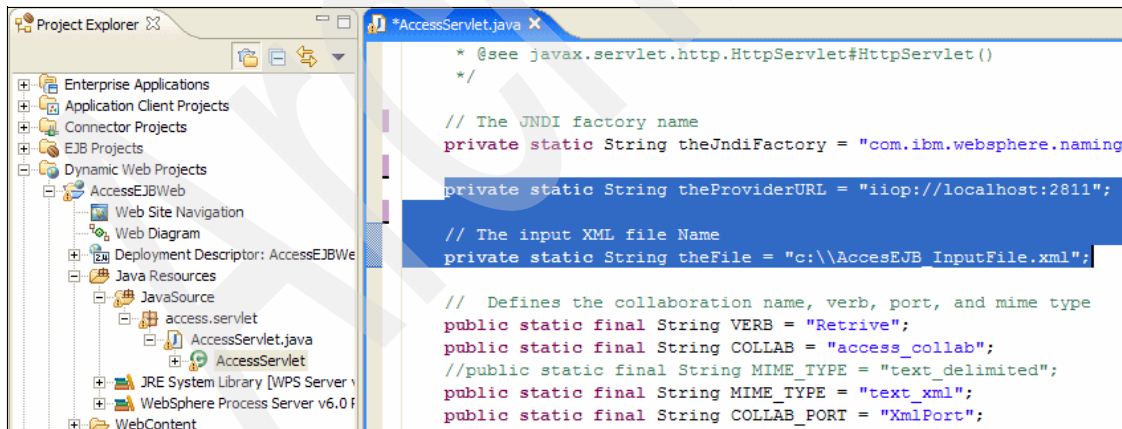


Figure B-15 Changes made in the AccessServlet.java file

Deploying the applications

The final step in the migration process is to deploy the projects to WebSphere Process Server:

1. Start the embedded instance of WebSphere Process Server.
2. Right-click the server, and select **Add and Remove Projects**.
3. In the Add and Remove Projects window, click **Add All >>** to add all of the projects in the workspace to the server. Then, click **Finish** to initiate the deployment.
4. Verify that all of the the modules are deployed and have started correctly in WebSphere Process Server.

B.2.3 Testing and verification

In this section, we explain how to run events through the migrated example in WebSphere Process Server and to verify the results.

Testing the AccessEJB example

To test the example:

1. Open the Destination Connector by using the Visual Test Connector (VTC) tool:
 - a. Click **File** → **Create/Select Profile**.
 - b. Click **File** → **New Profile**.
 - c. Click **Browse**, and locate the DestinationConnector.cfg file.
 - d. For the Connector Name, type DestinationConnector to name the profile.
 - e. For the Broker Type, select **WAS**. Click **OK**.
 - f. Select the profile that you have created, and click **OK**.
 - g. Click **File** → **Connect**.
2. As shown in Figure B-16 on page 612, load the Access Servlet client in the browser by typing the following URL with the correct port number that you are using:

`http://localhost:9080/AccessEJBWeb/AccessServlet`

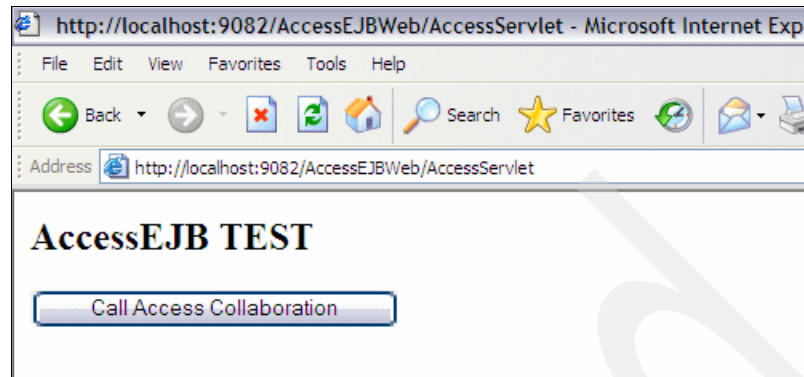


Figure B-16 Loading the AccessServlet client

3. Click **Call Access Collaboration**.

AccessServlet calls the `executeCollaborationExtended()` remote method in the `CwSession` bean and passes the data file. The session bean, in turn, calls the `DynamicRouting` module through the interface partner `RoutingPacketPartner`.

`DynamicRouter` uses `RoutingSelector` to route the call to the `access_collab` module through the interface partner `xmltest_PLAYBG_PortTypePartner`. When the EJB call reaches `access_collab_XmlPort` BPEL in the `access_collab` module, business logic is applied to the data before invoking `DestinationConnector`.

The connector receives the business object `xmltest_PLAY` in the form of `xmltest_PLAY`. Retrieve it as shown in Figure B-17.

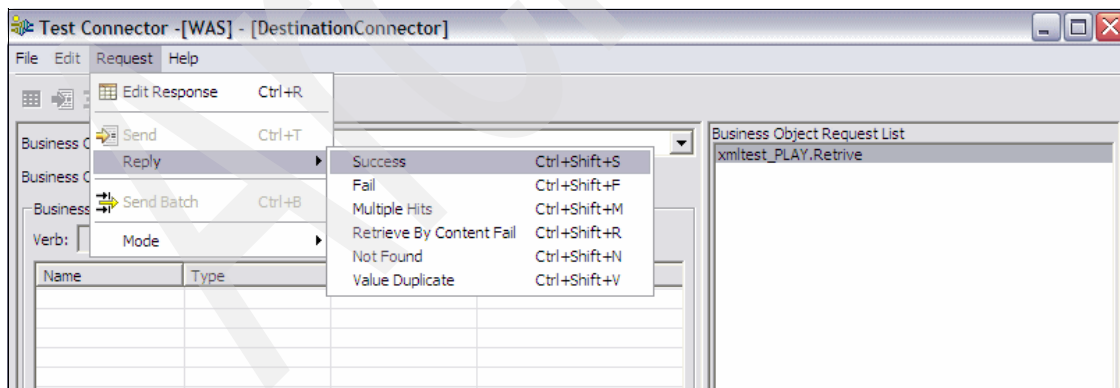


Figure B-17 Reply from the VTC

After replying `Success` from the connector, the event goes back to the `AccessServlet` as a response, as shown in Figure B-18.

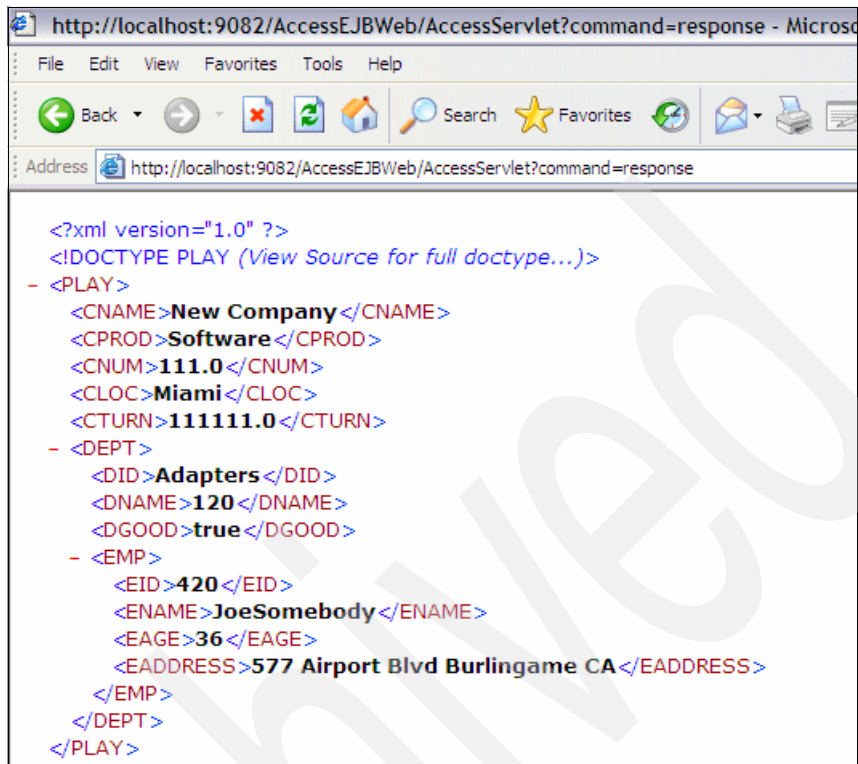


Figure B-18 Response received on AccessServlet

Figure B-19 on page 613 shows the output in the SystemOut.log file.

```
[1/29/07 15:30:45:366 PST] 00000064 SystemOut      o The inputData is : <?xml version="1.0" ?><!DOCTYPE PLAY
><PLAY><CNAME>IBM</CNAME><CPROD>Software</CPROD><CNUM>1000</CNUM><CLOC>Miami</CLOC><CTURN>12345789.00</CTURN><DEPT><DID>Adap
ers</DID><DNAME>120</DNAME><DGOOD>true</DGOOD><EMP><EID>420</EID><ENAME>JoeSomebody</ENAME><EAGE>36</EAGE><EADDRESS>577
Airport Blvd Burlingame CA</EADDRESS></EMP></DEPT></PLAY>
[1/29/07 15:30:45:366 PST] 00000064 SystemOut      o Using JNDI Context Factory:
com.ibm.websphere.naming.wsnInitialContextFactory
[1/29/07 15:30:45:366 PST] 00000064 SystemOut      o Using ProviderURL: iiop://localhost:2811
[1/29/07 15:30:45:436 PST] 00000064 SystemOut      o >>> In PreRoute_access_collab_xmlPortImpl file in DynamicRoutingApp >>>
[1/29/07 15:30:45:486 PST] 00000064 MediationImpl w com.ibm.wbiserver.mediation.MediationImplementationHandler processMessage
CWLAU0018E: The value of the PreferredInteractionStyle attribute 'async' on the interface(s) of target component
xmlPort_Input_PortType, is ignored during an invoke interaction using an interface map xmlPort_To_access_xmlPort with source
operation Sync and target operation Execute_Sync. This interaction was handled as synchronous-to-synchronous call.
[1/29/07 15:30:45:496 PST] 00000064 SystemOut      o >>> Inside the access_xmlPort BPEL process >>>
[1/29/07 15:30:45:757 PST] 00000064 SystemOut      o The attribute values of bo are:<StartHeader>
<Version = 3.0>
<EndHeader>
<StartBO:xmltest_PLAY>

    BusinessObject = xmltest_PLAY

    ....

<EndBO:xmltest_PLAY>
[1/29/07 15:30:45:867 PST] 00000064 SystemOut      o The result from collab Execution is : <?xml version="1.0"?>
<!DOCTYPE PLAY>
<PLAY><CNAME>New
Company</CNAME><CPROD>Software</CPROD><CNUM>111.0</CNUM><CLOC>Miami</CLOC><CTURN>11111.0</CTURN><DEPT><DID>Adapters</DID><D
NAME>120</DNAME><DGOOD>true</DGOOD><EMP><EID>420</EID><ENAME>JoeSomebody</ENAME><EAGE>36</EAGE><EADDRESS>577 Airport Blvd
Burlingame CA</EADDRESS></EMP></DEPT></PLAY>
[1/29/07 15:30:46:007 PST] 00000069 SystemOut      o >>> Inside the outbound_processing POJO component in
DestinationConnectorModule >>>
```

Figure B-19 Output in the SystemOut.log file

B.3 Conclusion

In this example, we showed how to migrate the WebSphere Access EJB feature. We explained how to migrate this example by using the standard migration tools.

Remember that real applications are generally more complex than the application in this simple example. Actual applications are likely to require more customization to work properly after the migration.

Migrate WBI Adapter for EJB Architecture

In this chapter, we show how to migrate WebSphere Business Integration (WBI) Adapter for Enterprise JavaBeans (EJB) Architecture to the WebSphere Process Server environment. There are two upgrade paths for WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture:

- ▶ Reusing a WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture with WebSphere Process Server environment
- ▶ Migrating a WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture to a WebSphere Process Server Enterprise Java Bean binding

We will use two samples to demonstrate these migration approaches.

In the first example, we explain how to reuse a WebSphere Business Integration Adapter that was running on WebSphere InterChange Server with the WebSphere Process Server environment. The specific WebSphere Business Integration Adapter that is used in this example is a WebSphere Business Integration Adapter for Enterprise Java Beans Architecture, but any kind of WebSphere Business Integration Adapter will work. The standard migration tools help generate the required intermediate components that make WebSphere Process Server able to use any WebSphere Business Integration Adapter.

In the second example, we show how to migrate a WebSphere Business Integration Adapter for Enterprise Java Beans Architecture to an Enterprise Java Bean (EJB) binding on WebSphere Process Server. In this example, we show how to set up the EJB binding in WebSphere Process Server and implement the generated skeleton Java component.

The standard migration tools are one option among others to migrate WebSphere InterChange Server applications to WebSphere Process Server. See Part 2, “Migration implementation concepts” on page 71, for an overall discussion about the possible options for migration implementation. Part 3, “Migration tooling” on page 177, provides detailed information about the standard migration tools.

We present two simple examples in this chapter to illustrate the complete upgrade by using the standard migration tools. In reality, complex WebSphere InterChange Server solutions generally require changes before or after migrating. You must make modifications to either the WebSphere InterChange Server artifacts before migration or to the generated WebSphere Process Server artifacts after migration.

This appendix includes the following topics:

- ▶ C.1, “Target environment” on page 617
- ▶ C.2, “Running the WebSphere Business Integration Adapter with WebSphere Process Server: EJB” on page 617
- ▶ C.3, “Migrating to an EJB binding” on page 631
- ▶ C.4, “Conclusion” on page 648

C.1 Target environment

Table C-1 identifies the software that we used in the target environment.

Table C-1 Software versions used

Software	Version
WebSphere Process Server	6.2
WebSphere Integration Developer	6.2
WebSphere InterChange Server	4.3.0 Fix Pack 6
WebSphere Business Integration Adapter Framework	2.6.0.12
WebSphere Business Integration Adapter for Enterprise Java Beans Architecture	1.2

C.2 Running the WebSphere Business Integration Adapter with WebSphere Process Server: EJB

In this section, we explain how to reuse a WebSphere Business Integration Adapter for Enterprise Java Beans Architecture that was running on WebSphere InterChange Server with the WebSphere Process Server environment (Figure C-1).

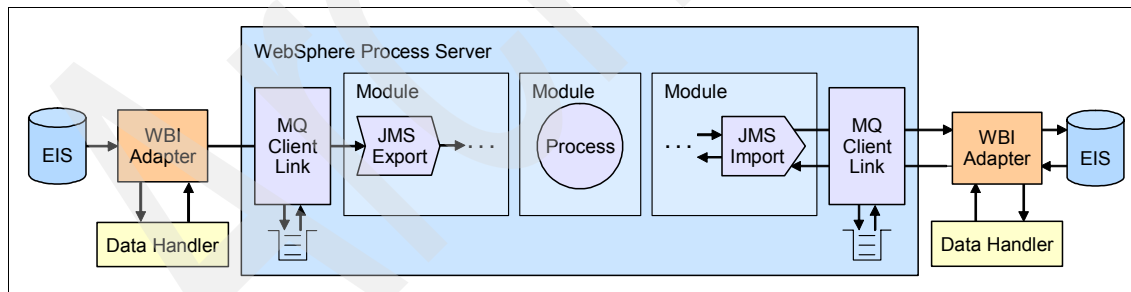


Figure C-1 WebSphere Business Integration Adapters with WebSphere Process Server

This landscape keeps the WebSphere Business Integration Adapter setup and all Enterprise Information System (EIS) connectivity as is. The project is upgraded to run in the WebSphere Process Server environment.

C.2.1 Premigration overview

In this section, we provide a high-level overview of the premigrated WebSphere InterChange Server project. This information serves as a useful cross-reference to the original artifacts when exploring the new Service Component Architecture (SCA) components that are generated by the server migration tool. You can also import the integrated component library (the `EJBSample.jar` file) that is available in the additional materials for this book (refer to Appendix D, “Additional material” on page 651) into WebSphere InterChange Server System Manager. This example focuses on an outbound operation. As illustrated in Figure C-2, a business object is sent from the dummy `BIA_PortConnector` to the `InvokeWASEJB` collaboration object. The collaboration synchronously invokes an Enterprise JavaBean (EJB) by using the `BIA_EJBConnector`. The EJB is deployed on a WebSphere Application Server V5.1 server.

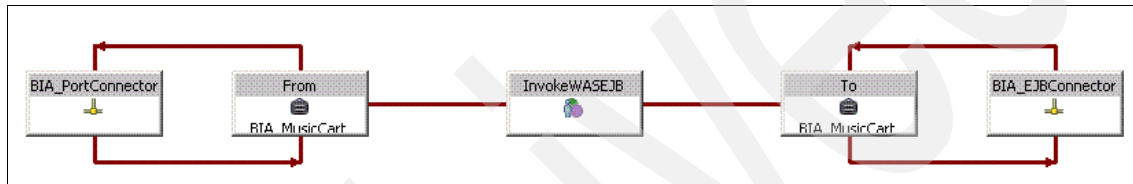


Figure C-2 Prebuilt inbound example on WebSphere InterChange Server

C.2.2 Operational model

The operational model in Figure C-3 on page 619 illustrates the WebSphere Process Server artifacts that are created by the server migration tool. The server migration assumes that the adapter is running with WebSphere Process Server in compatibility mode and creates Java Message Service (JMS) import bindings that enable communication with WebSphere Business Integration Adapter for Enterprise Java Beans Architecture.

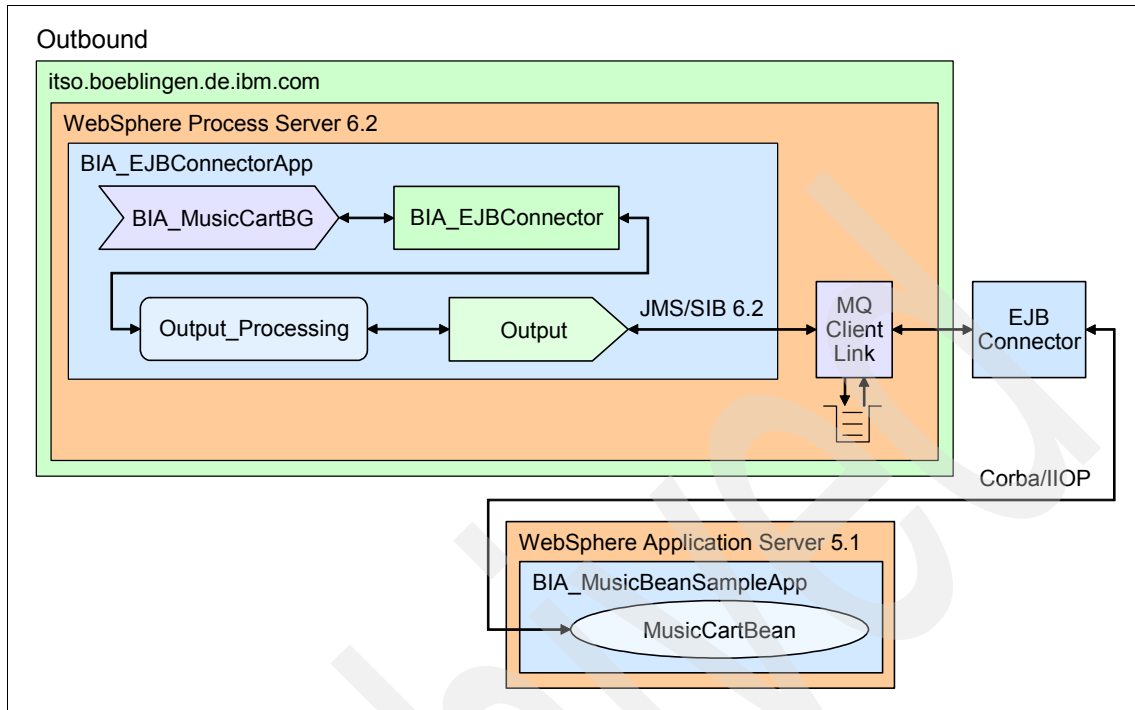


Figure C-3 WebSphere Business Integration Adapter with WebSphere Process Server: EJB

In this configuration, the original WebSphere Business Integration Adapter remains in place and operational. The JMS import binding, Output, is invoked with a Service Data Object (SDO) that originates from the business process. The JMS import binding converts the SDO to an XML representation of a WebSphere Business Integration business object and delivers it to the request queue destination of the Service Integration Bus. The WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture receives the business object and invokes the remote EJB. The EJB response is received by the adapter, which converts it to a WebSphere Business Integration business object before delivering it to the response queue destination of the Service Integration Bus.

C.2.3 Development

In this section, we provide the detailed steps to migrate the WebSphere Business Integration Adapter for Enterprise Java Beans Architecture project to WebSphere Process Server.

Files for download: The samples in this section are available for download as explained in Appendix D, “Additional material” on page 651.

Follow these steps to migrate the WebSphere Business Integration Adapter for Enterprise Java Beans Architecture project to WebSphere Process Server:

1. Start WebSphere Integration Developer in a new workspace. In the Workspace Launcher window (Figure C-4), for Workspace, type D:\EJBsample_Migration, and click **OK**.

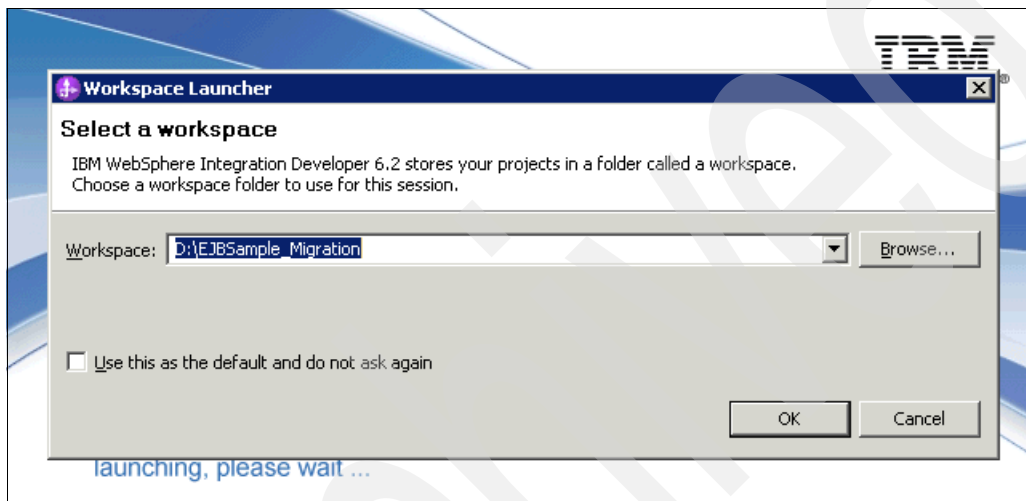


Figure C-4 Creating a new workspace

2. Close the Welcome view.

3. In the Business Integration perspective window (Figure C-5), select **File** → **Import**.

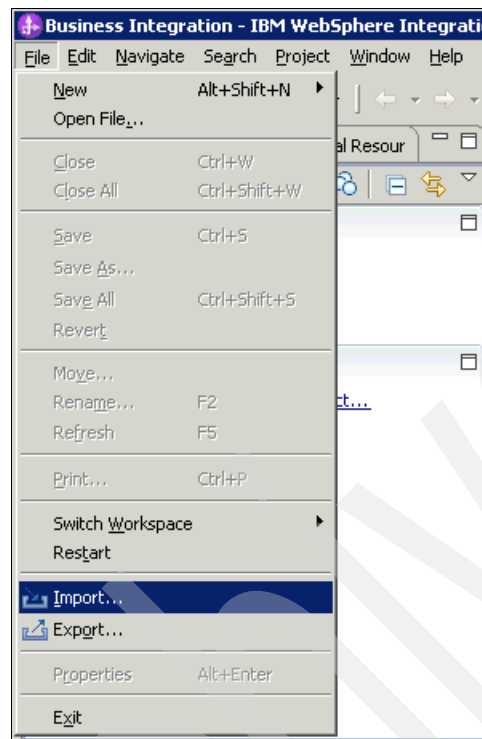


Figure C-5 Selecting the Import option

4. In the Import: Select window (Figure C-6), select **WebSphere InterChange Server Repository**, and click **Next**.

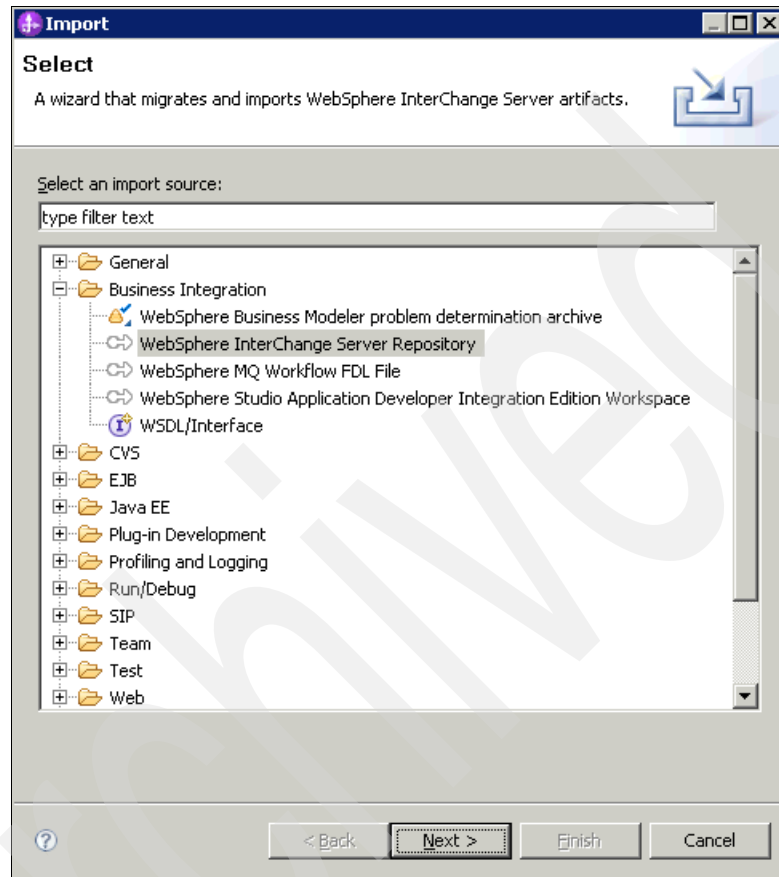


Figure C-6 Importing the WebSphere InterChange Server Repository file

5. In the WebSphere InterChange Server Import Wizard: Select WebSphere InterChange Server Repository Detail window(Figure C-7):
 - a. For WebSphere InterChange Server repository path, click **Browse JARs**. Specify the file path of the ICS repository. Type
D:\RedbookScenarios\EJBsample.jar.
 - b. For the WebSphere Integration Developer library name, type
EJBsample_Lib.
 - c. Click **Next**.

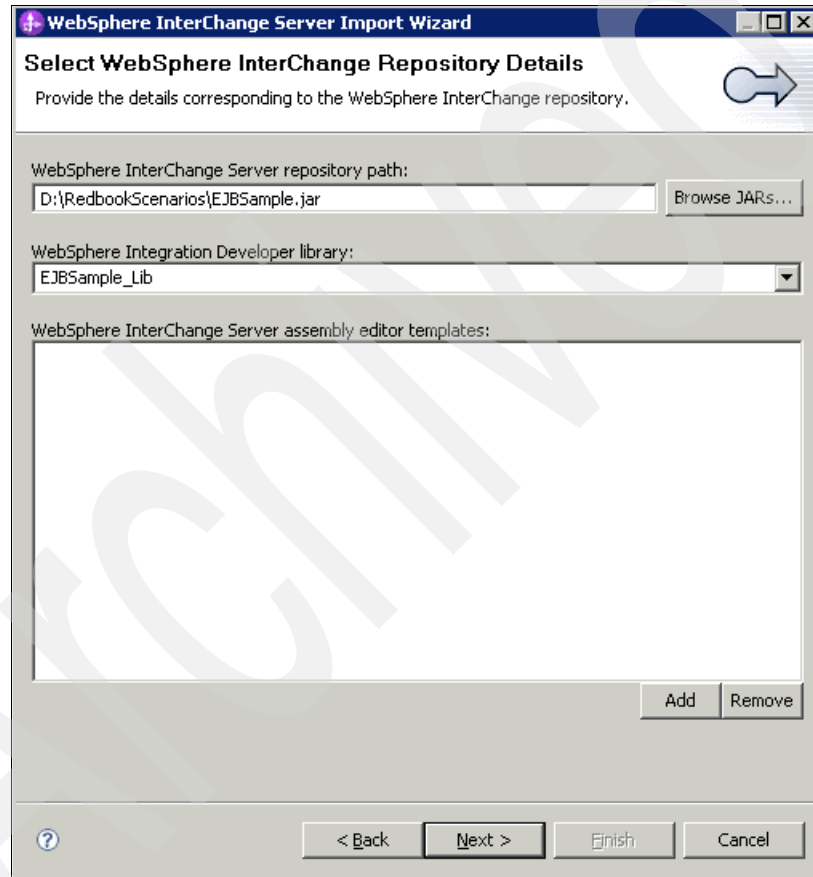


Figure C-7 Select WebSphere InterChange Server Repository Detail

6. In the WebSphere InterChange Server Import Wizard: Configure Connector Migration window (Figure C-8):
 - a. Keep the default configuration for BIA_PortConnector.
 - b. Click **BIA_EJBConnector**.
 - c. Select the **JMS to EJB WBI Adapter** binding for BIA_EJBConnector.

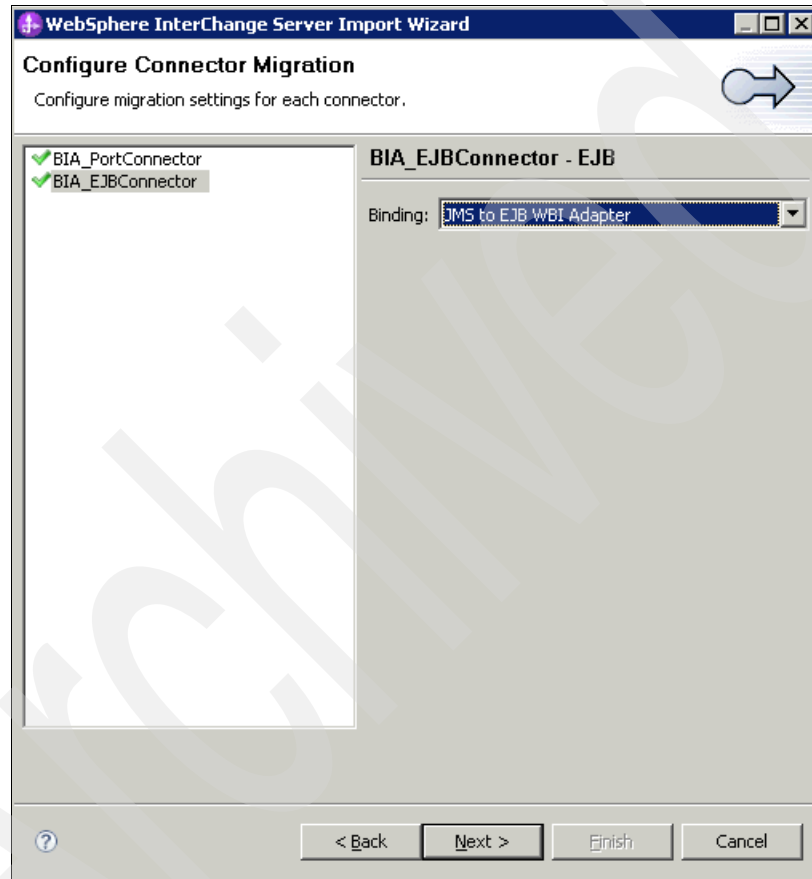


Figure C-8 Configure Connector Migration

7. In the WebSphere InterChange Server Import Wizard: Conversion Options window (Figure C-9), select the three recommended options, and click **Next**.

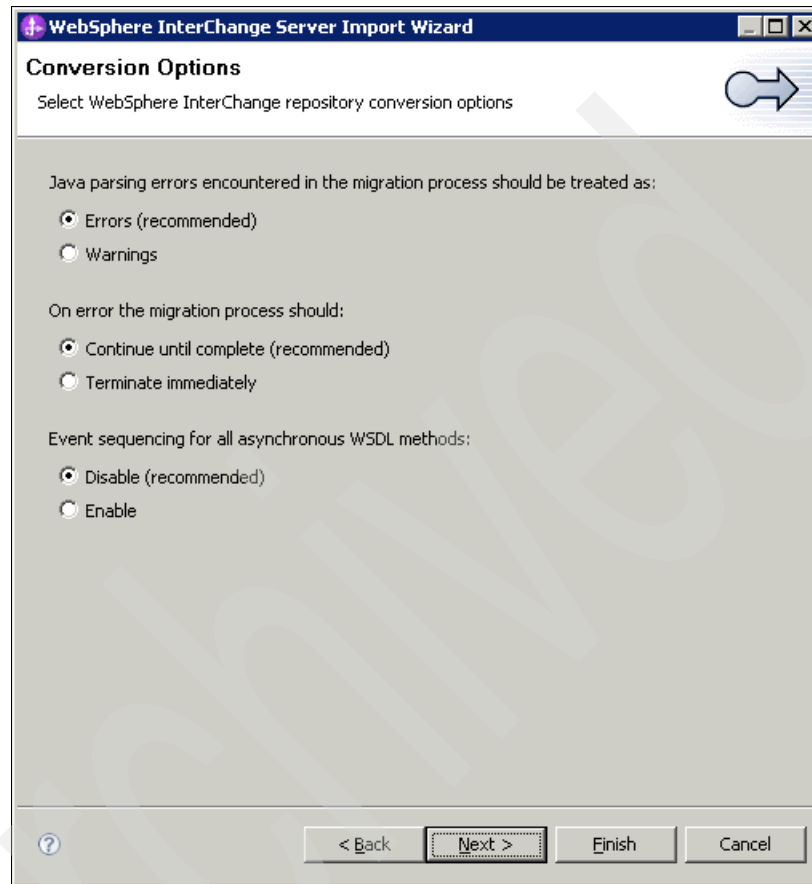


Figure C-9 Conversion Options

8. In the WebSphere InterChange Server Import Wizard: Migration Summary window (Figure C-10 on page 626), click **Finish**.

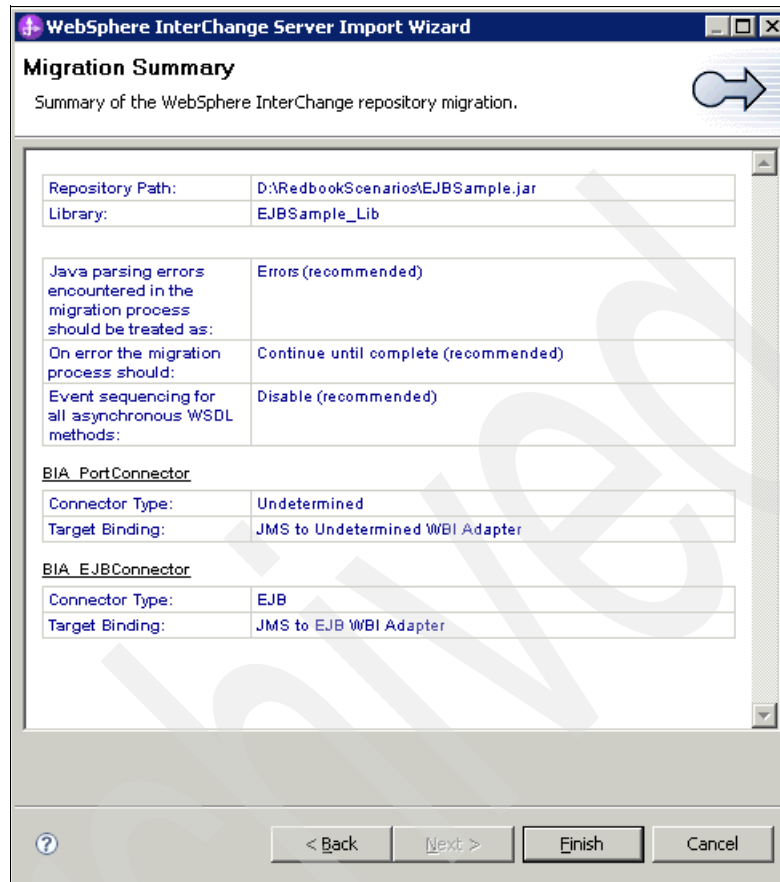


Figure C-10 Migration Summary

9. The Migration Results windows shows the result. Click **Close**.

You have now completed the migration of the outbound EJB example to WebSphere Process Server. The four migrated projects are in the WebSphere Integration Developer Business Integration Perspective (Figure C-11 on page 627).

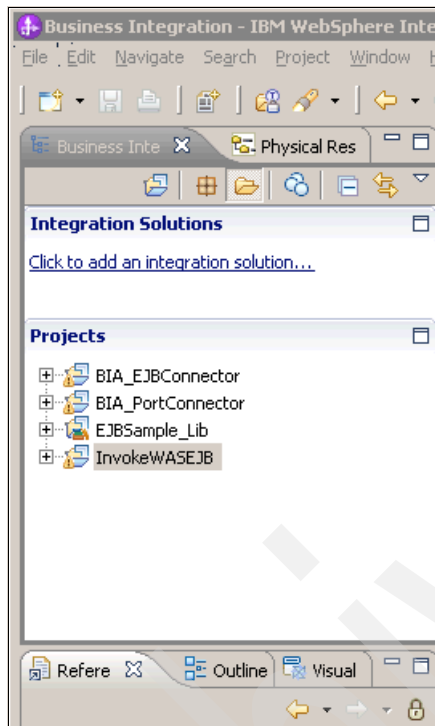


Figure C-11 Migrated projects

C.2.4 Modifying the adapter configuration files

In this section, we explain how to modify the stand-alone WebSphere Business Integration Adapter for Enterprise Java Beans Architecture, which interacts with WebSphere Process Server:

1. By using the Connector Configurator tool, open the `BIA_EJBConnector.cfg` file from the `C:\IBM\WebSphereAdapters\connectors\BIA_EJB` folder.

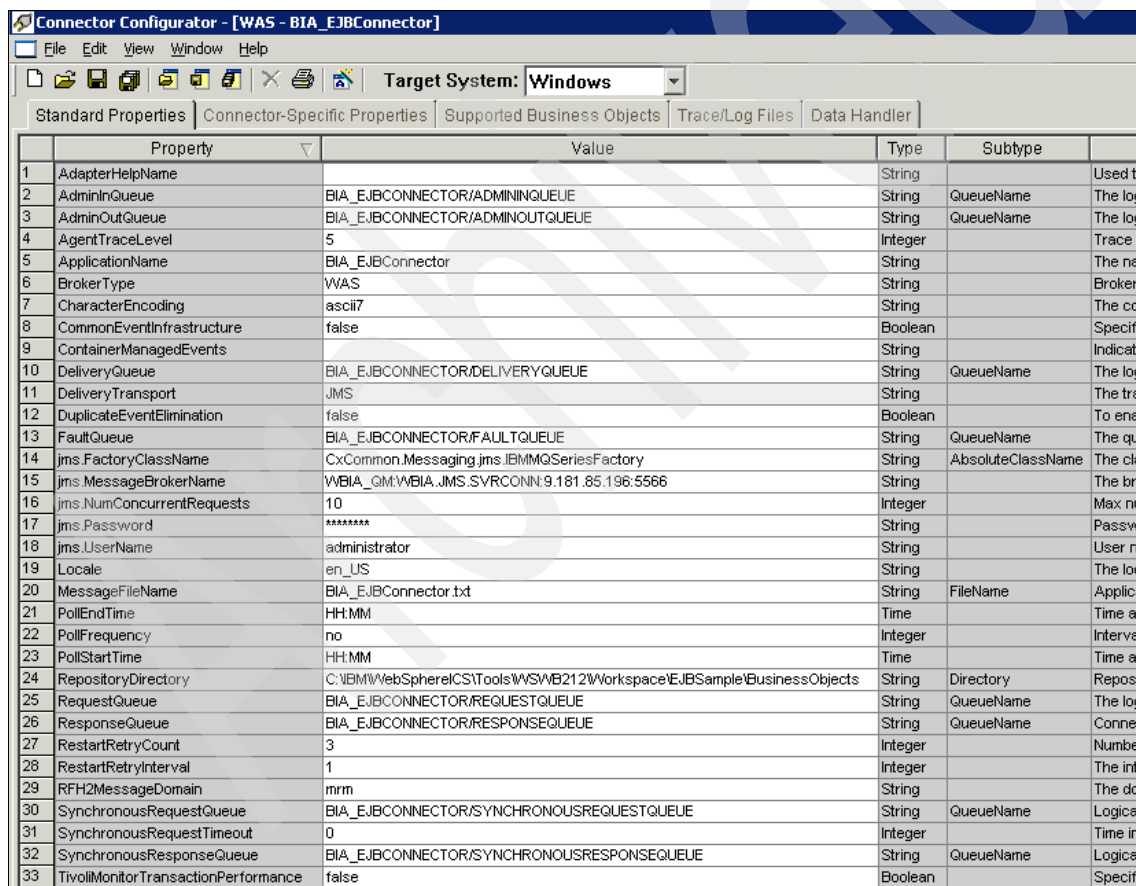
The path `C:\IBM\WebSphereAdapters\connectors\BIA_EJB` is where you stored the connector configuration file for `BIA_EJBConnector`.

2. Change the following properties as shown in Figure C-12 on page 628:
 - a. For the `BrokerType` field, change the value to `WAS`.
 - b. In the `jms.MessageBrokerName` field, type `WBIA_QM:WBIA.JMS.SVRCONN:9.181.85.196:5566`.

The `9.181.85.196` matches the IP address of the WebSphere Process Server.

The number 5566 needs to match the port number for the SIB_MQ_ENDPOINT_ADDRESS port in WebSphere Process Server. You can look up this value for your installation in the Integrated Solutions Console by selecting **Application servers** → **server1** → **Ports**.

- c. In the `jms.Username` field, type any value. This sample runs without security enabled, and the value in this field does not matter, as long as the field is not empty.
- d. For the `RepositoryDirectory`, change the path to the location of your business objects, for example:
`C:\IBM\WebSphereICS\Tools\WSWB212\Workspace\EJBSample\BusinessObjects`
- e. Save the changes.



Connector Configurator - [WAS - BIA_EJBConnector]					
File Edit View Window Help					
Target System: Windows					
Standard Properties Connector-Specific Properties Supported Business Objects Trace/Log Files Data Handler					
	Property	Value	Type	Subtype	Used to
1	AdapterHelpName		String		Used to
2	AdminInQueue	BIA_EJBCONNECTOR/ADMININQUEUE	String	QueueName	The logi
3	AdminOutQueue	BIA_EJBCONNECTOR/ADMINOUTQUEUE	String	QueueName	The logi
4	AgentTraceLevel	5	Integer		Trace le
5	ApplicationName	BIA_EJBConnector	String		The nam
6	BrokerType	WAS	String		Broker t
7	CharacterEncoding	ascii7	String		The con
8	CommonEventInfrastructure	false	Boolean		Specify
9	ContainerManagedEvents		String		Indicates
10	DeliveryQueue	BIA_EJBCONNECTOR/DELIVERYQUEUE	String	QueueName	The logi
11	DeliveryTransport	JMS	String		The tran
12	DuplicateEventElimination	false	Boolean		To enab
13	FaultQueue	BIA_EJBCONNECTOR/FAULTQUEUE	String	QueueName	The que
14	jms.FactoryClassName	CxCommon.Messaging.Jms.IBMMQSeriesFactory	String	AbsoluteClassName	The clas
15	jms.MessageBrokerName	WBIA_QM:WBIA.JMS.SVRCONN:9.181.85.196:5566	String		The brok
16	jms.NumConcurrentRequests	10	Integer		Max num
17	jms.Password	*****	String		Passw
18	jms.UserName	administrator	String		User na
19	Locale	en_US	String		The loca
20	MessageFileName	BIA_EJBConnector.txt	String	FileName	Applicat
21	PollEndTime	HH:MM	Time		Time at
22	PollFrequency	no	Integer		Interval
23	PollStartTime	HH:MM	Time		Time at
24	RepositoryDirectory	C:\IBM\WebSphereICS\Tools\WSWB212\Workspace\EJBSample\BusinessObjects	String	Directory	Reposit
25	RequestQueue	BIA_EJBCONNECTOR/REQUESTQUEUE	String	QueueName	The logi
26	ResponseQueue	BIA_EJBCONNECTOR/RESPONSEQUEUE	String	QueueName	Connect
27	RestartRetryCount	3	Integer		Number
28	RestartRetryInterval	1	Integer		The inte
29	RFH2MessageDomain	mrm	String		The dom
30	SynchronousRequestQueue	BIA_EJBCONNECTOR/SYNCHRONOUSREQUESTQUEUE	String	QueueName	Logical
31	SynchronousRequestTimeout	0	Integer		Time in
32	SynchronousResponseQueue	BIA_EJBCONNECTOR/SYNCHRONOUSRESPONSEQUEUE	String	QueueName	Logical
33	TivoliMonitorTransactionPerformance	false	Boolean		Specify

Figure C-12 Modifications on the connector configuration file

3. Open the `BIA_PortConnector.cfg` file and repeat the previous two steps on page 627.

C.2.5 Testing and verification

In this section, we explain how to run events through the migrated example in WebSphere Process Server and verify the results:

1. Verify that the WebSphere Process Server is started.
2. Export the application enterprise archive (EAR) files from WebSphere Integration Developer and deploy them in WebSphere Process Server:
 - a. Click **File** → **Export**.
 - b. Select the EAR file and click **Next**.
 - c. In the next window, for EAR Project, type `BIA_EJBConnectorApp` and specify the destination. The name of the EAR project is the name of the project as it is displayed in the Business Integration Perspective with App appended to it. Click **Finish** to generate the file.
 - d. Repeat step 1 to step 3 for the `BIA_PortConnectorApp` and `InvokeWASEJBApp` projects.
 - e. Install all three applications by using the Integrated Solutions Console.
 - f. Check the WebSphere Process Server SystemOut log file, and make sure that the applications start successfully without any errors.
3. Start the WebSphere Business Integration Adapter for Enterprise Java Beans Architecture with the modified `BIA_EJBConnector.cfg` file on page 627.
4. Start the Visual Test Connector with the modified `BIA_PortConnector.cfg` file on page 629.

5. Load the BIA_SampleMusicCartB0.bo into the Visual Test Connector, and click **Send Business Object** (Figure C-13).

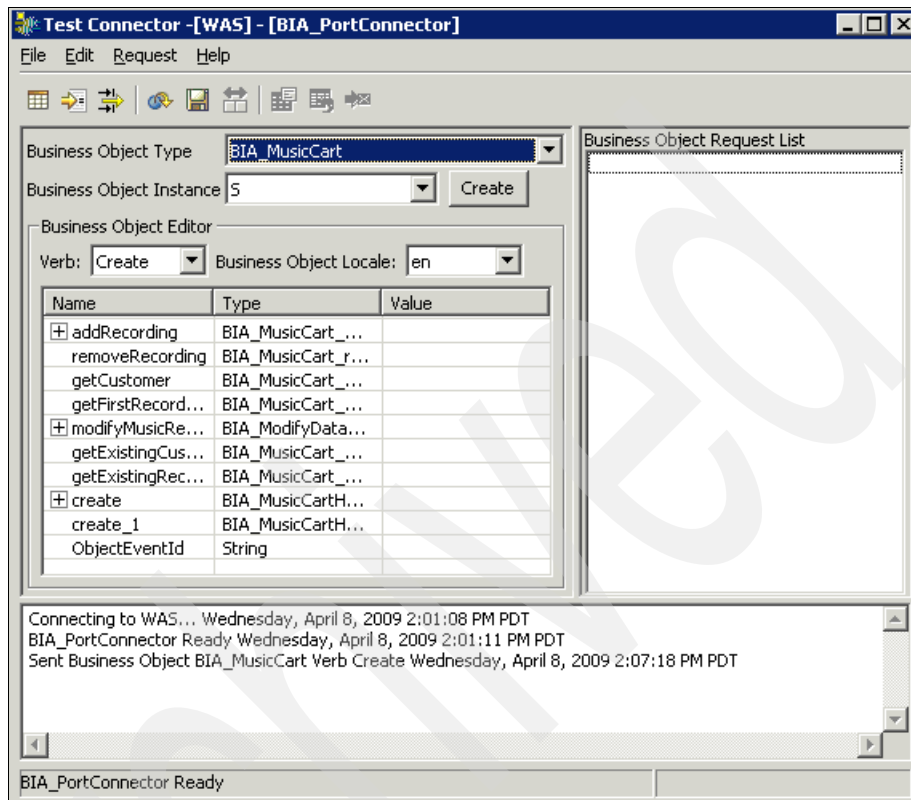


Figure C-13 Visual Test Connector to send the BIA_SampleMusicCartBO.bo

You see a successful log of the event in Example C-1 in the SystemOut.log file under *WPS Install\profile\profile name\logs\server1*.

Example C-1 Expected result in SystemOut.log

```
[4/8/09 14:07:41:641 PDT] fd2d5c2 SystemOut    0 Response is
***** BusinessObject = BIA_Music
Verb = Create
AttributeCount = 2
TLO = 1
BusinessObject = BIA_TLO_Music
Verb =
AttributeCount = 3
MusicInfo = 1
BusinessObject = BIA_Music_MusicInfo
```

```

Verb =
AttributeCount = 5
Title =MODIFIED_Brand New Day
Composer =MODIFIED_Sting
Publisher =MODIFIED_PolyGram Records
Year = 1978
ObjectEventId = CxCommon.BusinessObject@1674077128
PurchaseOrder = 1
BusinessObject = BIA_Music_PurchaseOrder
Verb =
AttributeCount = 2
Item = 1
BusinessObject = BIA_Music_Item
Verb =
AttributeCount = 4
RecordTitle = John Denver Hits
Quantity = 20
UnitPrice = 10
ObjectEventId = CxCommon.BusinessObject@1677878274
ObjectEventId = CxCommon.BusinessObject@1676698608
ObjectEventId = CxCommon.BusinessObject@1672897462
ObjectEventId = CxCommon.BusinessObject@1671717796

```

C.3 Migrating to an EJB binding

In this section, we explain how to migrate a WebSphere Business Integration Adapter for Enterprise Java Beans Architecture to an Enterprise JavaBeans (EJB) binding on WebSphere Process Server. The runtime environment for this example is contained within a single WebSphere Process Server instance. Figure C-14 on page 631 illustrates the target system topology.

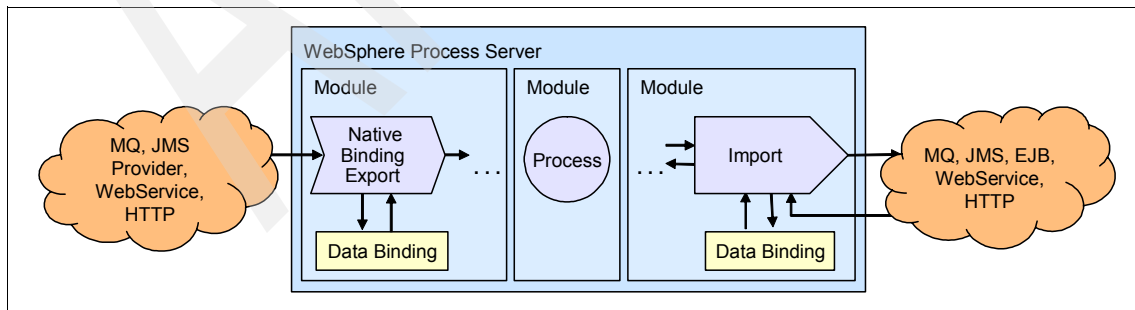


Figure C-14 WebSphere Process Server binding interacting directly with a transport or protocol

Important: The EJB binding only supports the outbound usage pattern. And, *only* the WebSphere Business Integration Adapter for Enterprise Java Beans Architecture that communicates with the Stateless Session Bean can be migrated to a WebSphere Process Server Enterprise Java Bean binding.

C.3.1 Premigration overview

In this section, we provide a high-level overview of the premigrated WebSphere InterChange Server project. This information serves as a useful cross-reference to the original artifacts when exploring the new Service Component Architecture (SCA) components that are generated by the server migration tool. The integrated component library (the ICSEJBScenario.jar file) that is available in the additional materials for this book (refer to Appendix D, “Additional material” on page 651) can also be imported into WebSphere InterChange Server System Manager.

This example focuses on an outbound operation. As illustrated in Figure C-15, a business object is sent from the PortConnector to CustomerRetrieve collaboration object. The collaboration synchronously invokes an Enterprise Java Bean (EJB) by using the EJBConnector. The EJB is deployed on a WebSphere Process Server V6.2 server.



Figure C-15 CustomerRetrieve collaboration object for outbound processing

C.3.2 Operational model

The operational model in Figure C-16 illustrates the outbound system configuration, when migrating WebSphere Business Integration Adapter for Enterprise Java Beans Architecture to a WebSphere Process Server Enterprise Java Bean binding.

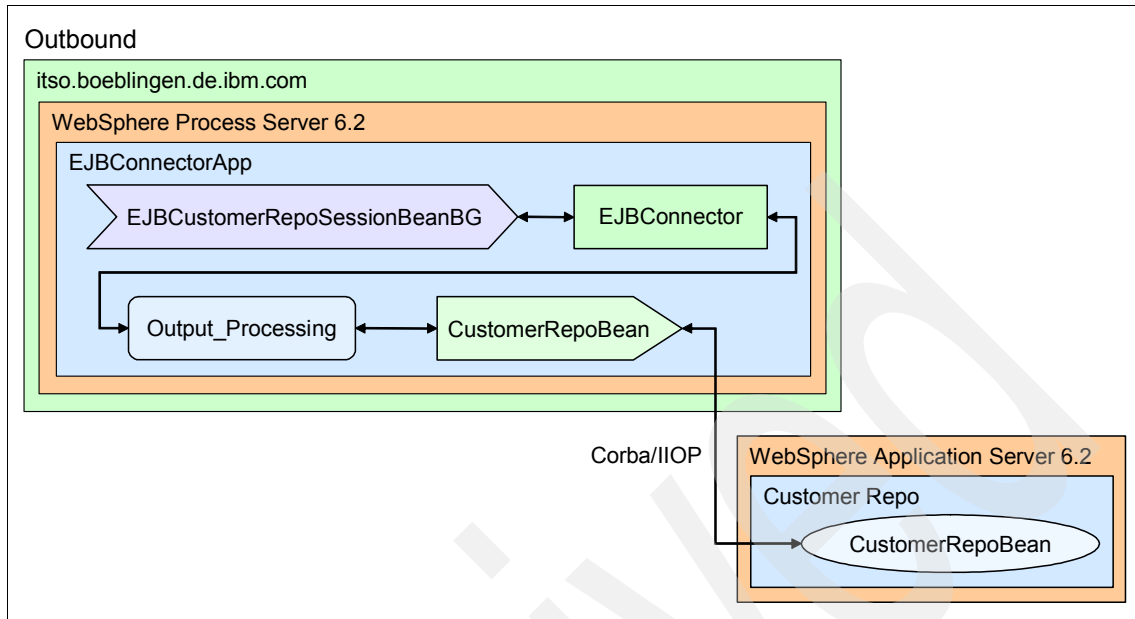


Figure C-16 Operational model for the EJB binding example

In this configuration, the original WebSphere Business Integration Adapter is migrated to an EJB import binding named `CustomerRepoBean`. An invocation that initiated from a business process goes through a series of conversions and ultimately arrives at the Output Java component. The Output Java component is a generated skeleton. You need to implement the logic to convert SDO into a Java object `CustomerInformation`, which is the expected argument of the remote EJB service `CustomerRepoBean`. The EJB import invokes the remote EJB and returns the response back to the business process. The Output Java component will reverse the `CustomerInformation` to SDO along the response path.

C.3.3 Development

In this section, we provide the detailed steps to migrate the WebSphere Business Integration Adapter for Enterprise Java Beans Architecture project to WebSphere Process Server Enterprise Java Bean binding.

Files for download: The samples in this section are available for download as explained in Appendix D, “Additional material” on page 651.

Follow these steps:

1. Start WebSphere Integration Developer in a new workspace. In the Workspace Launcher window (Figure C-17), for Workspace, type D:\ICSEJBScenario_Migration, and click **OK**.

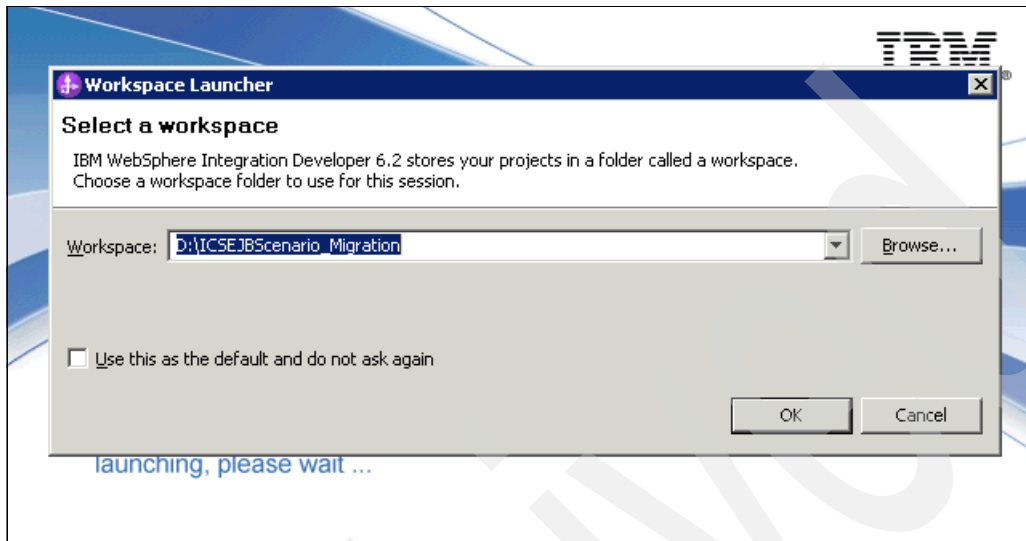


Figure C-17 Creating a new workspace

2. Close the Welcome view.
3. In the Business Integration perspective window, select **File** → **Import**.
4. In the Import: Select window, select **WebSphere InterChange Server Repository**, and click **Next**.
5. In the WebSphere InterChange Server Import Wizard: Select WebSphere InterChange Repository Details window (Figure C-18), perform these steps:
 - a. For the WebSphere InterChange Server repository path, click **Browse**. Specify the file path of the ICS repository. Type:
D:\RedbookScenarios\ICSEJBScenario.jar
 - b. For the WebSphere Integration Developer library name field, type
ICSEJBScenario_Lib
 - c. Click **Next**.

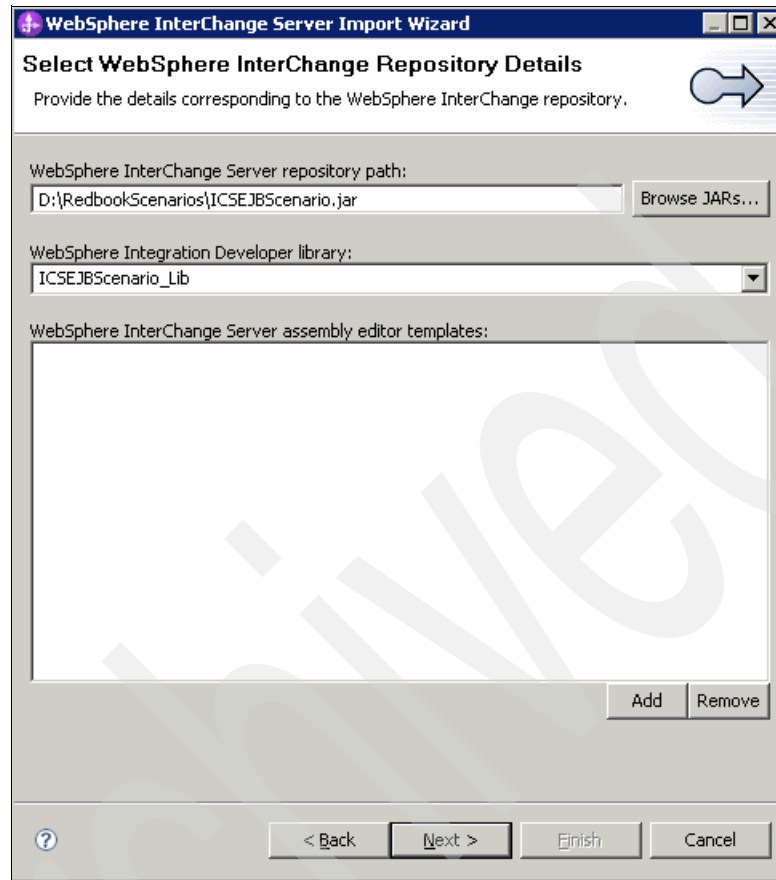


Figure C-18 Select WebSphere InterChange Repository Details

6. In the WebSphere InterChange Server Import Wizard: Configure Connector Migration window (Figure C-19), perform these steps:
 - a. Keep the default configuration for PortConnector.
 - b. Click **EJBConnector**.
 - c. Select the **EJB Binding** for the EJBConnector.
 - d. Click **Add**, and enter D:\RedbookScenarios\CustomerRepoEJB.jar in the Select EJB JARs field.

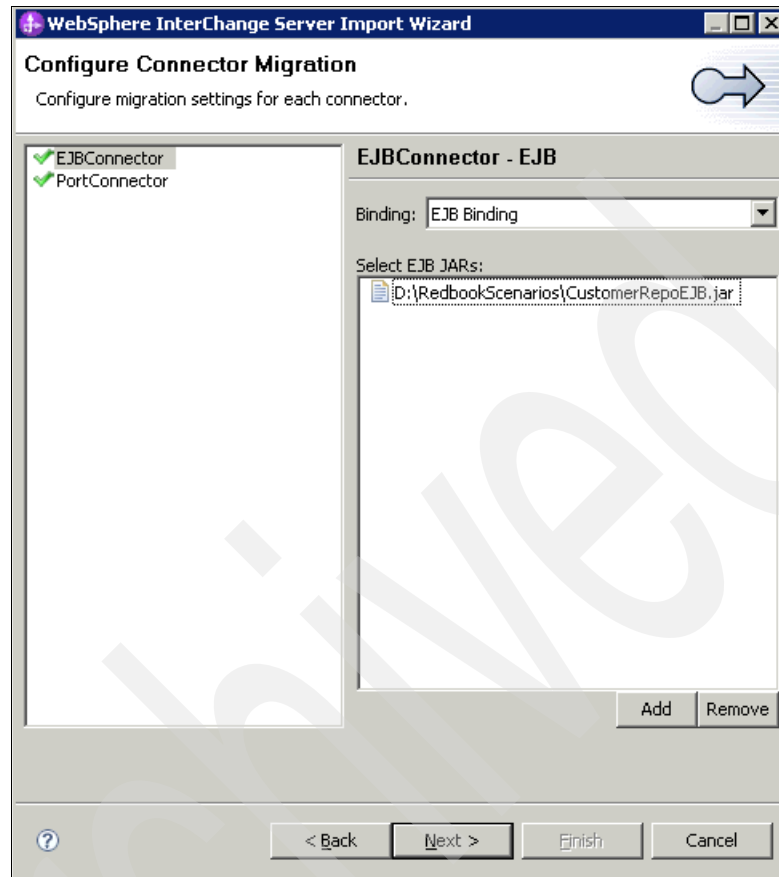


Figure C-19 Configure Connector Migration

7. In the WebSphere InterChange Server Import Wizard: Conversion Options window, select the three recommended values, and click **Next**.
8. In the WebSphere InterChange Server Import Wizard: Migration Summary window (Figure C-20), click **Finish**.

WebSphere InterChange Server Import Wizard

Migration Summary

Summary of the WebSphere InterChange repository migration.

Repository Path:	D:\RedbookScenarios\ICSEJBScenario.jar
Library:	ICSEJBScenario_Lib [new]

Java parsing errors encountered in the migration process should be treated as:	Errors (recommended)
On error the migration process should:	Continue until complete (recommended)
Event sequencing for all asynchronous WSDL methods:	Disable (recommended)

EJBConnector

Connector Type:	EJB
Target Binding:	EJB Binding
EJB Jars:	D:\RedbookScenarios\CustomerRepoEJB.jar

PortConnector

Connector Type:	Undetermined
Target Binding:	JMS to Undetermined WBI Adapter

? < Back Next > Finish Cancel

Figure C-20 Migration Summary

9. The Migration Results window shows the result (Figure C-21). Click **Close**.

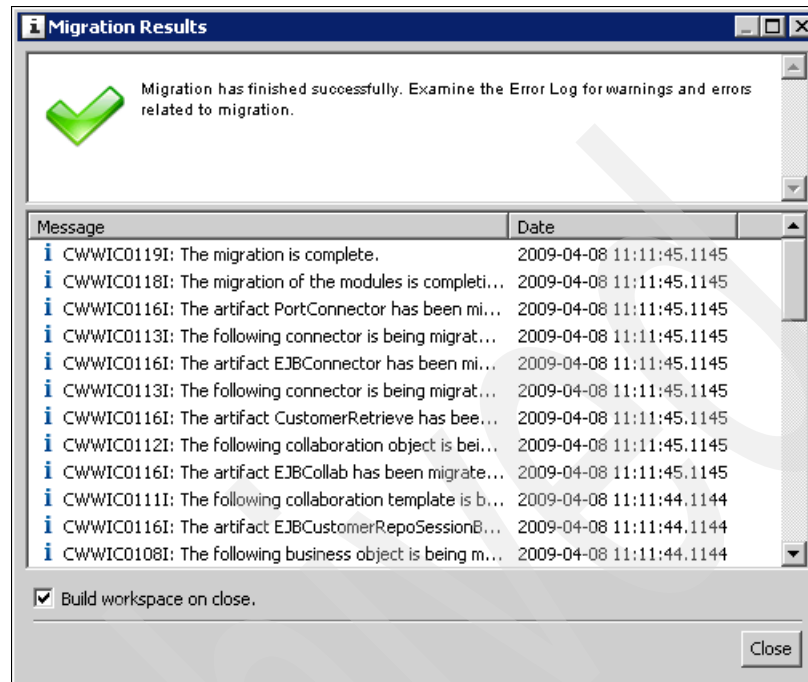


Figure C-21 Migration Results

You have now completed the migration of the outbound EJB example to WebSphere Process Server Enterprise Java Bean binding. You can find four projects in the WebSphere Integration Developer Business Integration Prospective (Figure C-22).

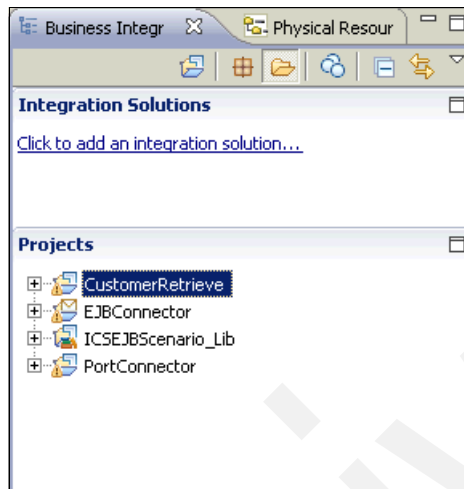


Figure C-22 Migrated projects

C.3.4 Implement the Output skeleton Java component

In this section, we will provide sample code for the Implementation of the Output skeleton Java component. Follow these steps:

1. Open the Assembly Diagram for the EJBConnector project (Figure C-23).

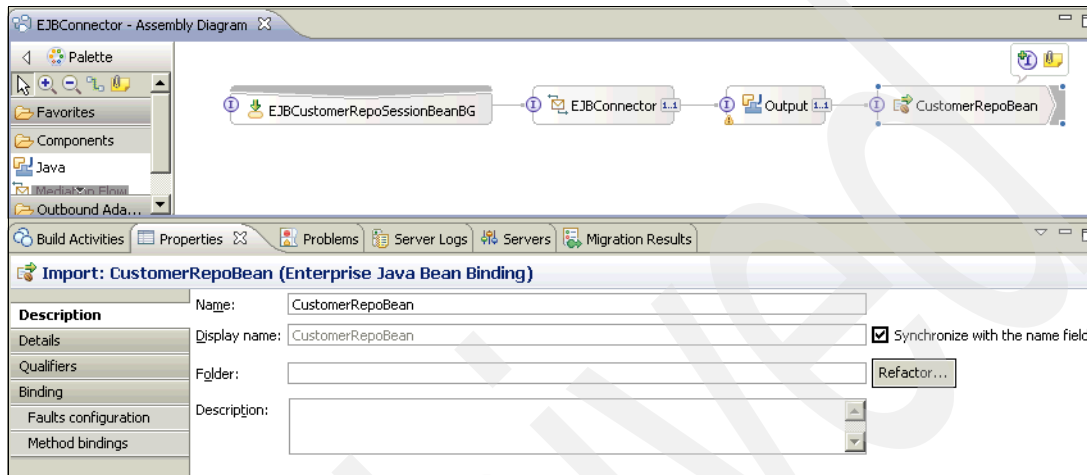
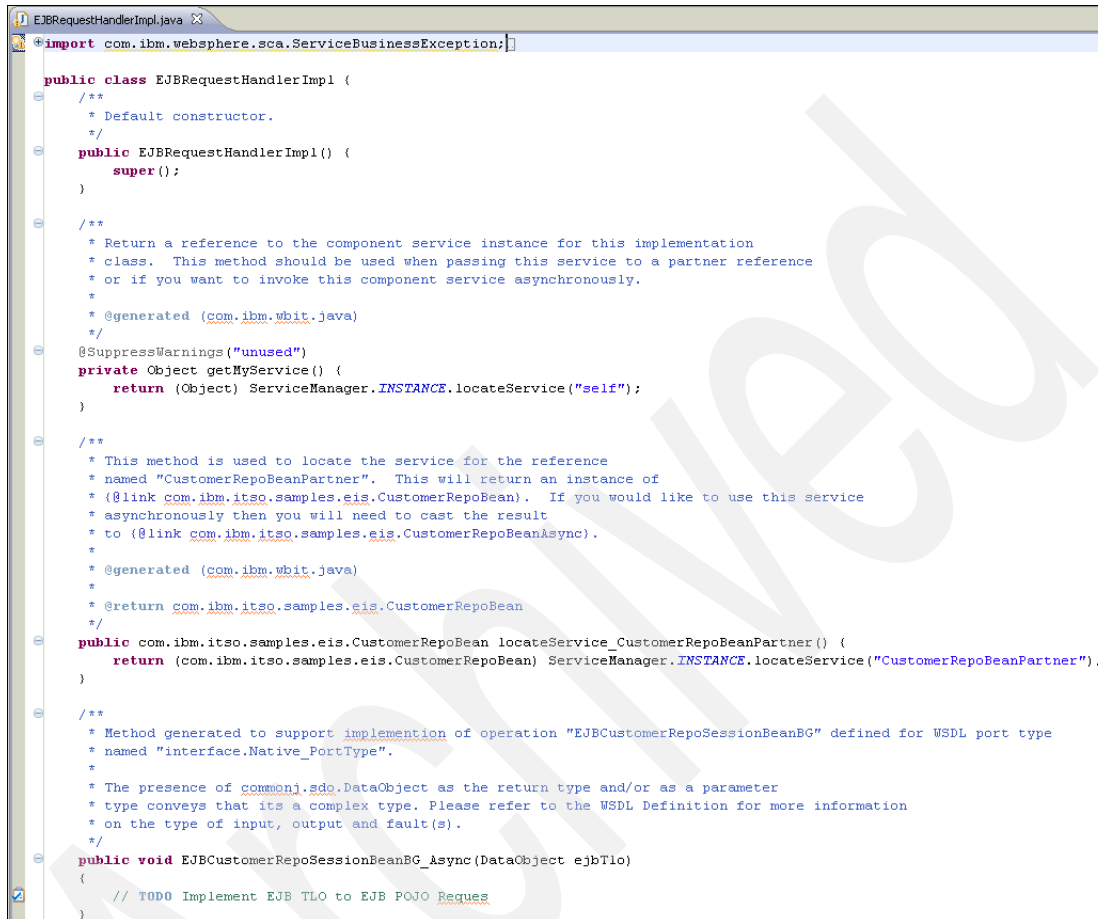


Figure C-23 Enterprise JavaBean Binding

2. Double-click the Output component, and the generated skeleton Java file opens (Figure C-24).



```
1 EJBRequestHandlerImpl.java
import com.ibm.websphere.sca.ServiceBusinessException;

public class EJBRequestHandlerImpl {
    /**
     * Default constructor.
     */
    public EJBRequestHandlerImpl() {
        super();
    }

    /**
     * Return a reference to the component service instance for this implementation
     * class. This method should be used when passing this service to a partner reference
     * or if you want to invoke this component service asynchronously.
     *
     * @generated (com.ibm.wbi.java)
     */
    @SuppressWarnings("unused")
    private Object getMyService() {
        return (Object) ServiceManager.INSTANCE.locateService("self");
    }

    /**
     * This method is used to locate the service for the reference
     * named "CustomerRepoBeanPartner". This will return an instance of
     * (@link com.ibm.itso.samples.eis.CustomerRepoBean). If you would like to use this service
     * asynchronously then you will need to cast the result
     * to (@link com.ibm.itso.samples.eis.CustomerRepoBeanAsync).
     *
     * @generated (com.ibm.wbi.java)
     *
     * @return com.ibm.itso.samples.eis.CustomerRepoBean
     */
    public com.ibm.itso.samples.eis.CustomerRepoBean locateService_CustomerRepoBeanPartner() {
        return (com.ibm.itso.samples.eis.CustomerRepoBean) ServiceManager.INSTANCE.locateService("CustomerRepoBeanPartner");
    }

    /**
     * Method generated to support implementation of operation "EJBCustomerRepoSessionBeanBG" defined for WSDL port type
     * named "interface.Native_PortType".
     *
     * 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 void EJBCustomerRepoSessionBeanBG_Async(DataObject ejbTlo)
    {
        // TODO Implement EJB TLO to EJB POJO Request
    }
}
```

Figure C-24 EJBRequestHandlerImpl

3. You need to understand the mapping between CustomerInformation and EJBCustomerRepoSessionBeanBG, and implement the mapping logic by yourself. Here is the example logic for the Output component:
 - a. Update the method EJBCustomerRepoSessionBeanBG_Async with Example C-2 on page 642.

Example C-2 Sample for EJBCustomerRepoSessionBeanBG_Async method

```
public void EJBCustomerRepoSessionBeanBG_Async(DataObject ejbTlo)
{
    // TODO Implement EJB TLO to EJB POJO Reques
    CustomerRepoBeanAsync crAsync =
(CustomerRepoBeanAsync)locateService_CustomerRepoBeanPartner();
    CustomerInformation ci = convertB02EJBData(ejbTlo);
    Ticket tic = crAsync.getCustomerInfoAsync(ci);

    CustomerInformation response = null;
    try {
        response = crAsync.getCustomerInfoResponse(tic, Service.WAIT);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    System.out.println(convertEJBData2B0(response));
}
```

- b. Update the method EJBCustomerRepoSessionBeanBG_Sync with Example C-3.

Example C-3 Sample for EJBCustomerRepoSessionBeanBG_Sync method

```
public DataObject EJBCustomerRepoSessionBeanBG_Sync(DataObject ejbTlo)
{
    CustomerRepoBean crSync = locateService_CustomerRepoBeanPartner();

    CustomerInformation request = convertB02EJBData(ejbTlo);
    CustomerInformation response = null;
    try {
        response = crSync.getCustomerInfo(request);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    DataObject tlo2bo = this.convertEJBData2B0(response);
    return tlo2bo;
}
```

- c. Add the method `convertEJBData2BO` to convert from the Java object to SDO with Example C-4.

Example C-4 Sample to convert from a Java object to SDO

```
public DataObject convertEJBData2BO(CustomerInformation ci){
    String addr = ci.getAddr();
    String firstName = ci.getFirstName();
    String id = ci.getId();
    String lastName = ci.getLastName();

    BOFactory bof =
    (BOFactory)ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory");
    DataObject sessionBeanBG =
    bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/EJBCustomerRepoSessionBean/EJBCustomerRepoSessionBeanBG", "EJBCustomerRepoSessionBeanBG");
    DataObject sessionBeanBO =
    sessionBeanBG.createDataObject("EJBCustomerRepoSessionBean");

    DataObject infoBO =
    bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/EJBCustomerInformation", "EJBCustomerInformation");

    DataObject setAddrBO =
    bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/EJBCustomerInformation_setAddr", "EJBCustomerInformation_setAddr");
    setAddrBO.setString("String_1", addr);

    DataObject setFirstNameBO =
    bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/EJBCustomerInformation_setFirstName", "EJBCustomerInformation_setFirstName");
    setFirstNameBO.setString("String_1", firstName);

    DataObject setIdBO =
    bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/EJBCustomerInformation_setId", "EJBCustomerInformation_setId");
    setIdBO.setString("String_1", id);

    DataObject setLastNameBO =
    bof.create("http://www.ibm.com/websphere/crossworlds/2002/BOSchema/EJBCustomerInformation_setLastName", "EJBCustomerInformation_setLastName");
    setLastNameBO.setString("String_1", lastName);

    infoBO.setDataObject("setAddr", setAddrBO);
    infoBO.setDataObject("setFirstName", setFirstNameBO);
}
```

```

        infoB0.setDataObject("setId", setIdB0);
        infoB0.setDataObject("setLastName", setLastNameB0);

        DataObject getCustomerInfoB0 =
bof.create("http://www.ibm.com/websphere/crossworlds/2002/B0Schema/EJBCustomerRepoSes
sionBean_getCustomerInfo", "EJBCustomerRepoSessionBean_getCustomerInfo");
        getCustomerInfoB0.setDataObject("Return_Value", infoB0);

        sessionBeanB0.setDataObject("getCustomerInfo", getCustomerInfoB0);

        return sessionBeanBG;
    }

```

- d. Add a new method `convertEJBData2BO` to convert from SDO to a Java object with Example C-5.

Example C-5 Sample to convert from SDO to a Java object

```

public CustomerInformation convertB02EJBData(DataObject sessionBeanBG){

    CustomerInformation ci = new CustomerInformation();

    DataObject sessionBeanB0 =
sessionBeanBG.getDataObject("EJBCustomerRepoSessionBean");
    DataObject infoB0 =
sessionBeanB0.getDataObject("getCustomerInfo").getDataObject("CustomerInformation");
    String addr = infoB0.getDataObject("getAddr").getString("Return_Value");
    String firstName =
infoB0.getDataObject("getFirstName").getString("Return_Value");
    String id = infoB0.getDataObject("getId").getString("Return_Value");
    String lastName =
infoB0.getDataObject("getLastName").getString("Return_Value");

    ci.setAddr(addr);
    ci.setFirstName(firstName);
    ci.setId(id);
    ci.setLastName(lastName);

    return ci;
}

```

C.3.5 Testing and verification

In this section, we explain how to run events through the migrated example in WebSphere Process Server and verify the results:

1. Start the WebSphere Process Server V6.2 server from the WebSphere Integration Developer Servers view (Figure C-25).

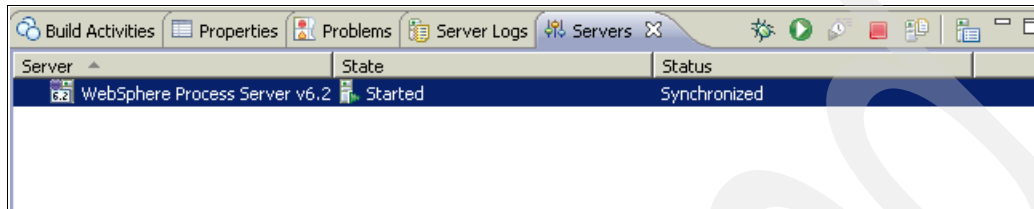


Figure C-25 Start WebSphere Process Server in WebSphere Integration Developer

2. To test the module by using the Unit Test Environment (UTE), add the **EJBConnector** and **CustomerRetrieve** modules to the server.
3. Make sure that the EJB service CustomerRepo is already deployed and running on the server.
4. From the Business Integration perspective, highlight the **CustomerRetrieve** module, and click **Test** → **Test Module**.

5. In the unit test window (Figure C-26), follow these steps:
 - a. For Module, select **CustomerRetrieve**.
 - b. For Component, select **CustomerRetrieve_from**.
 - c. For Interface, select **Execute_PortType**.
 - d. For Operation, select **Execute_Sync**.
 - e. In the parameter section, set the verb to **Retrieve**.
 - f. In the parameter section, for user ID, type **its0**. Make sure to enter this value under getId → Return_Value.
 - g. Click **Continue**.

▼ Detailed Properties

Configuration: Default Module Test

Module: CustomerRetrieve

Component: CustomerRetrieve_from

Interface: Execute_PortType

Operation: Execute_Sync

Initial request parameters

Name	Type	Value
EJBCustomerRepoSessionBeanBG	EJBCustomerRepoSessionB...	✓
verb	verb<string>	✓ Retrieve
EJBCustomerRepoSessionBean	EJBCustomerRepoSessionB...	✓
getCustomerInfo	EJBCustomerRepoSessionB...	✓
CustomerInformation	EJBCustomerInformation	✓
getAddr	EJBCustomerInformation_g...	✓
setAddr	EJBCustomerInformation_s...	✓
getFirstName	EJBCustomerInformation_g...	✓
setFirstName	EJBCustomerInformation_s...	✓
getId	EJBCustomerInformation_g...	✓
Return_Value	Return_Value<string>	✓ its0
Primary_Key	Primary_Key<string>	✓
@ version	token	✓ 0.0.0
@ delta	boolean	✓ false
@ locale	string	✓

Figure C-26 Test setup

- Examine the objects that are returned from the remote CustomerRepo service, and see how it maps to a business graph that is returned from the EJBConnector module (Figure C-28).

Detailed Properties

Module: [CustomerRetrieve](#)
Source component: [CustomerRetrieve from](#)
Source reference: [to Output](#)
Target component: [EJBCollab from To to](#)
Target interface: [to Output PortType](#)
Target operation: [Retrieve_Sync](#)

Response parameters

Name	Type	Value
EJBCustomerRepoSession...	EJBCustomerRepoSessionBeanBG	✓
verb	verb<string>	✓
EJBCustomerRepoSessionI	EJBCustomerRepoSessionBean	✓
getCustomerInfo	EJBCustomerRepoSessionBean_ge...	✓
CustomerInformat	EJBCustomerInformation	✓
Return_Value	EJBCustomerInformation	✓
getAddr	EJBCustomerInformation_getAddr	✓
setAddr	EJBCustomerInformation_setAddr	✓
String_1	String_1<string>	✓ 3039 E. Cornwallis Road, Bldg 305, Researc..
Primary_K	Primary_Key<string>	✓
@ version	token	✓
@ delta	boolean	✓
@ locale	string	✓
getFirstName	EJBCustomerInformation_getFirst...	✓
setFirstName	EJBCustomerInformation_setFirstN...	✓
String_1	String_1<string>	✓ Jane
Primary_K	Primary_Key<string>	✓
@ version	token	✓
@ delta	boolean	✓
@ locale	string	✓
getId	EJBCustomerInformation_getId	✓
setId	EJBCustomerInformation_setId	✓
String_1	String_1<string>	✓ itso
Primary_K	Primary_Key<string>	✓

Figure C-28 Return object mapped to business graph

C.4 Conclusion

In this chapter, we showed how to migrate WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture to WebSphere Process Server environment. We demonstrated two upgrade paths for WebSphere Business

Integration Adapter for Enterprise JavaBeans Architecture. And, we tested the migrated application in the WebSphere Process Server environment.

Archived

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247415-01>

Alternatively, you can go to the IBM Redbooks Web site at:

<http://www.ibm.com/redbooks>

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247415-01.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
All chapter Samples.zip	Compressed code samples for all chapters
Chapter 12 samples.zip	Compressed code samples for Chapter 12, “Post-migration tasks” on page 253
Chapter 14 samples.zip	Compressed code samples for Chapter 14, “Preparation for the technical solutions” on page 279
Chapter 15 samples.zip	Compressed code samples for Chapter 15, “Data access scenario with technology adapters” on page 293
Chapter 16 samples.zip	Compressed code samples for Chapter 16, “Data synchronization scenario with technology adapters” on page 343
Chapter 17 samples.zip	Compressed code samples for Chapter 17, “Data synchronization scenario with application adapters” on page 431
Appendix B samples.zip	Compressed code samples for Appendix B, “Access EJB migration” on page 591
Appendix C samples.zip	Compressed code samples for Appendix C, “Migrate WBI Adapter for EJB Architecture” on page 615

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

adapters Software programs that know how to interact with an underlying business application and act as an interface between two disparate systems, thereby enabling information exchange between them.

administration console The Web interface to the WebSphere Application Server, WebSphere Process Server, or WebSphere Enterprise Service Bus. It offers powerful options for configuration and deployment.

API See *application programming interface*.

application programming interface (API) A source code interface that a computer system or program library provides to support requests for services to be made of it by a computer program. The software that provides the functionality that is described by an API is said to be an implementation of the API. The API is abstract, in that it specifies an interface and does not get involved with implementation details.

binding Determine how a service connects to and interacts with an application. Bindings are a critical part of an import or export component. They are used to create services to communicate with an Enterprise Information System (EIS).

BPC See *Business Process Choreographer*.

BPEL See *Business Process Execution Language*.

bus Provides the possibility to transfer data between different nodes that are connected to the bus.

business object A business object represents a data entity. A data entity can be a collection of data that can be treated as a unit.

business process A business Process defines a flow of actions to solve a special task. It can be stored in a BPEL file.

Business Process Choreographer (BPC) A component of the WebSphere Process Server runtime architecture that provides support for business processes and human tasks. It offers a way to model business processes based on the WS-BPEL specification and to model interactions that involve human beings, 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 (SOA).

Business Process Execution Language (BPEL) An open standard to store process definitions in a file.

CEI See *Common Event Infrastructure*.

Common Base Event A specification based on XML that defines a mechanism for managing events, such as logging, tracing, management, and business events, in business enterprise applications.

Common Event Infrastructure (CEI) A part of the SOA core that can be used to capture events to monitor applications, for example, in IBM WebSphere Business Monitor or Tivoli products. WebSphere Process Server builds on and uses CEI to emit a specific set of events for each SCA service component.

composite applications Applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes.

data binding Handles the transformation of data passed as a Service Data Object (SDO) in a Service Component Architecture (SCA) application and the native format for an EIS Java 2 Connector (J2C) or messaging Java Message Service (JMS).

data handler A Java class instance that converts between a particular serialized format and a business object. Is used by components of a business integration system that transfers information between a WebSphere Business Integration broker and some external process.

data map Maps different business objects together. They might include move, join, extract, assign, or custom operations.

debug perspective The perspective in the WebSphere Integration Developer that is used for debugging a business process and Java or anything associated with them.

event messages Messages produced by software on a machine indicating a specific event or error.

Extensible Stylesheet Language (XSL) 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. The family includes the XSL Transformations (XSLT), XSL Formatting Objects (XSL-FO), and XML Path Language (XPath) languages.

foreign bus Connects different buses together by configuration. Extending a bus network in this way enables all the buses in the network to exchange messages.

Human Task Manager A component of the WebSphere Process Server runtime architecture that supports the ad hoc creation and tracking of tasks. You can use existing LDAP directories (and operating system repositories and the WebSphere user registry) to access staff information.

inbound Entering into the Integration Solution System. The Integration Solution System can be WebSphere interChange Server or WebSphere Process Server.

integration broker Enables diverse applications to exchange information in dissimilar forms by handling the processing that is required for the information to arrive in the right place and in the correct format.

interface mapping Used to act as a connection between different interfaces. They include operation maps and parameter maps.

J2EE Java 2 Platform Enterprise Edition. The former name up to the version of the currently known Java EE. See *Java Platform, Enterprise Edition*.

Java EE Connector Architecture (JCA) A Java EE-based technology solution for connecting application servers and EISs.

Java EE See *Java Platform, Enterprise Edition*.

Java Platform, Enterprise Edition (Java EE) An environment for developing and deploying enterprise applications, defined by Sun™ Microsystems Inc. Consists of a set of services, APIs, and protocols that provides the functionality for developing multi-tiered, Web-based applications, based largely on modular software components that are running on an application server. For more details about Java EE, see the following Web page: <http://java.sun.com/javaee/>

Java snippet A small part of a program's code written in Java.

JCA See *Java EE Connector Architecture*.

Java Message Service (JMS) Part of the J2EE suite that provides standard APIs that Java developers can use to access the common features of enterprise message systems. Supports publish/subscribe and point-to-point models and allows the creation of message types that consist of arbitrary Java objects.

Java Naming and Directory Interface (JNDI) An API that is specified in Java technology that provides naming and directory functionality to applications that are written in the Java programming language. It is designed especially for the Java platform by using the Java object model. Provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. By using JNDI, applications that are based on Java technology can store and retrieve named Java objects of any type.

JMS See *Java Message Service*.

JNDI See *Java Naming and Directory Interface*.

LDAP See *Lightweight Directory Access Protocol*.

LDAP Data Interchange Format (LDIF) A standard data interchange format for representing (LDAP) directory content and directory update (add, modify, delete, and rename) requests. Represents update requests as a set of records, one record for each update request. In both cases, the data is presented in a plain text form.

LDAP directory Often reflects various political, geographic, and organizational boundaries, depending on the model chosen. LDAP deployments today tend to use Domain Name System (DNS) names for structuring the topmost levels of the hierarchy. Deeper inside the directory, entries might appear that represent people, organizational units, printers, documents, groups of people, or anything that represents a given tree entry (or multiple entries).

LDIF See *LDAP Data Interchange Format*.

Lightweight Directory Access Protocol (LDAP) A networking protocol for querying and modifying directory services that run over TCP/IP.

mapping A method of message transformation using drag and drop from references to message and database definitions.

mapping node A node in a message flow that uses message mappings to construct an output message that uses other messages or information from database tables.

MDB See *Message-Driven Bean*.

mediation A mapping with advanced actions. Can reorder and reformat every input so that the target systems get the needed items with the right name and value.

mediation flow Defines the mediation step by step. It can include Extensible Stylesheet Language (XSL) transformations and custom transformation by using Java snippets.

Message-Driven Bean (MDB) An EJB that is similar to a session bean, except that it responds to a JMS message rather than an RMI event. MDBs were introduced in the EJB 2.0 specification, which is supported by J2EE 1.3 and later. The message bean represents the integration of JMS with EJB to create an entirely new type of bean that can handle asynchronous JMS messages.

operation map Maps different interfaces. Interfaces can have different names for their operations that need to be mapped together.

outbound Exiting from the Integration Solution System. The Integration Solution System can be WebSphere InterChange Server or WebSphere Process Server.

parameter map Defines the mapping of the attributes in an operation map.

queue manager A system program that provides queuing services to applications. Enables communication between the WebSphere Message Broker components. Each component requires access to a queue manager.

SCA See *Service Component Architecture*.

SCA component A part of an SCA module that has a specific functionality. Can be assembled to a meaningful service.

SCA export A capability to provide an interface that can be called by other SCA modules by using an import.

SCA import A capability to call an interface of another SCA module that is provided by using an export.

SCA module A logical unit of different SCA components that build a service.

SDO See *Service Data Object*.

Service Component Architecture (SCA) The base for an SOA because different services can be easily connected to solutions.

Service Data Object (SDO) A data representation that is mostly used to connect to an EIS. Cache data so that the consumer does not have to be concerned about interacting with the EIS directly.

Service Integration Bus A logical entity based on the physical implementation of a message-driven bean, there is no inherent high availability or workload management functionality.

service-oriented architecture (SOA) A business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks or services. Helps to build composite applications. See *composite applications*.

SOA See *service-oriented architecture*.

SOAP A protocol for exchanging XML-based messages over computer networks, normally by using HTTP. Forms the foundation layer of the Web services stack and provides a basic messaging framework upon which more abstract layers can build.

Web Services Description Language (WSDL) An XML-based language that provides a model for describing Web services. An XML-based service description on how to communicate by using Web services. Defines services as collections of network endpoints or ports. A WSDL specification provides an XML format for documents for this purpose.

WebSphere Application Server A powerful application server offered by IBM. It is included as a powerful foundation in other products, such as IBM WebSphere Portal and WebSphere Process Server.

WebSphere MQ A messaging application that enables the Message Brokers Toolkit, Configuration Manager, and brokers to communicate. Provides many of the available transport protocols between business applications and message flows.

WebSphere MQ Explorer A graphical user interface for WebSphere MQ to administer WebSphere MQ components, such as queue managers, channels, and queues.

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 a reference to an interface.

WSDL See *Web Services Description Language*.

XML Schema Definition (XSD) An instance of an XML schema written in XML Schema definition language. An XML Schema Definition file has the extension .xsd. The prefix “xsd” is also typically used in the XML elements of the XSD file to indicate the XML Schema namespace.

XSD See *XML Schema Definition*.

XML Schema instance An XSD and typically has the filename extension “.xsd.” The language is sometimes informally referenced as XSD.

XSL See *Extensible Stylesheet Language*.

Abbreviations and acronyms

ADK	An Adapter Development Kit	EMD	Enterprise Metadata Discovery
ALE	Application Link Enabling	ERP	Enterprise Resource Planning
ARM	Application Response Measurement	ESB	enterprise service bus
ASBO	application-specific business object	FEM	failed event manager
ASBOs	application-specific business objects	FFDC	First-failure data capture
BIVA	Business Integration Value Assessment	GBO	generic business object
BPEDB	business process engine database	GBOs	generic business objects
BPEL	Business Process Execution Language	HA	high availability
BPM	Business Process Management	HACMP	High Availability Cluster Multi-Processing for AIX
CCI	Common Client Interface	HAPI	Heritage API
CEI	Common Event Infrastructure	IBM	International Business Machines Corporation
CRM	Customer Relationship Management	IDE	integrated development environment
CVS	Concurrent Versions System	IDL	Interface Definition Language
CoE	Center of Excellence	IIOP	Internet Inter-ORB Protocol
DTD	document type definition	ISV	Independent Software Vendor
EAI	Enterprise Application Integration	ITSO	International Technical Support Organization
EAR	enterprise archive	J2C	J2EE Connector Architecture
EARs	Enterprise Application Archives	J2EE	Java 2 Platform, Enterprise Edition
EDI	electronic data interchange	J2SE	Java 2 Platform, Standard Edition
EIS	Enterprise Information System	JAR	Java archive
EISs	Enterprise information systems	JCA	J2EE Connector Architecture
EJB	Enterprise JavaBeans	JCo	Java Connector
		JDBC	Java Database Connectivity
		JDK	Java Developer Kit
		JMS	Java Message Service
		JNDI	Java Naming and Directory Interface

JSF	JavaServer Faces	SNMP	Simple Network Management Protocol
JVM	Java virtual machine	SOA	service-oriented architecture
JVMs	Java virtual machines	SOMA	Service-oriented Modeling and Architecture
KPIs	key performance indicators	SOP	sales order processing
LDAP	Lightweight Directory Access Protocol	SSO	single sign-on
LLBP	long-lived business process	TCO	total cost of ownership
LLBPs	long-lived business processes	TOG	The Open Group
MDB	Message-Driven Bean	URL	Uniform Resource Locator
MDBs	Message-Driven Beans	UTE	Unit Test Environment
MFC	Mediation Flow Component	VCS	Version Control System
MIB	Management Information Base	VTC	Visual Test Connector
MIME	Multipurpose Internet Mail Extension	WID	WebSphere Integration Developer
MSCS	Microsoft Cluster Server	WSDL	Web Services Description Language
ND	Network Deployment	XSD	XML Schema Definition
ODA	Object Discovery Agent		
ORB	Object Request Broker		
PMI	Performance Monitoring Infrastructure		
QoS	quality of service		
RAR	resource adapter archive		
RFC	Remote Function Call		
ROI	return on investment		
RUP	Rational Unified Process		
SAX	Simple API for XML		
SCA	Service Component Architecture		
SDO	Service Data Object		
SDOs	Service Data Objects		
SIBus	Service Integration Bus		
SIMM	Service Integration Maturity Model		
SLA	service level agreement		
SLAs	service level agreements		

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 publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 660. Note that several of the documents referenced here might be available in softcopy only:

- ▶ *WebSphere Business Process Management V6.1.2 Production Topologies*, SG24-7665
- ▶ *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413
- ▶ *WebSphere Business Integration for SAP*, SG24-6354

Online resources

These Web sites are also relevant as further information sources:

- ▶ Open Group Service Integration Maturity Model
<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-simm/>
<http://www.opengroup.org/projects/osimm/>
<http://www.openinnovations.us/events/q306/arsanjani-diaz.htm>
- ▶ Introduction to SOA governance
<http://www-306.ibm.com/software/solutions/soa/gov/>
- ▶ License procurement and maintenance Extended Entitlement
<http://www.ibm.com/software/integration/wps/library/entitlement.html>
- ▶ Business Process Management (BPM) and Workflow design patterns
<http://www.workflowpatterns.com>

- ▶ DeveloperWorks article: “WebSphere Process Server security overview”
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0602_khangaonkar/0602_khangaonkar.html
- ▶ “Dynamicity and versioning (right granularity of modules to facilitate advanced concepts)”
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0602_brown/0602_brown.html
- ▶ The SDO JSR235 specification
<http://jcp.org/en/jsr/detail?id=235>
- ▶ “Migrating WebSphere InterChange Server artifacts to WebSphere Process Server artifacts, Part 1: Migrating collaboration templates to BPEL”
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0612_seacat/0612_seacat.html
- ▶ “Migrating WebSphere InterChange Server artifacts to WebSphere Process Server artifacts, Part 2: Understanding the WebSphere Process Server SCA components”
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0703_seacat/0703_seacat.html
- ▶ WebSphere Process Server V6.2 information center
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/welcome_wps.html
- ▶ WebSphere Integration Developer V6.2 information center
<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.nav.doc/topics/welcome.html>

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Index

A

- Access EJB 248
 - support in WebSphere Process Server 250
- activities 96
- Adapter Artifacts Importer 208
- adapter configuration 244
- adapter mapping 134
- Adapter Upgrade Kit
 - WebSphere Adapters 245
 - WebSphere Business Integration Adapter 240
- adapters 77
 - application 134
 - custom 147
 - equivalent functions 136
 - technology 134
 - types 134
- administrative console 61
- administrative tasks 61
 - administrative console 61
 - scheduling 62
 - scripting 62
- application adapters 134
- application domain 11
- application module errors 560
- application-specific business object (ASBO) 164
 - impact from changing WebSphere Business Integration Adapters 161
- architectural design patterns 152
- architecture
 - design 47
 - domain 11
- Artifact Importer 208, 211
- artifact migration 215
 - business objects 217
 - collaboration 227
 - WebSphere InterChange Server 227
 - files generated 216
 - maps 222
 - prerequisites and constraints 216
 - relationship artifacts 239
 - WebSphere Adapters 245
 - WebSphere Business Integration Adapters 244
 - adapter configuration 244
 - business objects 244
 - WebSphere InterChange Server, Server Access Interface 248
- as-is migration
 - data access project migration 295
- assembly editor 184
- assessment phase 33, 39
 - current IT environment context 39
 - current WebSphere InterChange Server project context 39
 - deliverables 62
 - high level planning 64
 - options to deliver requirements 63
 - requirements list 63
 - risk assessment and mitigation plan 63
 - ROI study 63
 - impact analysis 40–41
 - architecture and design 47
 - deployment 50
 - development 47
 - methodology 45
 - operations 58
 - organizational 41
 - project delivery 42
 - run time, quality of service, security 54
 - testing 49
 - requirements gathering 40
- assessment refinement 66
 - decisions, precise estimates for effort, budget 67
 - detailed study 66
- asynchronous inbound service calls, collaborations 234
- asynchronous recovery 80
- automatic project build 407, 485

B

- beneficial internal IT factors 35
- bindings
 - EIS 102
 - EJB 102
 - generic JMS 103
 - HTTP 103

- JMS 103
- Web service 103
- WebSphere MQ 104
- WebSphere MQ JMS 103
- WebSphere Process Server 101
- BOconverter 165
- BPM enabled by SOA, business, IT alignment 41
- break nodes, collaborations 233
- build time
 - software upgrade process 48
 - supported platforms 48
- business flow manager 17
- business graph 222
 - map in WebSphere Process Server 226
- Business Integration perspective 183
- business logic 31, 156
- business object 76, 163, 219, 222
 - application-specific 76
 - artifact migration 217
 - WebSphere Business Integration Adapter 244
 - WebSphere InterChange Server 217
 - generic 76
 - metadata 222
 - migration of 220
 - SDOs 93
 - services 222
 - translation into SDOs 163
- business object editor 183–184
- business object map 95, 223
 - WebSphere Process Server 226
- business object mapping editor 187
- business process 96
 - instance in a failed or stopped state 167
 - instances in failed state 61, 514
 - logic 156
 - WebSphere Process Server 228
- business process or workflow pattern 31
- business rules 97
- business state machine 17, 96
- business view domain 11

C

- capacity planning 53
- CEI (Common Event Infrastructure) 93
- Center of Excellence (CoE) 41
- CoE (Center of Excellence) 41
- collaboration 75, 227–228

- asynchronous inbound service calls 234
- break nodes 233
- custom code 236
- custom properties 236
- decision nodes 231
- LLBP 236
- loops 233
- migration of 231
- multiple triggering ports 235
- object 228
- template 228
- transactional 80
- WebSphere InterChange Server 227
- Common Event Infrastructure (CEI) 93
- complex data handler 89
- component architecture of the solution 50
- component mapping 125
- component models 153, 155
- componentized (hub and spoke) integration 15
- composite services 17
- connectivity 153
- connector
 - code 14
 - configuration 14
 - controller 78
- contactR 239
- Create Customer Collaboration 515
- cross-reference, mapping rule 223
- current environment 39
- custom code and standard migration tool constraints 173
- custom code in collaborations 236
- custom data handler 89
- custom properties, collaborations 236
- custom, mapping rule 223
- CUSTOMERID 424

D

- data access pattern 27
- data access project migration 295
 - deployment 319
 - development 297
 - operational model 296
 - premigration 296
 - software environment 295
 - startup error message 327
 - target environment 295
 - testing module setup 331

- data bindings
 - custom 108
 - EIS 105
 - HTTP 107
 - JMS 106
 - WebSphere MQ 107
 - WebSphere Process Server 104
- data handlers 87
 - complex 89
 - custom 89, 108
 - delimited 88
 - EDI 89
 - FixedWidth 88
 - replacing WebSphere Business Integration data handlers 145
 - WebSphere Business Integration 87
 - XML 87
- data synchronization pattern 25
- data synchronization project migration
 - deployment 407, 485
 - development 438
 - end-to-end solution testing 419, 497
 - Jython script and JDBC data sources 401, 478
 - premigration 346, 434
 - preparation 348, 434
 - software environment 346, 433
 - target environment 346, 433
 - testing on WebSphere InterChange Server 348, 435
 - workaround for dynamic relationship 388, 467
- data synchronization with dependencies pattern 26
- data transformation 145, 153
- data translation 153
- database
 - setup, software migration 281
- database connection pools 79
- database connectivity service 79
- DataBinding 105
- decision nodes, collaborations 231
- decisions, precise estimates for effort, budget 67
- deferred recovery 81
- delimited data handler 88
- deliverables 62
 - high level planning 64
 - options to deliver requirements 63
 - requirements list 63
 - risk assessment and mitigation plan 63
 - ROI study 63
- deployment 50, 68
 - capacity planning 53
 - component architecture of the solution 50
 - data access project migration 319
 - data synchronization project migration 407, 485
 - packaging 50
 - product coexistence 53
 - topology for high availability and scalability 51
- detailed study, migration planning 66
- development 47, 68
 - concepts and skills building 48
 - data access project migration 297
 - data synchronization project migration 438
 - enhanced error handling 519
 - SCA 47
 - software upgrade process for build time 48
 - supported platforms for build time 48
 - team development 49
- diagnostic trace 565
- domains, SIMM 11
- dynamic relationship, workaround 388, 467
- dynamically reconfigurable services 20

E

- Eclipse 3.0 181
- EDI data handler 89
- editor 183
 - assembly 184
 - business object 184
 - business object mapping 187
 - interface 186
 - interface mapping 186
 - process 185
 - relationship 189
 - visual snippet 189
- EIS binding 102
- EIS data binding 105
- EIS subclasses 246
- EJB binding 102
- EMD (Enterprise Metadata Discovery) 246
- end-to-end privacy 56
- end-to-end security 56
- end-to-end solution testing
 - data synchronization project migration 419, 497
- enhanced error handling 513
- Enterprise Metadata Discovery (EMD) 246
- Enterprise Service Discovery wizard 191
- error handling 56, 167
 - enhanced 513

- development 519
- target environment 512
- testing and verification 530
- error management 60
 - event 60
- errors, application module 560
- event 60
- event management service 78
- event sequencing 55, 167
- export 101
- export binding 101

F

- failed event 61, 167, 514
- Failed Event Manager 60
- fine-grained administration capabilities 129
- FixedWidth data handler 88
- fixed-width string 88
- fixing long queue names 560
- flow control 55, 129, 166
- Flow Manager 60
- foundation classes 246
- functional gaps between products 166

G

- generic business objects 76
- generic JMS binding 103
- go live 69
- groups 563

H

- HAPI (Heritage API) 146
- Heritage API (HAPI) 146, 168
- high availability topology 51
- high level planning 64
- HTTP binding 103
- HTTP data bindings 107
- human tasks 96

I

- identity relationship 237, 425
- impact analysis 40
 - architecture and design 47
 - deployment 50
 - capacity planning 53
 - component architecture of the solution 50
 - packaging 50

- product coexistence 53
- topology for high availability and scalability 51
- development 47
 - concepts and skills building 48
 - platforms for build time 48
 - SCA 47
 - software upgrade process for build time 48
 - team development 49
- methodology 45
- operations
 - administrative tasks 61
 - concepts and skills building 62
 - error management 60
 - monitoring 58
- organizational 41
 - BPM enabled by SOA, business, IT alignment 41
 - Center of Excellence 41
 - SOA and IT governance 41
- performances, tuning 58
- project delivery 42
 - costs, funding 42
 - resources and skills enablement 43
 - service level agreement 45
 - timeline and product roadmap 44
- quality of service 55
- run time
 - installed 54
 - software upgrade process 55
 - supported platforms 55
- run time, quality of service, security 54
- security 56
- testing 49
 - performance testing, benchmark 50
 - regression testing 49
 - unit testing 49
- implementation phase 33, 67
 - deployment 68
 - development 68
 - go live 69
 - testing 68
- import 101
- import binding 101
- inbound event processing 100
- information domain 11
- infrastructure domain 11
- integration 153
 - concepts and solutions 152

- layers 153
- runtime 148
- test client 191
- tools 148
- integration layers
 - connectivity 153
 - data transformation 153
 - integration logic 154
- integration logic 154, 156
 - mediation flow 158
 - module 158
- integrity 56
 - plus privacy 56
- InterChange Server repository 78
- interface 95
- interface editor 186
- interface mapping editor 186
- interface maps 95, 223
- interruptible process 228
- IT governance 41
- IT migration factors, internal 35
 - beneficial 35
 - nonbeneficial 36
- iterator node 233

J

Java

- developments 168

JDBC data sources

- data synchronization project migration 401, 478

JMS binding 103

- service integration bus 139

JMS data bindings 106

join, mapping rule 223

JText Connector file system setup for migration scenarios 290

JVM logs 566

Jython script

- data synchronization project migration 401, 478

L

layer 153

library, shared 190

LLBP (long-lived business processes)

- business process or workflow pattern 31
- collaborations 236

log and trace configuration options 565

logging level 561

logical rollback 155

long queue names 560

long-lived business processes (LLBP)

- business process or workflow pattern 31
- collaborations 236

lookup relationship 237

loops, collaborations 233

M

managed runtime environment 148

map artifact 551

mapping

- ASBOs 164
- collaboration templates to BPEL processes 127
- components 125

mapping rules

- WebSphere InterChange Server 223
- WebSphere Process Server 224

maps 76, 222

artifact migration

- WebSphere InterChange Server 222
- WebSphere Process Server 223

business object 223

impact from changing WebSphere Business Integration Adapter 161

interface 223

migration of 224

mediation flow component, integration logic 158

mediation flows 94

message

- startup error in data access project migration 327
- WebSphere Process Server log and trace files 568

messaging infrastructure 53

- service integration bus (SIBus) 53

messaging-based dead letter queues 61, 167, 513

methodology, migration planning 45

methods domain 11

migrated code

- performances of 174
- quality of 173

migration

- as-is
 - data access project 295
- implementation challenges 160
- functional gaps between products 166
- specific Java developments 168

- WebSphere Business Integration Adapters 161
 - integration concepts and solutions 152
 - routing logic to WebSphere Process Server 539
 - troubleshooting 559
 - usage patterns 23
 - migration implementation
 - architectural design patterns 152
 - component model 153
 - integration layers 153
 - selection criteria 169
 - custom code and standard migration tool constraints 173
 - performances of automatically migrated code 174
 - quality of automatically migrated code 173
 - technical approaches 169–170
 - migration path, defining
 - migration optimization 38
 - preparation 38
 - target 37
 - migration planning 33
 - assessment phase 33, 39
 - considerations 34
 - implementation phase 33, 67
 - internal IT factors 35
 - beneficial 35
 - nonbeneficial 36
 - path evaluation 34
 - preparation phase 33, 64
 - project phases 33
 - recommendations for defining the migration path 37
 - migration scenarios
 - file system setup for JText Connector 290
 - testing environment 281
 - WebSphere MQ configuration 289
 - migration tools 125
 - constraints 49
 - Migration wizard 181
 - reposMigrate utility 181
 - supported versions 180
 - WebSphere Integration Developer 181
 - WebSphere InterChange Server 180
 - Migration wizard 192
 - migration method 192
 - module 190
 - monitoring 58
 - PMI 59
 - WebSphere InterChange Server 58
 - WebSphere Process Server 59
 - move, mapping rule 223
 - multicast capabilities 537
 - multireceiver capabilities 537
- N**
- native store and forward capability 150
 - native_stderr.log file 560
 - native_stdout.log file 560
 - nonbeneficial internal IT factors 36
 - nonidentity relationship 237
- O**
- Open Group Service Integration Maturity Model 10
 - open standards-based implementation 148
 - operational model
 - data access project migration 296
 - operations 58
 - concepts and skills building 62
 - optimization options, advanced
 - WebSphere InterChange Server 535
 - organization
 - Center of Excellence 41
 - domain 11
 - SOA and IT governance 41
 - outbound event processing 100
- P**
- packaging 50
 - participant 237
 - patterns 23
 - Performance Monitoring Infrastructure (PMI) 59
 - performance testing and benchmarking 50
 - performances and tuning 58
 - persist service call in-transit state 81
 - PMI (Performance Monitoring Infrastructure) 59
 - point-to-point integration 14
 - premigration
 - data access project migration 296
 - data synchronization project migration 346, 434
 - preparation
 - data synchronization project migration 348, 434
 - preparation phase 33, 64
 - assessment refinement 66
 - skills building 64
 - WebSphere Process Server project experience

- 64
- privacy 56
 - end-to-end 56
 - integrity plus 56
- Process Designer 75
- process editor 185
- product coexistence 53
- product roadmap 44
- project
 - costs and funding 42
 - resources and skills enablement 43
- project delivery 42
 - costs and funding 42
 - resources and skills enablement 43
 - service level agreement 45
 - timeline and product roadmap 44
- project timeline and product roadmap 44

Q

- QoS (quality of service) 54–55
- quality of automatically migrated code 173
- quality of service (QoS) 54–55
- queue-based messaging 85

R

- Rational Software Development Platform 181
- recovery 80
- Redbooks Web site 660
 - Contact us xxxvii
- regression testing 49
- relational databases 53
- relationship artifacts 239
- relationship definition 238
- relationship editor 189
- relationship instance data 238
- relationship role 238
- relationships 95, 237
 - artifact migration
 - relationships 237
 - WebSphere InterChange Server 237
 - WebSphere Process Server 237
 - categories 237
 - coexistence 240
 - identity 425
- remote agent support 149
- repos_copy command 205
- repository 78
- repository file

- WebSphere InterChange Server 297, 351, 438
- reposMigrate command 139
- reposMigrate utility 205
 - syntax 207
- requirements
 - gathering 40
 - list 63
 - main options for delivery 63
- Retrieve Customer Collaboration 515
- return on investment study 63
- risk assessment and mitigation plan 63
- roadmap, SIMM 10
- role definitions 238
- role-based access control 56
- routing logic optimization
 - WebSphere InterChange Server 535
- routing logic, migration into WebSphere Process Server 539
- run time 54
 - installed 54
 - software upgrade process 55
 - supported platforms 55
- run time, quality of service, security 54
- runtime integration 148

S

- SCA (Service Component Architecture) 47
 - WebSphere Process Server 92
- SCA components 95
- scalability topology 51
- scheduling, administrative tasks 62
- scripting for administration 62
- SDO (Service Data Object)
 - business object translation 163
 - business objects 93
- security 54, 56
 - end to end 56
 - integrity plus privacy 56
- selectors 95
- separation of concerns 155
 - benefits 160
 - integration logic and business logic 156
 - WebSphere InterChange Server 156
 - WebSphere Process Server 157
- server and adapters all together approach 170
- server artifacts 123
 - migration 125, 215
 - business objects 217

- collaboration 227
 - files generated 216
 - maps 222
 - prerequisites and constraints 216
 - WebSphere InterChange Server, Server Access Interface 248
- server migration
 - constraints 124
- Service Component Architecture (SCA) 47
 - WebSphere Process Server 92
- service components layer 91, 95
- Service Data Object (SDO)
 - artifact migration, WebSphere Process Server 218
 - business object translation 163
 - business objects 93
- service integration bus (SIBus)
 - JMS binding 139
 - messaging infrastructure 53
 - messaging-based dead letter queues 167
- Service Integration Maturity Model (SIMM)
 - domains 11
 - integration service levels 12
 - componentized (hub and spoke) 15
 - composite services 17
 - dynamically reconfigurable services 20
 - integrated (point to point) 14
 - services 16
 - silo 13
 - virtualized services 19
 - roadmap 10
 - WebSphere InterChange Server 9
 - levels 2-5 21
 - WebSphere Process Server 9
 - levels 2-7 21
- service level agreement (SLA) 45
- serviceDeploy command 205
- serviceDeploy utility 207
- services integration 16
- services pattern 29
- set value, mapping rule 223
- shared library 190
- SIBus (service integration bus)
 - JMS binding 139
 - messaging infrastructure 53
 - messaging-based dead letter queues 167
- siloed application 13
- SIMM (Service Integration Maturity Model)
 - domains 11
 - integration service levels 12
 - componentized (hub and spoke) 15
 - composite services 17
 - dynamically reconfigurable services 20
 - integrated (point to point) 14
 - services 16
 - silo 13
 - virtualized services 19
 - roadmap 10
 - WebSphere InterChange Server 9
 - levels 2-5 21
 - WebSphere Process Server 9
 - levels 2-7 21
- skills building 64
- SLA (service level agreement) 45
- SOA core layer 91–92
- SOA governance 41
- software environment
 - data access project migration 295
 - data synchronization project migration 346, 433
 - migration scenarios 281
- software upgrade process for build time 48
- solution migration troubleshooting
 - advanced methods 568
 - application module errors 560
 - data synchronization scenario 569–570
 - deployment in WebSphere Process Server 560
 - diagnostic trace 565
 - failed to start adapter 568
 - fixing long queue names 560
 - general methods 560
 - JVM logs 566
 - WebSphere Process Server log and trace 561
- split, mapping rule 223
- standard migration tools 125
- startServer.log file 560
- startup error message, data access project migration 327
- stateless session bean binding 102
- stopServer.log file 560
- store and forward 55
- stream 88
- submap, mapping rule 223
- supported platforms for build time 48
- supporting services layer 91, 94
- SystemErr.log file 560
- SystemOut.log file 560

T

- target environment
 - data access project migration 295
 - data synchronization project migration 346, 433
 - enhanced error handling 512
- team development 49
- technology adapters 134
- test phase 68
- testing 49
 - environment
 - database setup 281
 - setup 281
 - software 281
 - module setup 331
 - performance testing, benchmark 50
 - regression 49
 - unit 49
- testing and verification
 - enhanced error handling 530
- The Open Group 10
- tools integration 148
- topic-based messaging 85
- topology for high availability and scalability 51
- trace level components and packages 563
- trace.log file 560
- tracing level 561
- transaction service 79
- transactional collaborations 80
- Transition for Operations methodology 46
- triggering ports, collaborations 235
- troubleshooting
 - solution migration
 - advanced methods 568
 - application module errors 560
 - data synchronization scenario 569–570
 - deployment in WebSphere Process Server 560
 - diagnostic trace 565
 - failed to start adapter 568
 - fixing long queue names 560
 - general methods 560
 - JVM logs 566
 - WebSphere Process Server log and trace 561
- two-step approach for adapters 170

U

- uninterruptible process 229

Unit Test Environment (UTE)

- WebSphere Process Server 279
- unit testing 49
- upgrade
 - benefits in WebSphere Process Server 128
 - considerations 124
 - equivalent adapter functions 136
 - limitations in WebSphere Process Server 129
 - planning 124
 - server artifact migration 125
 - server migration constraints 124
- upgrade paths 137
 - WebSphere Business Integration Adapter to WebSphere Adapter 140
 - WebSphere Business Integration Adapter with WebSphere Process Server 138
 - WebSphere Business Integration Adapters with a binding 141
- usage patterns 23
 - business process or workflow 31
 - data access 27
 - data synchronization 25
 - data synchronization with dependencies 26
 - services 29
- user registry 57
- UTE (Unit Test Environment)
 - WebSphere Process Server 279

V

- virtualized services 19
- visual snippet editor 189

W

- Web service binding 103
- Web Services Description Language (WSDL)
 - Port Type interfaces 186
- WebSphere Access EJB 248
- WebSphere Adapter for Flat Files 101
- WebSphere Adapter for JDBC
 - data synchronization scenario failed to run 569–570
- WebSphere Adapter for SAP 100
- WebSphere Adapter Toolkit
 - custom adapter 147
- WebSphere Adapters 99
 - artifact migration 245
 - components 246
 - WebSphere Business Integration Adapters

- correlation with 135
 - differences with 247
- WebSphere Business Integration Adapter
 - custom 87
- WebSphere Business Integration Adapter for EJB 85
- WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture 142
 - replacing 142
- WebSphere Business Integration Adapter for HTTP, replacing 145
- WebSphere Business Integration Adapter for JDBC 86
- WebSphere Business Integration Adapter for JMS 85
- WebSphere Business Integration Adapter for JText 84
- WebSphere Business Integration Adapter for SAP 83–84
- WebSphere Business Integration Adapter for Web Services
 - replacing 144
- WebSphere Business Integration Adapter for Web services 86
- WebSphere Business Integration Adapter for WebSphere MQ 86
- WebSphere Business Integration Adapter Framework 82
- WebSphere Business Integration Adapters 82
 - Adapter Upgrade Kit 240
 - Artifact Importer 208, 211
 - artifact migration 244
 - WebSphere Adapters 245
 - WebSphere InterChange Server 240
 - WebSphere Process Server 242
 - benefits of upgrading 148
 - categories 134
 - correlation to WebSphere Adapters 134
 - data transformation 146
 - differences with WebSphere Adapters 247
 - factors for upgrade path 133
 - impact on ASBOs and maps 161
 - impact on runtime aspects 166
 - implementation with WebSphere Process Server 138
 - limitations 149
 - migration implementation challenges 161
 - migration tools 208
 - overview 132
 - replacing for WebSphere MQ and JMS 142
 - upgrade paths 133, 137
 - WebSphere Adapter 140
 - upgrade with a binding 141
 - WebSphere Process Server 243
 - WebSphere Process Server bindings 135
- WebSphere Business Integration data handlers 87
 - replacing 145
- WebSphere Business Integration Toolset 77
- WebSphere Enterprise Service Bus 94
- WebSphere Integration Developer
 - editors 183
 - Enterprise Service Discovery wizard 191
 - integration test client 191
 - migration terms and concepts 181
 - migration tools 181
 - Migration wizard 192
 - migration method 192
 - module 190
 - shared library 190
 - tools for WebSphere Process Server 47
- WebSphere InterChange Server 74
 - Adapter Artifacts Importer 208
 - administrative tasks 61
 - architectural overview 74
 - artifact mapping 127
 - artifact migration
 - collaborations 227
 - relationships 237
 - assessment phase 39
 - business objects 217
 - component mapping 126
 - development
 - team 47
 - tools 47
 - equivalent features in WebSphere Process Server 128
 - features 78
 - fine-grained administration capabilities 129
 - flow control 129
 - health monitoring 58
 - integration 9
 - JAR File import wizard 139
 - mapping rules 223–224
 - maps
 - artifact migration 222
 - migration 225
 - migration path evaluation 34
 - migration tools 180

- native capabilities 129
- quality of service 55
- recovery 80
- relationships 237
- repos_copy command 205
- repository file 297, 351, 438
- routing logic 535
- security levels 56
- separation of concerns 156
- Server Access Interface 248
- server artifact upgrade 123
- SIMM levels 2-5 21
- testing data synchronization example 348, 435
- toolkit 47
- tools for failed events 60
- usage patterns 23
- WebSphere Business Integration Adapters 240
- WebSphere JCAs 247
- WebSphere MQ binding 104
- WebSphere MQ configuration for migration scenarios 289
- WebSphere MQ data bindings 107
- WebSphere MQ JMS binding 103
- WebSphere Process Server 90
 - Access EJB support 250
 - administrative tasks 62
 - applications 52
 - architectural overview 90
 - artifact migration
 - bindings 248
 - business processes 228
 - maps 223
 - relationships 237
 - SDO 218
 - WebSphere Business Integration Adapters 242
 - as-is migration 295
 - benefits of upgrade 128
 - bindings 101
 - upgrade from WebSphere Business Integration Adapters 135
 - business graph map 226
 - business object map 226
 - component models 155
 - components to cluster 52
 - data bindings 104
 - development tools 47
 - enhanced error handling 512–513
 - development 519
 - testing and verification 530
 - equivalent features in WebSphere InterChange Server 128
 - error categories 60
 - failed to start adapter 568
 - features 97
 - health monitoring 59
 - integration 9
 - Java pid 560
 - limitations of upgrade 129
 - log and trace 561
 - messages in files 568
 - migration
 - path evaluation 34
 - routing logic 539
 - project experience 64
 - quality of service 55
 - reposMigrate utility 205
 - scenario deployment 560
 - separation of concerns 157
 - server artifacts 123
 - SIMM levels 2-7 21
 - startup error message 327
 - Unit Test Environment (UTE) 279
 - usage patterns 23
 - workaround for dynamic relationship 388, 467
- workbench 183
- wsadmin command 207
- WSDL (Web Services Description Language)
 - Port Type interfaces 186

X

- XML data handler 87



Migrating WebSphere InterChange Server and Adapters to WebSphere Process Server V6.2

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Migrating WebSphere InterChange Server and Adapters to WebSphere Process Server V6.2 and Best Practices

Shows how to migrate WebSphere InterChange Server and WBI Adapters

Helps plan the architectural usage patterns and migration

Includes migration tools, examples, and scenarios

IBM WebSphere Process Server is the next generation business process integration server that has evolved from proven business integration concepts, application server technologies, and the latest open standards. In this IBM Redbooks publication, we provide guidance for WebSphere InterChange Server users about how to migrate IBM WebSphere InterChange Server and WebSphere Business Integration Adapters to WebSphere Process Server. We discuss the critical concepts that are related to integration solution architecture, migration project planning, and the technical implementation approach. We provide a detailed discussion about the capabilities of the migration tools. In addition, we include various migration examples that show how to upgrade IBM WebSphere InterChange Server and WebSphere Business Integration Adapters to WebSphere Process Server and WebSphere Adapters. The four parts are: Part 1 introduces the high-level concepts required to comprehend the migration roadmap. Part 2 discusses relevant migration implementation concepts. Part 3 covers the standard migration tools to upgrade from WebSphere InterChange Server to WebSphere Process Server. Part 4 provides comprehensive examples to migrate end-to-end integration solutions.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks