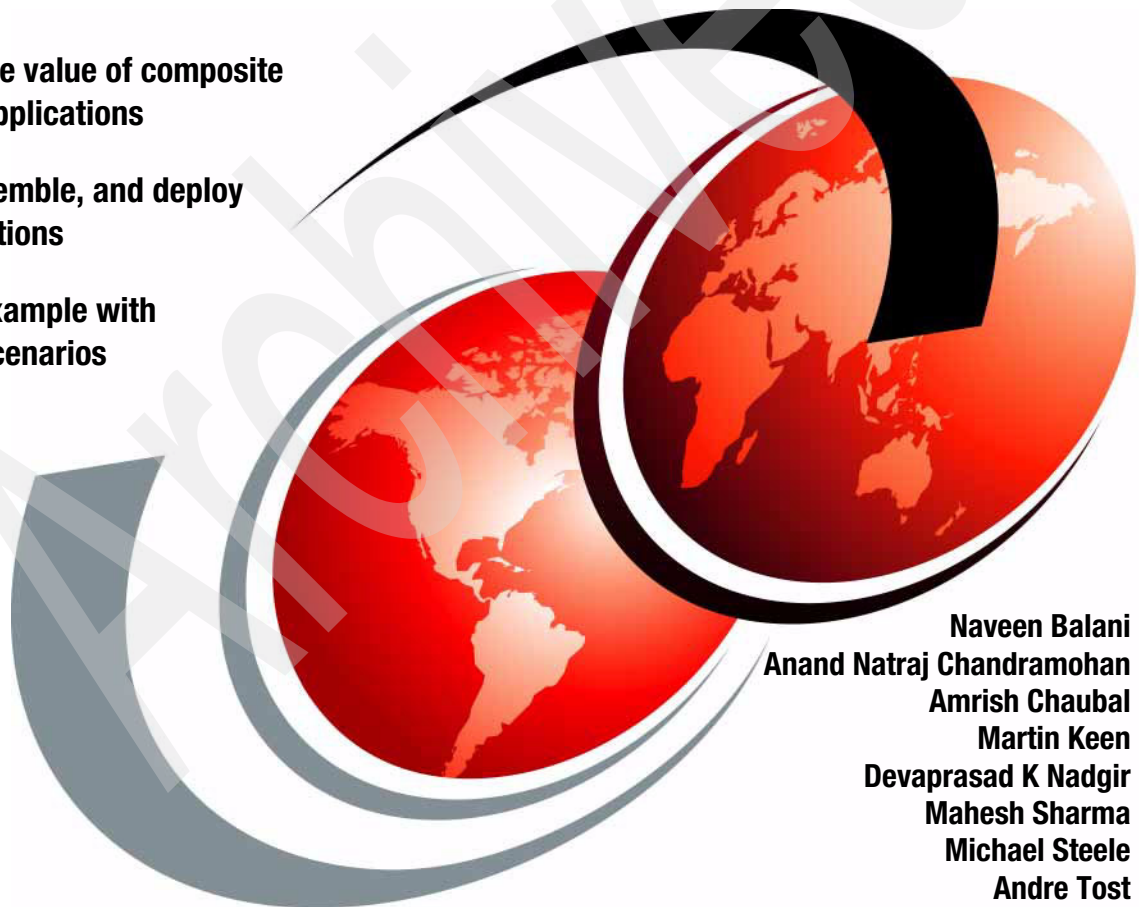


# Getting Started with IBM WebSphere Business Services Fabric V6.1

Discover the value of composite  
business applications

Model, assemble, and deploy  
Fabric solutions

Learn by example with  
practical scenarios



Naveen Balani  
Anand Natraj Chandramohan  
Amrish Chaubal  
Martin Keen  
Devaprasad K Nadgir  
Mahesh Sharma  
Michael Steele  
Andre Tost





International Technical Support Organization

**Getting Started with IBM WebSphere Business  
Services Fabric V6.1**

June 2008

Archived

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

## **First Edition (June 2008)**

This edition applies to WebSphere Business Services Fabric V6.1.

© Copyright International Business Machines Corporation 2008. 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

<b>Notices</b> .....	xi
Trademarks .....	xii
<b>Preface</b> .....	xiii
The team that wrote this book .....	xiii
Become a published author .....	xvi
Comments welcome .....	xvi
<b>Part 1. Introduction</b> .....	1
<b>Chapter 1. Welcome to this Redbooks publication</b> .....	3
1.1 An introduction to this book .....	4
1.2 How to read this publication .....	4
1.2.1 Part 1: Introduction .....	4
1.2.2 Part 2: Building Fabric solutions .....	5
1.2.3 Part 3: Integration .....	6
<b>Chapter 2. Business Process Management enabled by SOA</b> .....	7
2.1 Key business challenges .....	8
2.2 Portfolio of required disciplines .....	11
2.3 In closing .....	13
<b>Chapter 3. Introduction to WebSphere Business Services Fabric</b> .....	15
3.1 Fabric introduction .....	16
3.2 Key technologies, concepts and architecture .....	17
3.2.1 Business services .....	18
3.2.2 Composite business applications .....	23
3.2.3 Dynamic assembly of business services .....	26
3.2.4 Business services repository .....	32
3.2.5 Business policies .....	34
3.2.6 Service provisioning and enrollments .....	38
3.3 Fabric key benefits .....	40
3.3.1 Business process simplification .....	40
3.3.2 Business process decision logic consolidation .....	43
3.3.3 Provides a business services metadata model .....	43
3.3.4 Enables business service portfolio management .....	44
3.3.5 Provides business services visibility .....	45
3.3.6 Provides business services metadata governance .....	46

<b>Chapter 4. Technical overview of WebSphere Business Services Fabric</b>	<b>47</b>
4.1 Product overview	48
4.2 WebSphere Business Services Fabric product positioning	48
4.2.1 The reusable building blocks of composite business applications	49
4.2.2 WebSphere Business Services Fabric product components	50
4.2.3 Business Services Foundation Pack	53
4.2.4 Business Services Tools Pack	55
4.2.5 Industry content packs	56
4.2.6 Software development kit	57
4.3 Product component overview	58
4.3.1 Project setup, project development, and project deployment	58
4.3.2 Fabric administrative console	61
4.3.3 Composition Studio	67
4.3.4 Business Services Repository	69
4.3.5 Dynamic Assembler	71
4.4 Fabric policies	77
4.5 Business Services Fabric Modeling Tool	82
<b>Part 2. Building Fabric solutions</b>	<b>83</b>
<b>Chapter 5. Business scenario for this book</b>	<b>85</b>
5.1 Introduction	86
5.2 Business scenario overview	86
5.2.1 Use case scenario	86
5.2.2 AS-IS business scenario	89
5.3 Business challenges	95
5.4 IT challenges	96
5.5 Summary	98
<b>Chapter 6. Installing and configuring Fabric</b>	<b>99</b>
6.1 Overview of products	100
6.2 Typical deployment topology for the Business Services Foundation Pack	100
6.3 Development topology for the Business Services Tool Pack	102
6.4 Installing the pre-requisites	103
6.5 Installing the Business Services Foundation Pack	105
6.6 Verifying the Business Services Foundation Pack installation	111
6.6.1 Accessing the Fabric Tools Console	112
6.6.2 Troubleshooting your installation	121

6.7	Installing the Business Services Tool Pack . . . . .	126
6.8	Verifying the Business Services Tool Pack installation . . . . .	130
6.9	Installing Fabric on z/OS . . . . .	132
6.10	Known limitations . . . . .	133
6.11	Clustering and failover. . . . .	134
<b>Chapter 7. Overview of developing a Fabric solution . . . . .</b>		<b>137</b>
7.1	Introduction to IBM SOA Foundation life cycle . . . . .	138
7.2	Life cycle for developing composite business applications . . . . .	140
7.2.1	Model . . . . .	141
7.2.2	Assemble . . . . .	146
7.2.3	Deploy . . . . .	148
7.2.4	Manage . . . . .	150
7.2.5	Govern . . . . .	151
7.3	Conclusion. . . . .	151
<b>Chapter 8. Modeling Fabric solutions . . . . .</b>		<b>153</b>
8.1	Introduction . . . . .	154
8.1.1	Definition of modeling terms . . . . .	154
8.1.2	Steps in modeling a Fabric solution . . . . .	155
8.2	Identifying process variability points . . . . .	156
8.3	Identifying context information. . . . .	156
8.3.1	Roles and channels. . . . .	157
8.3.2	Content-based assertions . . . . .	157
8.4	Policy identification . . . . .	160
8.4.1	Policy definition . . . . .	160
8.5	Simplified process model . . . . .	161
8.5.1	Comparison with the core BPM approach. . . . .	161
8.5.2	Streamlined process in Fabric. . . . .	161
8.6	Fabric ontology core model. . . . .	162
8.6.1	Governance domain . . . . .	163
8.6.2	Business service domain . . . . .	163
8.6.3	Service consumer domain. . . . .	164
8.6.4	Service technical domain . . . . .	164
8.7	Modeling extensions . . . . .	165
8.7.1	Installing plug-ins and importing profiles . . . . .	165
8.7.2	Creating the business glossary model . . . . .	169
8.7.3	Creating the ontology model . . . . .	175
8.7.4	Glossary transformation . . . . .	181
8.7.5	Ontology modeling . . . . .	184
8.7.6	Ontology transformation . . . . .	189
8.7.7	Exporting the Fabric Content Archive . . . . .	192

<b>Chapter 9. Assembling Fabric solutions</b>	193
9.1 Assembly in WebSphere Integration Developer	194
9.2 Assembling a Fabric solution in WebSphere Integration Developer	196
9.2.1 Creating generalized WS-BPEL process	196
9.2.2 Making Fabric context information available to the Dynamic Assembler	198
9.2.3 Using the Dynamic Assembler SCA component	201
9.2.4 Using Dynamic Assembler extensions	206
9.2.5 Injecting context into the WS-BPEL process	214
9.3 Assembling in Composition Studio	219
9.3.1 Creating the Subscriber Model	221
9.4 Synchronizing WSRR with Fabric	223
9.4.1 Importing the WSDL in WSRR	224
9.4.2 Creating a WSRR project in Fabric	226
9.5 Loading the extended business model into Fabric	228
9.5.1 Creating an Ontology project	229
9.5.2 Creating namespaces	230
9.5.3 Importing the Ontology extension project	231
9.5.4 Creating a Fabric project	234
9.5.5 Configuring environments	239
9.5.6 Replicating the project in Composition Studio	242
9.5.7 Creating business context for your CBA	247
9.5.8 Adding capabilities to endpoints	263
9.5.9 Creating business policies	270
9.5.10 Using policy simulation	278
9.6 Summary	287
<b>Chapter 10. Deploying Fabric solutions</b>	289
10.1 Deploying the Fabric solution	290
10.2 Publishing the business service metadata	290
10.3 Deploying the SCA module	295
10.4 Enrollments and subscriptions management	297
10.5 Testing the SCA process	303
<b>Chapter 11. Migrating Fabric projects across environments</b>	313
11.1 Environment setup overview	314
11.2 Migrating the business process	315
11.2.1 Configuring environment variables	317
11.3 Promoting business service metadata	318
11.3.1 Exporting the metadata from UTE	318
11.3.2 Importing the business service metadata into the production environment	321

<b>Chapter 12. Adapting to changing business needs</b>	325
12.1 Business change	326
12.2 Change in the process	326
12.2.1 Without Fabric	327
12.2.2 With Fabric	327
12.3 Managing business change	328
12.4 Modeling the change	329
12.4.1 Identifying variability points	329
12.4.2 Ontology modeling	330
12.5 Assembling the change	334
12.5.1 Import the new onthology model into Fabric	334
12.5.2 Replicate the project in Composition Studio	334
12.5.3 Adding endpoints	334
12.5.4 Adding an assertion to the endpoints	339
12.5.5 Associate interfaces	344
12.5.6 Associating environments	345
12.5.7 Adding policies	348
12.5.8 Changing the Credit Check context specification	349
12.5.9 Running simulation	350
12.6 Deploying and testing	354
12.6.1 Deploying metadata	354
12.6.2 Testing the process	354
<b>Chapter 13. Best practices and troubleshooting</b>	357
13.1 Problem determination in Fabric Composition Studio	358
13.2 Problem determination in Fabric runtime	359
13.3 Problem determination in underlying WebSphere Process Server, WebSphere Integration Developer infrastructure	361
13.4 Common issues	363
13.5 General best practices and recommendations	364
<b>Chapter 14. Industry Content Packs</b>	369
14.1 Overview of Industry Content Pack	370
14.2 Industry Content Pack reference architecture	371
14.3 Types of assets	373
14.3.1 Industry Capability and Process Maps	373
14.3.2 Industry Business Services Templates	374
14.3.3 Industry Service Interfaces	375
14.3.4 Industry Business Glossary	375
14.3.5 Industry Common Services	376
14.3.6 Industry Business Object Model	377
14.3.7 Knowledge Assets	377

14.4 Banking Payments Content Pack .....	379
14.4.1 Leveraging the Banking Payments Content Pack .....	382
14.4.2 Reference implementation .....	389
14.5 Summary .....	399
<b>Part 3. Integration .....</b>	<b>401</b>
<b>Chapter 15. Integrating with WebSphere Service Registry and Repository</b>	
403	
15.1 Product overview .....	404
15.1.1 A registry and repository for SOA governance .....	404
15.1.2 Stages of service-oriented solution development .....	405
15.1.3 Technical product details .....	408
15.2 WebSphere Service Registry and Repository and the Fabric: three types of scenarios .....	415
15.2.1 Scenario 1: Deploy Fabric first and WebSphere Service Registry and Repository second .....	417
15.2.2 Scenario 2: Deploying WebSphere Service Registry and Repository first and Fabric second .....	420
15.2.3 Scenario 3: Deploying WebSphere Service Registry and Repository and Fabric simultaneously .....	423
15.3 Federation between WebSphere Service Registry and Repository and Fabric .....	424
15.3.1 Configuring a federation project .....	425
15.3.2 Verifying content federation .....	429
15.4 Using WebSphere Service Registry and Repository as part of a Fabric project - publishing content .....	432
15.4.1 Publishing an SCA integration module .....	433
15.4.2 Publishing using the Eclipse plug-in .....	436
15.4.3 Adding classifications .....	437
15.4.4 Adding artifacts to a life cycle .....	440
15.5 Using WebSphere Service Registry and Repository as part of a Fabric project - filtering federated content .....	442
15.5.1 Defining WebSphere Service Registry and Repository permissions for Fabric federation .....	443
15.5.2 Selective federation .....	446
15.6 Using WebSphere Service Registry and Repository as part of a Fabric project - finding content .....	449
15.6.1 Finding artifacts in WebSphere Service Registry and Repository .....	450
15.6.2 Finding artifacts in the Business Services Repository .....	454
15.6.3 Using both approaches together .....	458

<b>Chapter 16. Integrating with IBM Tivoli Composite Application Manager for SOA</b>	459
16.1 SOA management for composite business applications	460
16.2 Overview of IBM Tivoli Composite Application Manager for SOA	462
16.3 Integration between ITCAM for SOA and WebSphere Business Services Fabric	464
16.4 Installing and configuring ITCAM for SOA in ITSOBank	465
16.4.1 Infrastructure setup	465
16.4.2 Enabling data collection from WebSphere Business Services Fabric.	470
16.5 Data collection and monitoring for the ITSOBank Loan Process composite business application	472
16.6 Conclusion	479
<b>Chapter 17. Integrating with Lightweight Directory Access Protocol</b>	481
17.1 Introduction	482
17.2 Integrating Fabric with IBM Tivoli Directory Server	484
17.2.1 Foundation Pack administrator configuration	485
17.2.2 Configuring the federated user registry	487
17.2.3 Assigning a new Fabric administrator	492
17.3 Conclusion	499
<b>Part 4. Appendixes</b>	501
<b>Appendix A. Additional material</b>	503
Locating the Web material	503
How to use the Web material	503
<b>Abbreviations and acronyms</b>	505
<b>Related publications</b>	507
IBM Redbooks	507
How to get Redbooks	507
Help from IBM	508
<b>Index</b>	509

Archived



# 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®  
CICS®  
DataPower®  
DB2®  
developerWorks®

IBM®  
Rational®  
Redbooks®  
Redbooks (logo) ®  
Tivoli Enterprise Console®

Tivoli®  
WebSphere®  
z/OS®

The following terms are trademarks of other companies:

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

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

EJB, J2EE, J2ME, Java, JDBC, JVM, Streamline, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Microsoft, Visual Studio, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

# Preface

WebSphere® Business Services Fabric (Fabric) is a comprehensive service-oriented architecture (SOA) offering that is designed to extend the IBM® business process management platform to deliver flexible composite business applications.

In this IBM Redbooks® publication, we provide a complete overview of Fabric, from an architectural introduction, to an installation guide, and a step-by-step scenario that describes how to model, assemble, deploy, and manage composite business applications.

Part 1 of this book is an architectural and technical introduction to Fabric and related concepts.

In Part 2, we use a fictitious business scenario to show you how to build composite business applications in Fabric. We also provide extensive step-by-step instructions.

In Part 3, we discuss how to integrate Fabric with other IBM solutions and technologies, which includes WebSphere Service Registry and Repository, Tivoli® Composite Application Manager for SOA, and Lightweight Directory Access Protocol (LDAP).

## The team that wrote this book

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



*Figure 0-1 Team (left-to-right): Andre, Naveen, Amrish, Dev, Martin, Mahesh, and Michael*

**Naveen Balani** is an IT Architect with IBM India Software Labs. He leads the design and development activities for the WebSphere Business Service Fabric product. He likes to research upcoming technologies and is a regular contributor to IBM developerWorks® covering topics, such as Web services, ESB, JMS, SOA, architectures, open-source frameworks, semantic wWb, J2ME™, persuasive computing, the Spring series, AJAX, and various IBM products. He is also a coauthor of the Beginning Spring Framework 2 book.

**Anand Natraj Chandramohan** is an IT Architect in IBM India Software Labs. He has five years of experience in the Business Integration field. He has a degree in Computer Science and Engineering from the University of Madras. His areas of expertise include architecting, designing, and implementing end-to-end SOA and business process management (BPM)-based solutions using the Rational® and WebSphere portfolio. His areas of interests include SOA methods, SOA performance architecture, and tooling research.

**Amrish Chaubal** is an SOA IT Architect in the SOA Technology Practice of IBM India Software Labs and is responsible for anchoring and architecting WebSphere Business Service Fabric engagements with the IBM Business Partners in the Asia-Pacific region. He has 10 years of experience in the IT field. His area of expertise includes developing and architecting distributed enterprise applications.

**Martin Keen** is a Senior IT Specialist at the ITSO, Raleigh Center. He writes extensively about WebSphere products and SOA. He also teaches IBM classes worldwide about WebSphere, SOA, and enterprise service bus (ESB). Before joining the ITSO, Martin worked in the EMEA WebSphere Lab Services team in Hursley, United Kingdom (UK). Martin has a bachelor's degree in Computer Studies from Southampton Institute of Higher Education.

**Devaprasad K Nadgir** is a Solutions Architect in the IBM Software Group and works out of Bangalore, India. His current areas of focus involve incubation and delivery of service-oriented architecture solutions with IBM Business Partners and customers. Prior to this, he spent over eight years with IBM in various development roles. His areas of interest also include Linux® internals, Autonomic Computing, IBM JVM™ internals, on-demand Computing, and Architectural methods.

**Mahesh Sharma** is as an IT Architect in the IBM UK Software Group Services team. His areas of expertise include service-oriented architectures, methodologies, and solutions utilizing IBM SOA and ESB portfolios. His experience includes architecture, design, and implementation of business integration, middleware, and Web-based solutions, which he acquired across the globe having lived and worked in Europe, America, and Asia. He has a Masters degree in Computer Science from North Dakota State University, USA.

**Michael Steele** is a Solution Architect in the WebSphere Business Services Fabric Worldwide Technical Sales Team. He joined IBM with the IBM Webify acquisition. Prior to Webify, he worked as a consulting architect supporting large-scale client integrations, specializing in commercial rules-based solutions. He helped to build solutions on all major platforms in Healthcare, Property and Casualty Insurance, Manufacturing, and Governmental agencies. He has a Bachelors degree in Business and Computer Science from Kent State University.

**Andre Tost** is a Senior Technical Staff Member in the IBM Software Services for WebSphere organization, where he helps the IBM customers to establish service-oriented architectures. His special focus is on Web services and Enterprise Service Bus technology. Before his current assignment, he spent ten years in various partner enablement, development, and architecture roles in IBM software development, most recently for the WebSphere Business Development group. Originally from Germany, he now lives and works in Rochester, Minnesota.

Thanks to the following people for their contributions to this project:

Larry Deppa  
WebSphere Business Services Fabric Offering Team Leader

Laura Olson  
Certified Consulting IT Specialist, IBM Worldwide WebSphere Technical Sales and Competitive Analysis Team

Robert Golladay  
Business Unit Executive, WebSphere Business Services Fabric

Kimberly Thomas  
Office of the Business Unit Executive, WebSphere Business Services Fabric

Chris Walk  
Consulting IT Specialist, Worldwide WebSphere Technical Sales

Ron Todd  
Solution Architect, WebSphere Business Services Fabric

Stephen Gibney  
IT Specialist, Software Services for WebSphere

Giovanni Vigorelli  
Advisory IT Specialist, Technical Sales for WebSphere

## Become a published author

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

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review Redbooks form found at:  
[ibm.com/redbooks](http://ibm.com/redbooks)
- ▶ Send your comments in an e-mail to:  
[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)
- ▶ Mail your comments to:  
IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400



# Part 1

## Introduction

Archived





# Welcome to this Redbooks publication

In this chapter, we provide guidance on how to get the most out of this book. It contains the following sections:

- ▶ 1.1, “An introduction to this book” on page 4
- ▶ 1.2, “How to read this publication” on page 4

## 1.1 An introduction to this book

A warm welcome to this IBM Redbooks publication from the team that wrote its content. We gathered for five intensive weeks in Raleigh, North Carolina, USA, to create this book and continued to work together remotely through the review cycle. We hope that you find this book to be an insightful and useful read.

The motivation in writing this book was to show how you can use WebSphere Business Services Fabric (Fabric) V6.1 to build advanced SOA solutions through the assembly of composite business applications. Whether Fabric is completely new to you, or you already have some hands-on experience with it, you should find something of interest in this book.

To help demonstrate the capabilities of Fabric, we provide a vast array of step-by-step examples. We based our examples around a fictitious organization called *ITSOBank*. The examples in this book describe a Vehicle Loan Processing System scenario. The scenario models the Vehicle Loan Processing System, identifies the points-of-variability in the process, and then shows how Fabric solutions enable ITSOBank to build a dynamic and flexible solution.

We hope you can use this book as a starting point for building your own composite business applications for deployment to Fabric.

## 1.2 How to read this publication

In this section, we describe how we structured this book and provide guidance on which chapters you should read.

### 1.2.1 Part 1: Introduction

In Part 1, we provide an introduction to Fabric and related concepts. Part 1 contains the following chapters:

- ▶ Chapter 1, “Welcome to this Redbooks publication” on page 3  
This is the chapter that you are reading now.
- ▶ Chapter 2, “Business Process Management enabled by SOA” on page 7  
In this chapter, we describe the current environment and pressures that organizations experience as they strive to align their business objectives and goals with existing IT capabilities.

- ▶ Chapter 3, “Introduction to WebSphere Business Services Fabric” on page 15  
In this chapter, we provide a definition of the key technologies, concepts, and architectures that relate to Fabric. If you are new to Fabric, be sure to read this chapter.
- ▶ Chapter 4, “Technical overview of WebSphere Business Services Fabric” on page 47  
This chapter begins with an overview of the Fabric base products followed by details on each of the respective modules. We also discuss Fabric Dynamic Assembler and Fabric policies.

## 1.2.2 Part 2: Building Fabric solutions

In Part 2, we describe the fictitious ITSOBank Vehicle Loan Processing System scenario, and we show step-by-step how to implement this solution in Fabric. Read this section to gain hands-on knowledge of Fabric. Part 2 contains the following chapters:

- ▶ Chapter 5, “Business scenario for this book” on page 85  
In this chapter, we introduce the ITSOBank Vehicle Loan Processing System solution.
- ▶ Chapter 6, “Installing and configuring Fabric” on page 99  
In this chapter, we describe how to install and verify the Business Services Foundation Pack and Business Services Tool Pack.
- ▶ Chapter 7, “Overview of developing a Fabric solution” on page 137  
In this chapter, we give you an overview of what it takes to develop a composite business application using Fabric.
- ▶ Chapter 8, “Modeling Fabric solutions” on page 153  
In this chapter, we discuss modeling a composite business application and describe the use of the Fabric Modeling Tool plug-in for Rational Software Architect.
- ▶ Chapter 9, “Assembling Fabric solutions” on page 193  
In this chapter, we describe how to assemble a composite business application using WebSphere Integration Developer and Composition Studio.
- ▶ Chapter 10, “Deploying Fabric solutions” on page 289  
In this chapter, we describe how to deploy and test a composite business application in a unit test environment.

- ▶ Chapter 11, “Migrating Fabric projects across environments” on page 313  
In this chapter, we describe how to deploy a composite business application to a production environment.
- ▶ Chapter 12, “Adapting to changing business needs” on page 325  
In this chapter, we show you how to adapt to a change in the ITSOBank business scenario. We highlight the Fabric solution strengths in adapting to changing business needs.
- ▶ Chapter 13, “Best practices and troubleshooting” on page 357  
In this chapter, we describe the various best practices while developing, deploying, managing, and governing a Fabric solution.
- ▶ Chapter 14, “Industry Content Packs” on page 369  
In this chapter, we introduce the Fabric Industry Content Packs and show you how to leverage them to accelerate composite business application delivery.

### 1.2.3 Part 3: Integration

In Part 3, we show you how to integrate Fabric with other IBM products and technologies. Part 3 contains the following chapters:

- ▶ Chapter 15, “Integrating with WebSphere Service Registry and Repository” on page 403  
In this chapter, we describe how WebSphere Service Registry and Repository (WSRR) enables the publication, retrieval, and management of artifacts that are related to the development of services. We also describe how WSRR can be federated with Fabric.
- ▶ Chapter 16, “Integrating with IBM Tivoli Composite Application Manager for SOA” on page 459  
In this chapter, we introduce some of the basics of SOA management and show you how to integrate Fabric with IBM Tivoli Composite Application Manager for SOA.
- ▶ Chapter 17, “Integrating with Lightweight Directory Access Protocol” on page 481  
In this chapter, we discuss the integration of Fabric with federated user registries using the IBM WebSphere Virtual Member Manager. We focus on how you can integrate a Lightweight Directory Access Protocol (LDAP) directory, such as IBM Tivoli Directory Server, with Fabric to configure various types of users and their associated roles and business services.

# **Business Process Management enabled by SOA**

In this chapter, we describe the current environment and pressures that organizations experience as they strive to align their business objectives and goals with existing IT capabilities. With this consideration, we discuss utilizing existing assets as building blocks to solutions where the assets are realized as business services, combined together with orchestration and business policies to deliver composite business applications. We also discuss the benefits of technology that can help realize that vision.

## 2.1 Key business challenges

The business landscape is global and rapidly changing, and there are many forms of marketplace uncertainty. The impact from this uncertainty to business will only increase in the future, and there are many who believe these new demands are a permanent fixture. These challenges take on many forms in the marketplace, which includes global competition, macro-economic changes, industry consolidation through mergers and acquisitions, new regulatory requirements, and shifting consumer demands.

Executive concerns become focused on issues of growth and outmaneuvering their competition, optimizing business process to lower costs, and improving quality. So, what is the bottom line? Businesses are being forced to change in fundamental ways—structurally, operationally, and culturally—in response to the imperatives of globalization and new technology.

In any business scenario, you want to turn the situation to your advantage. As it turns out, underlying business needs are usually the same regardless of the business climate, and only the relative priorities shift. To address those core business needs, you have to be able to rapidly change and innovate.

Businesses today want to remain competitive and want to be able to change at the pace of business. To enable this, business leadership expects business processes to accommodate their business goals and objectives, which means that the process and founding technology must:

- ▶ Support innovative business models and new differentiated products and services.
- ▶ Change rapidly and continuously optimize operational capabilities.
- ▶ Provide a real-time operational view with the ability to intervene fast.

However, those same business leaders struggle because innovation stalls when it is difficult to understand differentiating processes or capabilities, and delivering rapid successive change can be difficult when working with rigid and inconsistent processes.

You can address many of today's business challenges with Business Process Management (BPM) enabled by service-oriented architecture (SOA).

Today, organizations are seeing a variety of benefits by automating and managing key business processes, such as:

- ▶ Improved business agility

An insurance company that began selling their business insurance products through multiple channels and deployed in five months.

- ▶ Streamline™ processes and reduce costs  
A health insurance company provides real-time claims corrections to physicians, clinics, and doctors.
- ▶ Improved customer service  
A retailer allows customers to purchase items online and pick the online purchase up at local stores.
- ▶ Improved worker productivity  
An insurance company automated underwriting with approvals in minutes, not days.
- ▶ Enhanced customer and partner collaboration  
A telecommunications company provides real-time activation of bundled service plans.
- ▶ Empowered business users  
A local government agency integrates case management for social services.

BPM enabled by SOA can provide tremendous benefits to your business. However, it starts with a vision for the organization's business process that unleashes its potential, for instance, let us apply BPM enabled by SOA capabilities to a specific process example and look at the resulting benefits. Let us start by understanding a bank account opening process and some potential shortfalls, as shown in Figure 2-1.

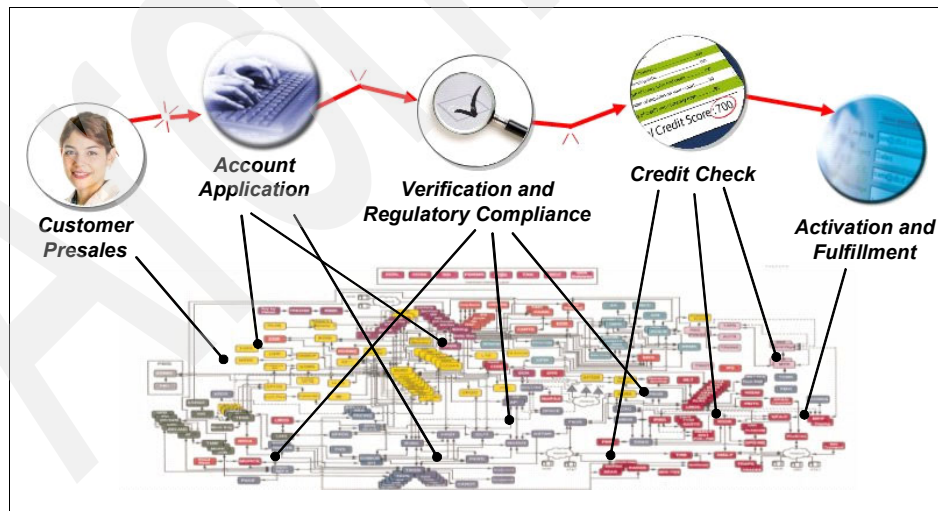


Figure 2-1 Banking account activation process

Currently the account opening process:

- ▶ Has different processes for similar products and services.
- ▶ Is complex and fragmented across multiple different systems.
- ▶ Has multiple back end IT systems for different products and channels
- ▶ Requires several days to complete the process
- ▶ Has several manual processing steps
- ▶ Is difficult and costly to modify
- ▶ Is expensive to maintain and support.

The bank's existing IT environment is very complex. Over many years, their IT systems have grown in size, sophistication, and complexity. Most large organizations have a significant investment in heritage systems and are not quick to replace them. Yet organizations need to expose these core systems and business processes out to a variety of constituents, over multiple communication channels and at differentiated service levels.

However, these systems are not easily modified. Business logic and functionality is embedded deep into the applications and not easily visible, modified, or understood. As a result, some previous solution patterns relied on removal and replacement of older systems or placement of newer systems beside existing ones. Additionally, a growing share of the IT budget is allocated to maintaining those systems instead of new solutions, and the lack of IT infrastructure flexibility results in lost business opportunities and other negative consequences.

Stepping out of this pattern is critical to achieving results today and also to preparing for tomorrow's challenges. The key is to embrace rather than postpone business process evolution, improvement, and change. An important enabler to achieving these results is to realize existing assets as business services and combine them with business-level policy to create composite business applications, which enables an environment that promotes quick understanding and rapid change. The objective is to make processes nimble, agile, and flexible to the business users and the IT resources that use and deliver them, while also providing additional benefits, such as:

- ▶ Creating opportunity for businesses to directly effect process behavior.  
Business users can enact rapid process change using policies instead of code.



- ▶ Unlocking siloed business functionality.  
Expose functionality from your current IT systems as reusable building blocks that you can dynamically change for different business scenarios.
- ▶ Innovating with confidence.  
Creating single uniform processes that leverage disparate systems and multiple delivery channels, while reliably addressing current business requirements.

Applying the ideas of business services, which leverage existing core assets and combine them together as flexible policy-driven processes can result in significant improvements when it comes to aligning business and IT. A BPM-enabled SOA allows us to:

- ▶ Build on existing IT assets
- ▶ Combine disparate pieces into reusable components
- ▶ Design processes with change in mind
- ▶ Customize core processes for multiple business scenarios, instead of deploying parallel or duplicate processes.

## 2.2 Portfolio of required disciplines

It also takes a variety of disciplines to deliver on the business objective of SOA, as shown in Figure 2-2 on page 12.

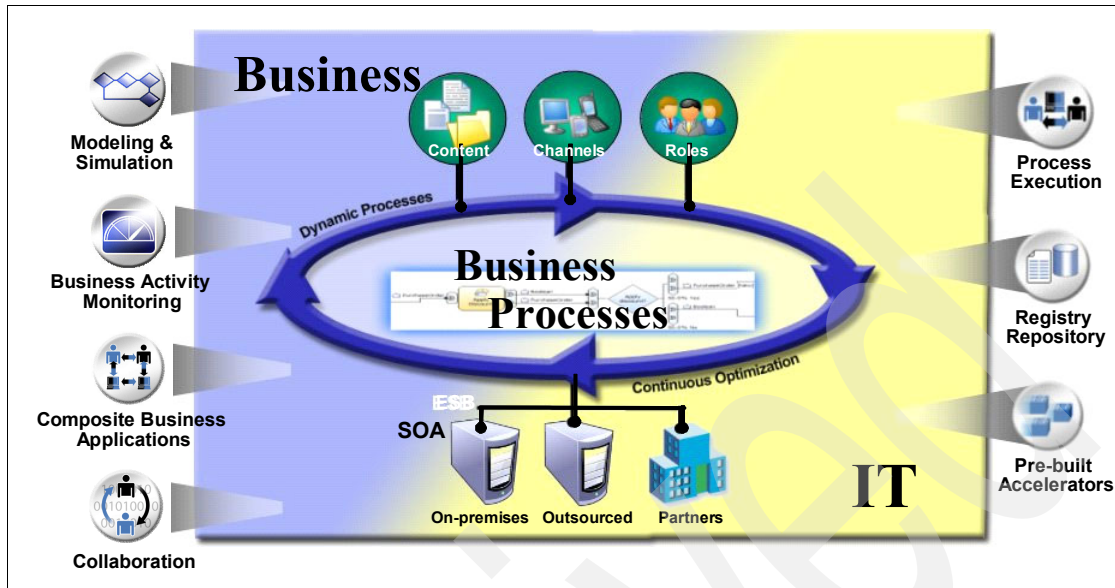


Figure 2-2 Portfolio of the disciplines that are required to deliver BPM enabled SOA

BPM is a discipline that combines software capabilities and business expertise to accelerate process improvement and to facilitate business innovation. BPM enabled by SOA provides a flexible architectural style in support of efficient process change and rapid process deployment. Of course this is what business leaders are striving for and what they need from their supporting IT systems. We need to be able to affect the behavior of a business process, based on business policy, without having to recode or remodel a process simply because we want a different behavior from it.

BPM governs an organization's cross-functional, core-business processes. It achieves strategic business objectives by directing the deployment of resources from across the organization into efficient processes that create customer value. This focus on driving overall bottom line success by integrating business functions and optimizing core work differentiates BPM from traditional functional-management disciplines. In addition, intrinsic to BPM is the principle of continuous improvement that perpetually increases value generation and sustains market competitiveness of the organization.

However, it is not just about improvement and optimization. BPM enabled by SOA is also about how quickly you can identify the parts of the business that drive change and how fast you can implement and deploy those changes to make improvements happen. It is critical to not only make the changes one time, but to be prepared for the inevitable future changes that every organization faces.

## 2.3 In closing

BPM enabled by SOA is an important enabler for achieving our business goals and objectives, and the business service is a key embracing concept. By abstracting business information, such as business context, business policies, and operational capabilities, as business services, and then using these business services as the building blocks for business processes, we can achieve greater business agility and responsiveness from these processes. Enabling this capability is a key benefit and capability of WebSphere Business Services Fabric.



# Introduction to WebSphere Business Services Fabric

WebSphere Business Services Fabric (Fabric) is a comprehensive service-oriented architecture (SOA) offering that is designed to extend the IBM Business Process Management (BPM) platform to deliver flexible composite business applications (CBA). We begin this chapter with a brief introduction to Fabric and continue to describe core technologies and concepts of Fabric. We also discuss the value and benefits that are obtained when Fabric is part of a solution.

This chapter contains the following sections:

- ▶ 3.1, “Fabric introduction” on page 16
- ▶ 3.2, “Key technologies, concepts and architecture” on page 17
- ▶ 3.3, “Fabric key benefits” on page 40

## 3.1 Fabric introduction

Fabric is a comprehensive SOA offering that delivers dynamic BPM capabilities to assemble and manage composite business applications. With Fabric, organizations can assemble business-level services into extended, cross-enterprise business processes and solutions. These solutions are dynamically delivered based on the business context of the service request.

As organizations adopt SOA as a way to improve business agility, drive innovation, and make the most of their IT investments, Fabric developed composite business applications can help the organizations gain improved flexibility by leveraging distributed, loosely coupled business-level services. These services are exposed from established systems, packaged applications, outsourced service providers, custom applications, and other third-party IT assets.

Composite business applications derive their functionality from an existing set of enterprise systems and applications. Through Fabric, these applications are represented as a collection of business services that deliver a prescribed business outcome.

Composite business applications created with Fabric provide the ability to:

- ▶ Transform rigid business processes into more agile solutions by flexibly guiding their behavior using business-level policies around subscribers, business services, and service agreements.
- ▶ Deliver the appropriate functionality to service consumers based on the context, content, and contract of the service request.
- ▶ Incrementally transform core business processes to be more efficient with reduced manual processing.
- ▶ Rapidly enable multichannel service delivery, providing higher service levels and lower customer support costs.
- ▶ Realize cost savings through significant IT asset reuse for new composite business applications.
- ▶ Deploy new products quickly with automated service entitlement capabilities to increase scalability and lower support costs.

Fabric also extends the IBM Business Process Management Platform at multiple levels:

- ▶ Extend IBM WebSphere Process Server to provide runtime and manage-time capabilities for business services.

- ▶ Extend IBM WebSphere Integration Developer to provide the tooling necessary to assemble business services.
- ▶ Integrate with WebSphere Services Registry and Repository to source technical Web services metadata that is used in the assembly of business services.

Fabric provides quality-of-service and information architecture enhancements, such as:

- ▶ Flexibility to change processes and service execution behavior across multiple business processes and disparate IT systems.
- ▶ Policy-driven business services to provide customized business functionality based on changing business contexts.
- ▶ Accelerated process change and easier ongoing maintenance with business level-policies that are managed from a central location.

Delivering these capabilities rests upon a comprehensive set of technologies and capabilities, which includes:

- ▶ Graphical design tooling that provides the ability to compose business services and to specify their relationships and associated metadata.
- ▶ A metadata repository that models and stores business-level metadata.
- ▶ A run-time engine that can provide real-time guidance to business services that are dynamically based on the associated metadata and business context.
- ▶ Governance tooling to manage the life cycle of the business level metadata.
- ▶ Service entitlements enabled using service subscriptions and the associated administrative consoles.
- ▶ Visibility through monitoring and understanding the deployed business services.

## 3.2 Key technologies, concepts and architecture

A key advantage to delivering BPM-enabled SOA applications with Fabric is the notion of achieving process flexibility and process agility in an easy and painless manner. Fabric provides the key concepts and technology to make this a reality.

In this section, we discuss those core Fabric concepts and technologies, including business services, composite business applications, dynamic assembly of business services, business services repository, business policies, and service entitlements.

### 3.2.1 Business services

The core building blocks of composite business applications are *business services*. Business services are also considered coarse grain Web services that represent business functions, transactions, or processes that are made available over an internal or external network. A business service represents a discrete business function and provides an abstract representation for a designer of composite business applications, while also providing a separation of concern between the business function and the implementation that supports it. A business service is not a technical service by definition and use. When you consider a technical service, such as say a Web-service, obvious characteristics are present, such as the operations that are provided, the port types that are supported, and binding information. In contrast, a business service addresses the remainder of the information that is necessary to place the service in context and apply it correctly, such as service availability, user entitlements, lines of business supported, and its relationship to other services.

A business service presents a bridge of understanding between business and IT, making it possible for business and IT to:

- ▶ Establish a consistent model for services so that the service is understood equally by both.
- ▶ Capture business constraints around service delivery.
- ▶ Map core business functions to supporting IT capabilities.
- ▶ Simplify design, implementation, and maintenance of business processes.
- ▶ Govern assembly of services into composite business applications.

We can describe business services as the intersection of three dimensions:

- ▶ Who uses them
- ▶ How they are made available
- ▶ What they provide in terms of functionality and capability

A business service is the encapsulation of these characteristics, as illustrated in Figure 3-1 on page 19.



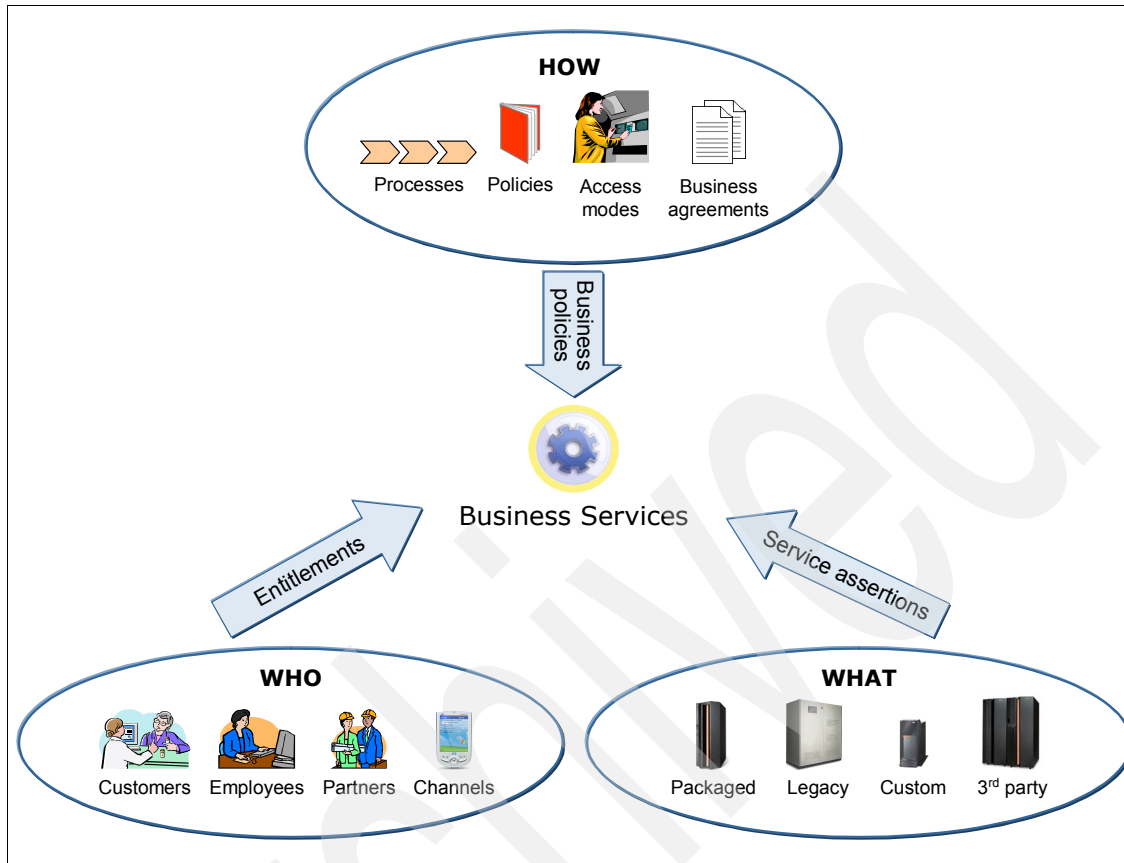


Figure 3-1 Dimensions of a business service

A business service has the following characteristics:

- ▶ Designed at the business level to represent a discrete business function.
- ▶ Uses business policies and metadata to describe and explain service characteristics, such as costs, availability, supported roles, supported channels, standards, and operational capabilities.
- ▶ Leverages industry models in support of interoperability.
- ▶ Can be provisioned through multiple communication channels.
- ▶ Is generally derived from disparate IT resources.
- ▶ Leverages interface and industry standards.

A business service is generally represented by three primary elements:

- ▶ **Business Metadata:**

Business policy and business service assertion information that is used to describe the service and its capabilities or restrictions. These directives could exist in an automated state, or they could be non-automated (white space) areas given that most processes are still highly fragmented and broken or manual in many areas. Creating the metadata that surrounds the business service gives unique capability to automate these white spaces.

- ▶ **Canonical Data Model:**

A standardized representation of data that generally leverages an industry or an organizational standard.

- ▶ **Business Function:**

A business service implementation might range in complexity from simple atomic services to composite services that embody business processes.

Leveraging a standardized representation of data for interfaces and descriptions are critical and important to achieving the value that can be associated with a business service. A business service is intended to be loosely associated with its prescribed implementations and easily reused.

Typical real world implementations do not share the same interface specifications very often and as such the business service becomes a point-of-consolidation and representation for these services. Numerous technologies can be embraced to normalize the implementation interfaces and should be implemented as required. As we discuss business services, the notion of a consistent and semantically similar interface are considered a given.

## **Anatomy of a business service**

We discussed the basic components and characteristics of a business service. Next, we discuss the anatomy of a business service —how these components and characteristics combine to deliver an example business services. We examine a Credit Check business service, as shown in Figure 3-2 on page 21.

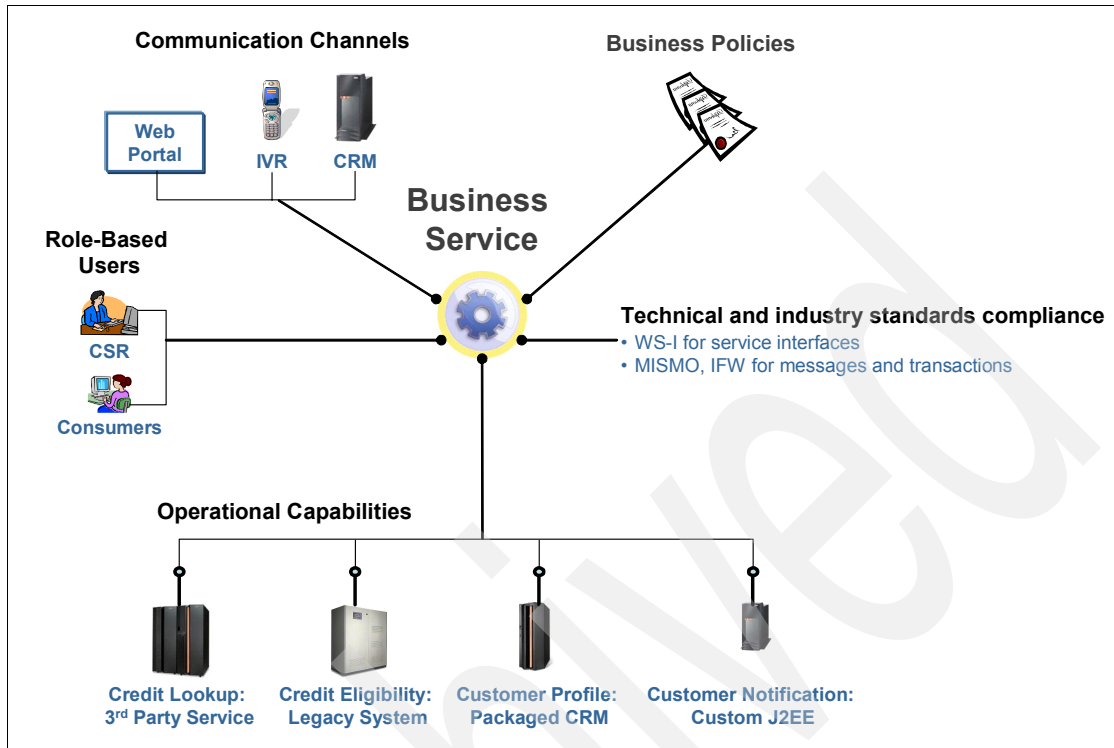


Figure 3-2 Credit Score business service

Credit Check is an example of a commonly used service. It is fairly common and relates to most all consumers. A credit check is performed anytime a consumer requests a large or long term transaction, such as applying for a credit card, purchasing a new car, and perhaps establishing a new cellular phone service.

The Credit Check service can be complex. Generally, no one piece of code or programming logic delivers the final result. Multiple factors come into play during a credit check process. A credit score must be requested and returned. Typically this is performed by third-party providers or by an internal data-store look-up where historical credit score results are retained. This score is often only one piece of the final determination because the score is combined with other factors, such as age, employment, and residential history, to determine and qualify the applicants risk. After the credit risk is determined, the customer record is updated, and a notification is made.

For our example, we examine the Credit Check business service that focuses on the discrete business function of Credit-Score only. Arbitrarily, an underwriting or rating system could be combined with the credit score to determine credit-risk,

which creates a more complex and composite business function. Ultimately this becomes a design consideration when creating the business services.

The business service has a number of dimensions that are easily visible, such as:

- ▶ Support for multiple roles and users.
- ▶ Support for multiple consumption channels.
- ▶ Leveraging business and technical industry standards, both for service descriptions and service message interactions.
- ▶ Multiple implementations.
- ▶ Metadata service descriptions as policies and assertions.

Other aspects of the business service that are not as easily described are how these dimensions come together when a business service is applied, such as:

- ▶ The service supports multiple user roles. Are they all treated the same? In most all cases the answer is no.
- ▶ The service has a diverse set of implementations. Which one should be used, why, and under what circumstances?
- ▶ The business service supports multiple channels. Does this mean that all channels are treated as equals? Maybe, but most likely some adjustments are necessary. How is this recognized and responded to?
- ▶ What metadata descriptions, labels, and attributes are appropriate and were defined? Are they specific to the industry or line-of-business? Are they common and standardized for broad and unambiguous understanding?
- ▶ Where and how are these dimensions related to the business service? Or to each other? If one changes, is there a cascading effect? How is this realized? Who realizes it?
- ▶ Where is this business service realized and managed?

As you read further, the questions that we just posed are answered. You will better understand how Fabric plays a unique and capable role in representing, defining, and realizing a business service.

However, these questions are very important and are best stated now and heard now, so that you can gain a comprehensive understanding of a business service. It is important to realize that a business service is not just a static list of information that describes some discrete business functionality. Rather it represents the dynamics and understanding that are put into play when this business service is called upon to deliver the prescribed business functionality.

In summary, what started as a simple description of a discrete business service now becomes the cornerstone building blocks for your composite business applications. As the building blocks of composite business applications, the business service must be reliable, reusable, and well understood. Considering the attributes and characteristics of a business service, how they are used and applied is critical to understanding the need for business services and will also assist you as you design and consider them in your solutions.

### 3.2.2 Composite business applications

A composite business application (CBA) is a collection of related and integrated business services that provide a specific business solution and support multiple business processes that are built on SOA. A business service is a building block and is designed and constructed for reusability, whereas a CBA is the derivative of a combination of business services.

At first glance, this may appear as just a simple collection of business services; however, a CBA is a broader more comprehensive view of the business solution. The CBA is more specialized to the business solution and drives the process, channels, roles, and business object model of the overall business solution. The CBA leverages the business services to deliver the final solution. A CBA also shares many of the same characteristics that are associated with business services, such as:

- ▶ Designed at business level to deliver a specific business outcome.
- ▶ Uses business policies and metadata to describe and explain service and solution characteristics, such as costs, availability, supported roles, supported channels, standards, and operational capabilities.
- ▶ Further increases straight-through processing by automating white spaces in the process.
- ▶ Leverages industry models in support of interoperability and common understanding.
- ▶ Generally supports multiple consumption channels (Web, B2B, and so on).
- ▶ Derived from multiple business services.

Embracing a business services platform, such as Fabric, provides designers with the ability to create solutions without tremendous amounts of integration. Business services and processes share more meaning through semantic glossaries, service interface specifications, and platform neutrality.

Figure 3-3 articulates the relationship of a CBA to business services.

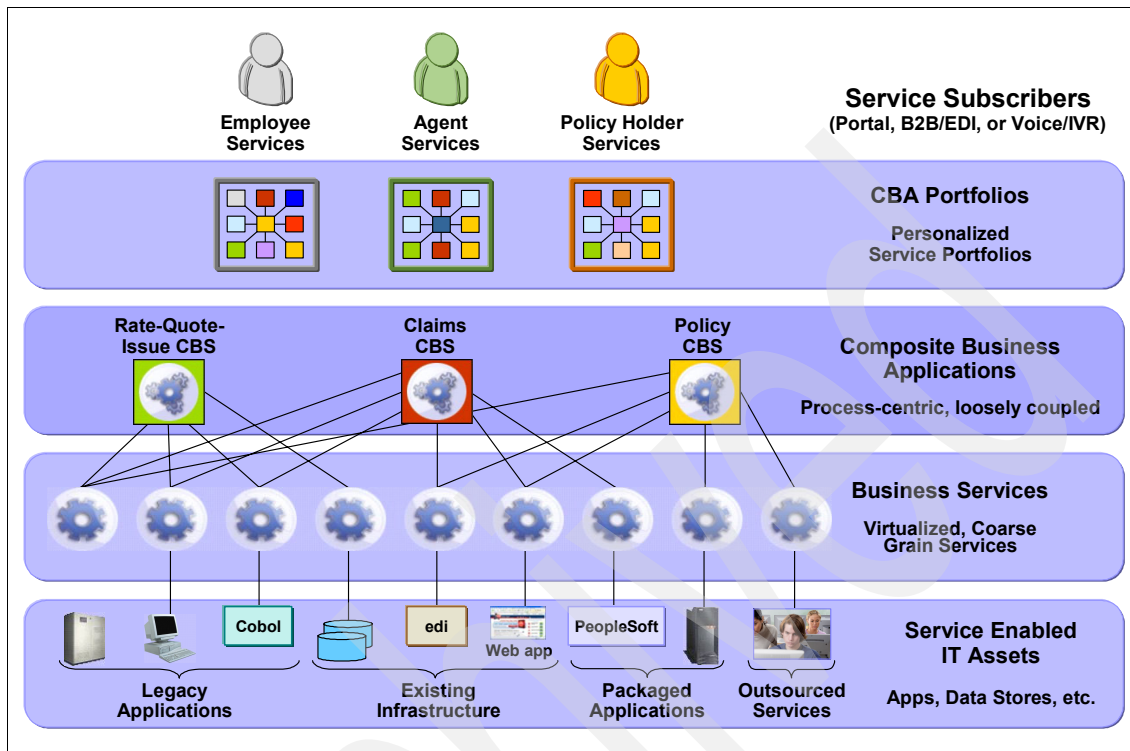


Figure 3-3 Composite business applications

Looking at Figure 3-3 and viewing the various layers, it becomes noticeable that business services are close to the back office IT infrastructure, while CBAs are closer to the users/subscribers and consumption channels.

It is clearly noted that business services leverage operational capabilities and that CBAs are intended to deliver a specific business solution. Perspective and use makes a CBA different from a business service.

The CBA is not generally intended to be reused. A CBA is also much more specialized, particularly when noting the solution data model, channels supported, roles, and organizations to which it is entitled. Attributes and characteristics generally associated with a CBA include:

- ▶ Consumer channels
- ▶ Business processes
- ▶ Business policies
- ▶ Business object model/data model

- Metadata model and extensions
- A more coarse grained business solution process model

Let us consider our previous credit check business service example, and now include that business service in a loan application processing CBA, as shown in Figure 3-4.

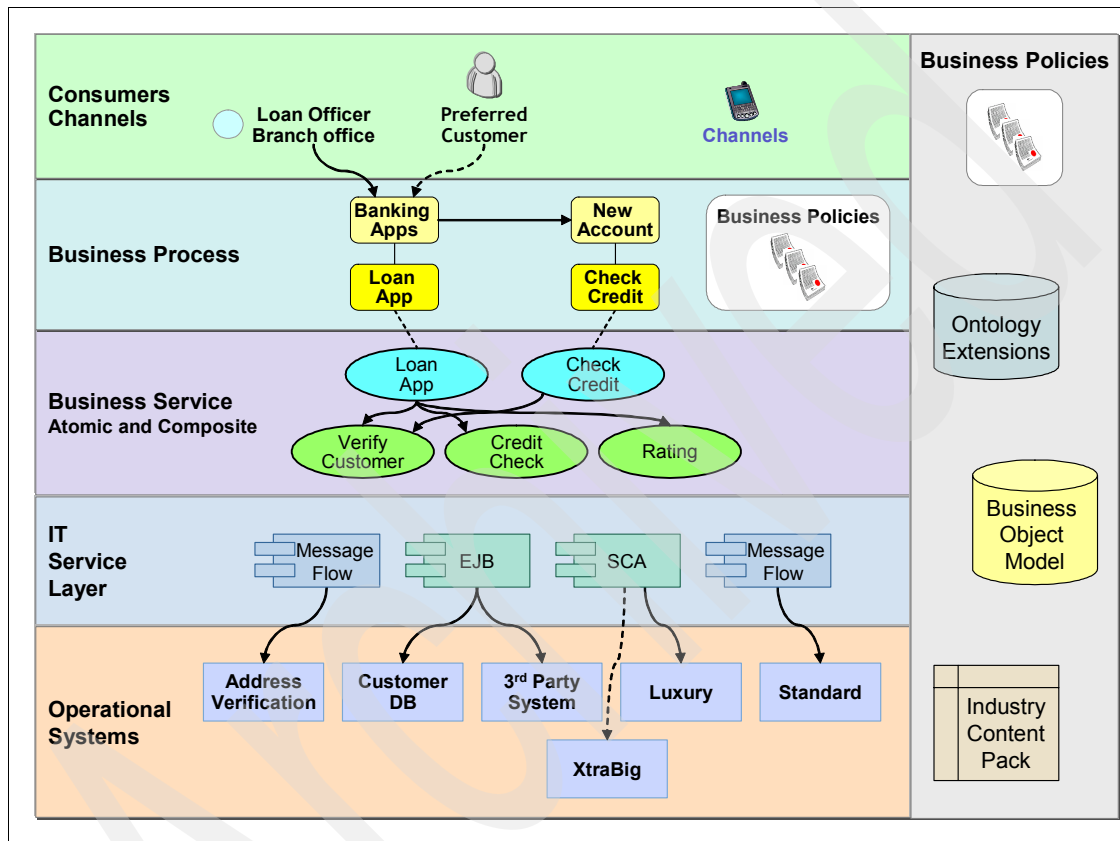


Figure 3-4 Composite Business Application example

In Figure 3-4, the CBA is broader in scope and in the number of components. Our Loan Application process includes multiple business services, such as the Credit Check Service, a Customer Verification Service, a Rating Service, and a Loan Processing Service.

In addition to the business services, the CBA includes a data model and extensions to the metadata model (ontology extensions, which we discuss later). User interfaces are generally part of the CBA; however, many times a CBA is faceless, so that it can support many consumption channels, which increases its

versatility and value to the organizations. Much like a business service, additional aspects of the CBA are not as easily recognized but must be accommodated, such as:

- ▶ The CBA supports multiple channels. Does this mean that all channels are treated as equals? Maybe, but most likely some adjustment is necessary. How is this recognized and responded to?
- ▶ What metadata description labels and attributes are appropriate and are defined? Are they specific to the industry or line-of-business? Are they common and standardized for broad and unambiguous understanding?
- ▶ A CBA consumes multiple business services, so how are the business services and CBAs cross referenced? This aspect becomes critical as we consider reuse of business services, particularly if demand and capacity implications are considered.

In summary, a CBA is a collection of business services that are organized and composed together as a means to deliver a very specific business outcome. A CBA also provides for consolidation of business level usage information that creates distinct opportunities for discovery and change over the life cycle of the application. With business services, a CBA also represents the dynamics and understanding that are put into play when this application is called upon.

### 3.2.3 Dynamic assembly of business services

Developing and delivering CBAs and business services with Fabric gives you the ability to realize very flexible and agile solutions, so that you can create solutions that you can manage at the pace of business. The key to making this concept both plausible and possible is the notion of dynamic assembly of services during CBA and business service execution.

CBAs and business services require descriptive metadata because it provides a comprehensive description of a business service and CBA. Additionally, we discussed that policy or business knowledge might be represented in the metadata. The metadata is very meaningful and necessary because it creates an understanding of the business service and CBA that is insightful of both business expectations and technical considerations. Refer to Figure 3-5 on page 27.



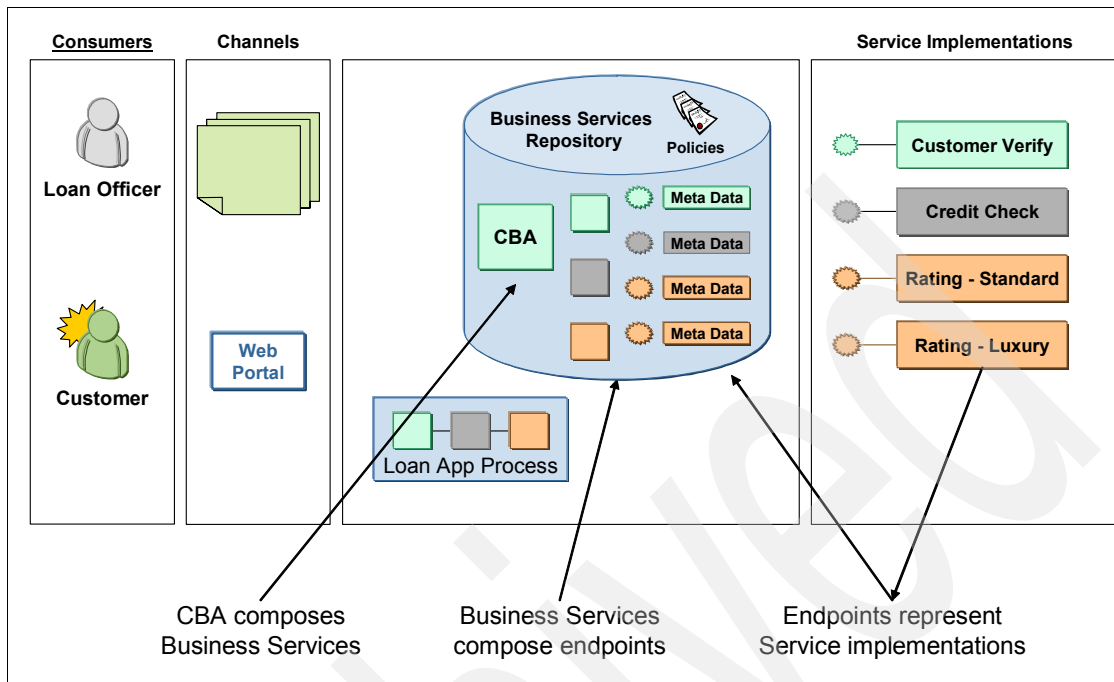


Figure 3-5 Dynamic assembly of business services - 1

Next, we leverage this understanding at execution time to influence the CBA and business service implementations, which ultimately allows the solution to be composed in declarative terms and delivered at the time of execution, specifically to the known circumstances. This is interesting, but programming logic can also deliver the same thing. Yes this is true; however, for this concept we are not required to create programming logic by writing procedural code. Instead, we leverage the declarative knowledge of the metadata and a runtime capability that can digest the metadata and also the current known circumstances of the moment, for example, the business context. After the metadata and business context are processed, the guidance is provided to the CBA and business service, which personalize or customize the current instantiation.

Recognizing that much of the guidance that is provided is based upon metadata and business context, if the metadata were to change the guidance that was used for the CBA or business service also changes. This powerful concept is key to achieving and realizing flexibility and agility.

Let us examine this concept further but with a focus on how it is realized by Fabric, as shown in Figure 3-6 on page 28.

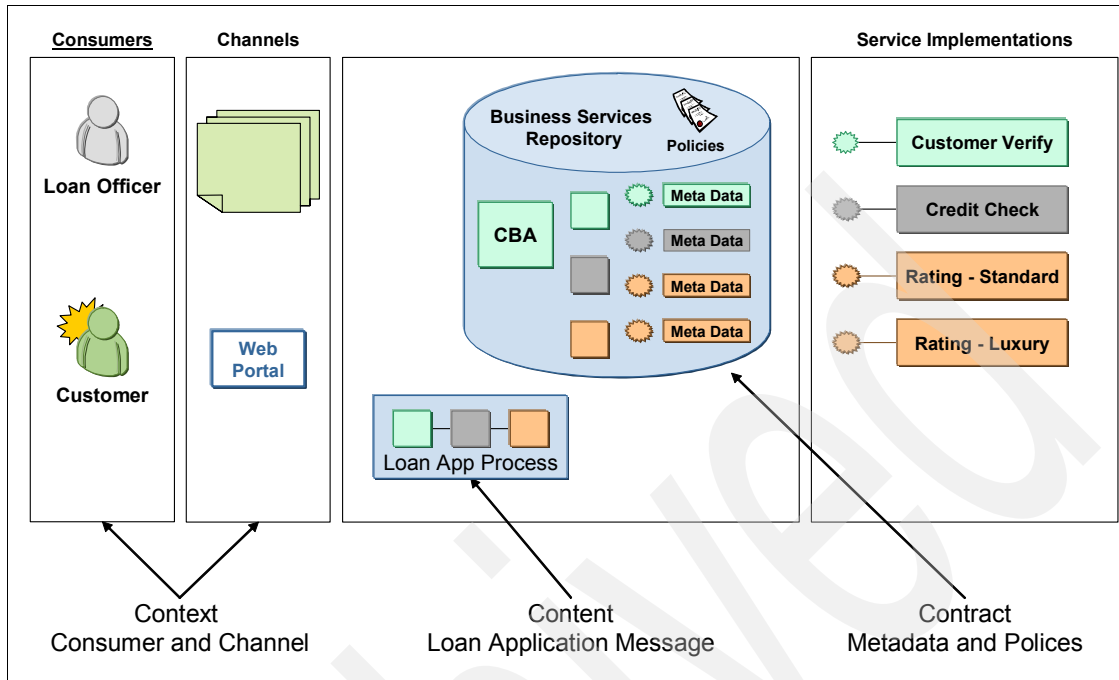


Figure 3-6 Dynamic assembly of business services - 2

Using Fabric terminology, the guidance that is provided to the business service at runtime is the intersection of three things:

- **Context**  
Describes the operating context of the CBA or business service, such as which role is instantiating the request, over which channel, at what time of day, or day of year, and so on.
- **Content**  
Describes information that is available in the data objects that are particular to this instantiation, for instance, if the Loan Application CBA were invoked, it would be important to know if the request was for a home loan or an automobile loan. The content of the invocation messages typically contain this data.
- **Contract**  
Describes the capabilities, restrictions, and preferences for a CBA or business service. The contract is defined at runtime and is a combination of the metadata and policies that are relevant for the particular circumstances.

The series of events that occur at runtime are pictured in Figure 3-7.

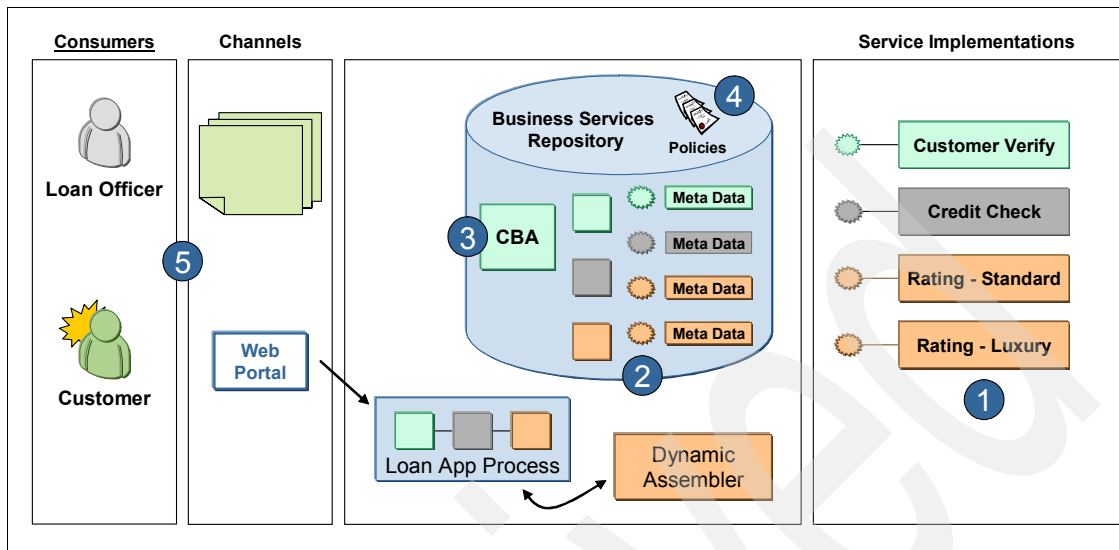


Figure 3-7 Dynamic assembly of business services - 3

Next, we examine the steps that are illustrated in Figure 3-7:

1. Item one represents the endpoint implementations. One endpoint is for a Luxury Home Loan rating and the other is for a Standard Home Loan rating. These endpoints are existing capabilities that are service enabled. The technical specification for the service implementations are made available from a service registry.
2. A Loan Rating business service is composed and metadata that describes the business service is published to a metadata repository. The business service is associated with both the Luxury Home Loan rating and the Standard Home Loan rating service implementations.
3. The Loan Process composite business application is now composed and metadata that describes the service is published into a metadata repository. The Loan Process CBA composes the Loan Rating business service and other services that are not mentioned, such as the Customer Verification service and the Customer Credit Check service.
4. A Policy was created and published to the metadata repository. The policy states that “all home loan applications that are greater than or equal to 500k are considered Luxury Homes, and all home loan applications under 500k are considered standard homes”.

5. A Bank Loan officer or a direct consumer can request a home loan. Two channels exist: a browser interface for the consumer direct loans and a branch office batch interface for Bank Loan officers.
6. For steps 6 through 10, refer to Figure 3-8. For our example, a direct consumer requested the loan application, and the home loan application is for 600k.

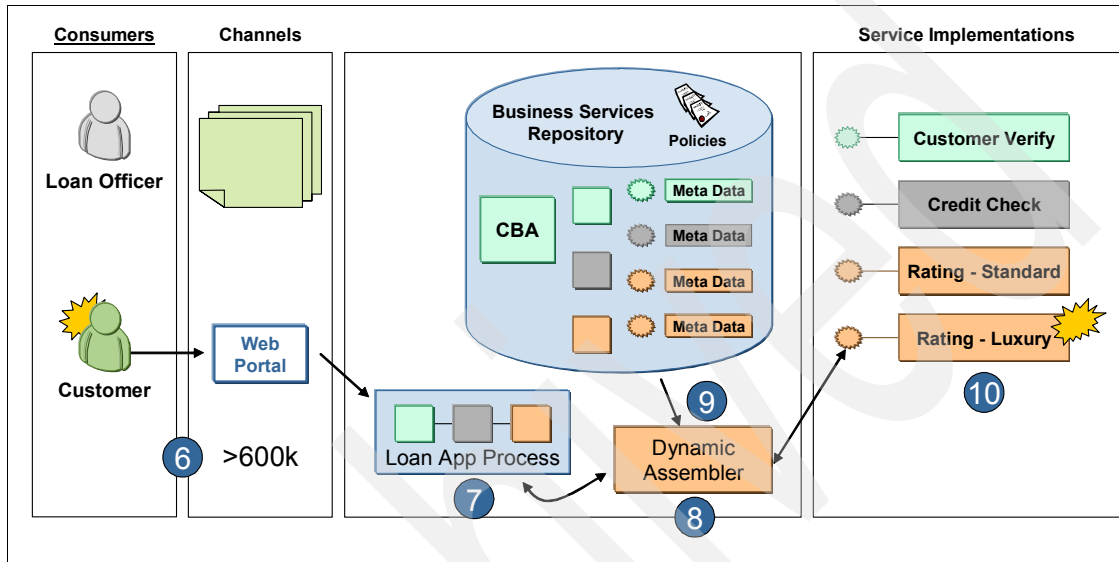


Figure 3-8 Dynamic assembly of business services - 4

7. The Loan Process CBA is instantiated with the loan application. We will consider the Customer Verification and Credit Steps completed. We are now at the Loan Rating Service.
8. The Loan Rating Service has two possible implementations: Luxury Home Rating or Standard Home Rating. The Loan Rating Service defers this decision to the runtime engine, which is the dynamic assembler.
9. The dynamic assembler reviews the context of this request (Direct Consumer over Browser Channel), the content of this call (Loan Application greater than 500k), and reviews all known metadata and policies to assemble the contract or selection policy. One policy is applicable because all loan applications greater than 500k are considered Luxury loans.
10. The dynamic assembler realizes that two endpoints exist for the Loan Rating Service; however, only one implementation meets the policy criteria of rating a luxury home. Given this, the dynamic assembler selects the Luxury rating service to complete the Loan Rating business service.

This simple example articulates the basic actions that Fabric takes to deliver on the concept of dynamic business services assembly. Looking one step further, it is clear how flexibility and agility are achieved using these concepts.

Let us assume that our Loan Process scenario received some additional business requirements. Now all loans over 900k are now being underwritten by a third party, XtraBigLoans, and XtraBigLoans requires that their rating engine be used.

Because we designed our Loan Processing solution using business services and composite business application concepts that Fabric implemented, only a few simple additions to the CBA are required to meet this new requirement:

- ▶ Addition of a new service implementation endpoint that supports XtraBigLoans.
- ▶ Associate the XtraBigLoans service with the Loan Rating business service.
- ▶ Create a new policy stating that all loans greater than or equal to 900k are XtraBigLoans.

Now all loan applications that are greater than 900k are rated using the XtraBigLoan rating service implementation, as shown in to Figure 3-9.

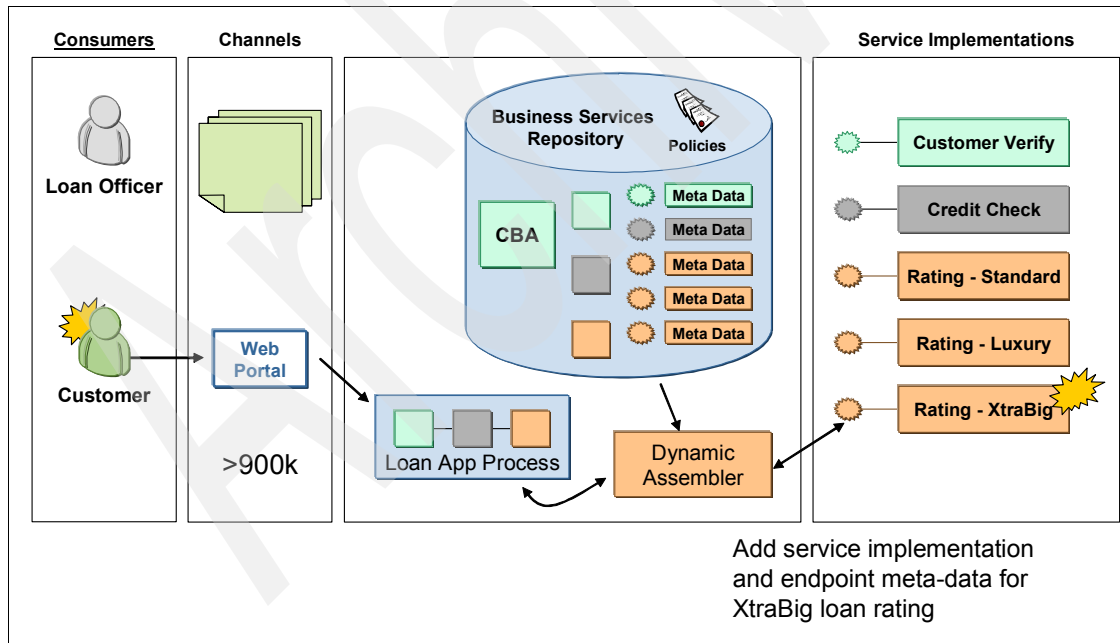


Figure 3-9 Dynamic assembly of business services - 5

With just a few simple additions our Loan Process was able to adapt and react to new business requirements, effectively managing the process at the pace of business.

### **Composite business application key attributes**

Completing our discussion on CBAs and business services, consider the following characteristics that define them:

- ▶ Describable:  
Ability to have its parts described through metadata that can be annotated and published for easy discovery.
- ▶ Composable:  
Ability to be assembled together using services that are exposed from ISV, historical, third-party, custom, Fabric assets, and newly automated white space areas, to represent a business process.
- ▶ Subscribable:  
Ability to be provisioned to consumers and subscribers as a set of portfolio entitlements.
- ▶ Dynamic:  
Composed and executed dynamically at runtime based on context, contract, and content.
- ▶ Interoperable:  
Ability to interoperate with disparate systems and technologies using a common set of industry semantics.
- ▶ Configurable:  
Ability to be customized and configured to support customer use cases.
- ▶ Governable:  
Ability to be managed and versioned through its life cycle.

## **3.2.4 Business services repository**

Business services are the building blocks of composite business applications, and a Business Service Repository (BSR) is the foundation where business services are represented. A BSR provides a place to model, store, and manage the metadata, policies, and entitlements that are necessary to describe and use business services.

Core to Fabric is a comprehensive BSR. The BSR is the central point-of-manageability to the business services and CBAs. The BSR is more

than just a collective bucket of information because it provides the means by which the information is organized and arranged for retrieval and to deliberately present understanding through organization of metadata and relationships between business services, policies, and subscriptions.

The BSR provides the basis for governance and life cycle management of information with regard to creation, ownership, modification, and current state-of-classification. Figure 3-10 shows the BSR metadata model.

The BSR also incorporates ontologies to model the business services knowledge domain. An ontology describes the vocabulary, structure, constraints, relationships, and behaviors that provide a rich and comprehensive representation and understanding.

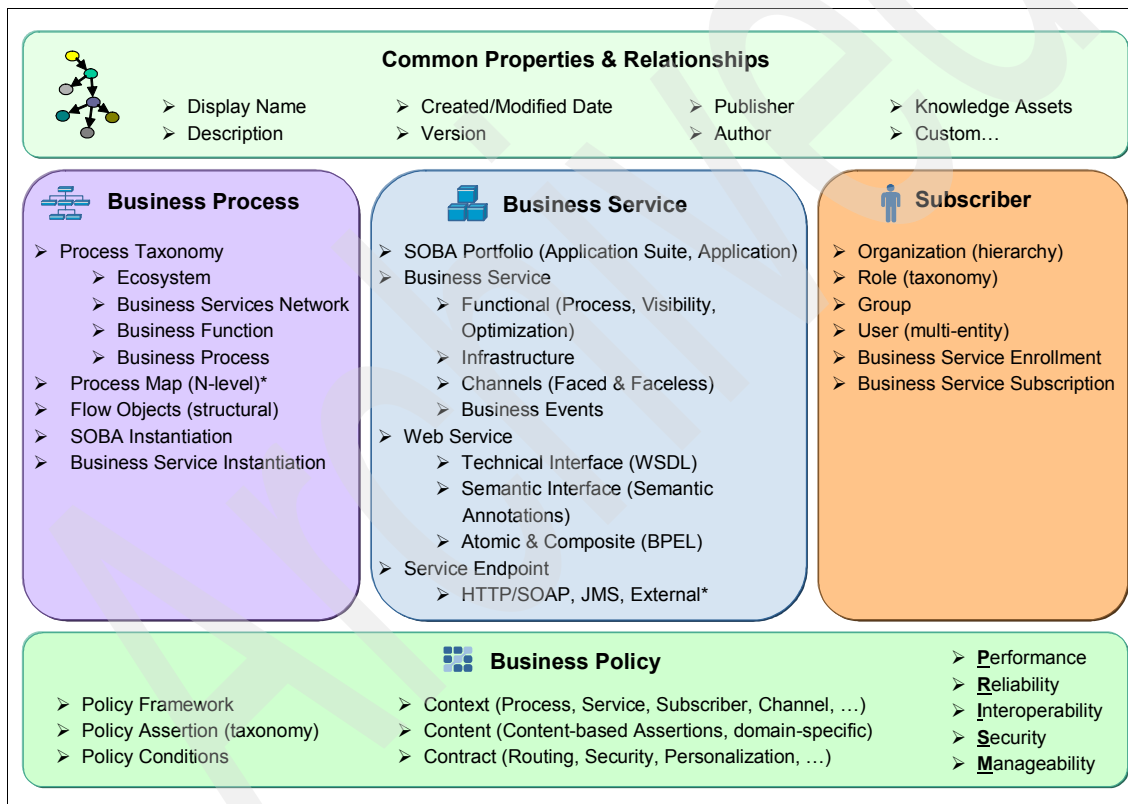


Figure 3-10 BSR metadata model

Ontologies allow the applications that refer to them to be standardized, while the information they contain can be changed over time. Additionally ontologies are modeled with Resources Definition Framework (RDF) and Web Ontology

Language (OWL), which makes them easily extensible and allows them to adapt to future requirements.

Incorporating an ontology provides a unique capability much like referential integrity that is found with relational database management systems. The ontologies are constructed using descriptive logic techniques, which require that certain pre-established relationships exist and are annotated. This unique and critical characteristic provides for integrity to the models during design and development phases.

You can extend the base BSR models using Fabric industry content packs. The industry content packs provide industry specific extensions to the models, which accelerate solution development.

The BSR is persisted to a relational database management system, which provides for enterprise level manageability.

### 3.2.5 Business policies

A large portion of our daily efforts are in some way directed by a policy of some kind. There are policies to tell us how to drive an automobile, how to dress for business, how to dine at formal affairs, and so on. In fact, it can be difficult to come up with an aspect of life or business that is not somehow affected by a policy, whether it be enforced or suggested.

For our conversation, the perspective is easily understood as policies that you can use to clearly articulate business and technical intent with regard to business services, composite business applications, and business processes.

Fabric business policies describe situations and how to address them by expressing business and technical intent and understanding. A Fabric policy is unique because it is a model of a situation and the prescribed outcome or understanding.

We often think of and discuss Fabric business policies using the term *rule* or *business rule*, which generally conjures up the basic premise of an IF..THEN construct. Where we can state something such as, IF x is true THEN A equals B. This terminology is often used to discuss Fabric policies; however, Fabric does not have an IF THEN construct.

In Fabric terms, a business policy represents a key piece of domain knowledge, that must be represented well, for example, easily understood, while also having the ability to be applied directly to a given situation.



## Context, contract, and content

Given the premise that Fabric offers a unique capability to represent policy as models or patterns that are easily understood and applied, Fabric policies are created and maintained within the Business Service Repository. Rather than use an IF...THEN statement, they are declared under the headings context, content, and contract, which defines the three basic constructs of a Fabric policy:

- ▶ The perspective of the service request (context)
- ▶ The request itself (content)
- ▶ The requirements that must be met (contract)

## Business policy example

Let us examine a few example policies so that we can clearly see how they would be represented and applied within Fabric. Earlier, we used the example of a Loan Rating Service, and we articulated a simple business policy that all home loan applications that are greater than or equal to 500k are considered Luxury loan products, and loans less than 500k are standard loan products. Our scenario has two loan rating services, one for standard loans and one for luxury loans.

This business policy looks like:

- ▶ PolicyName - LuxuryLoan
  - Context - Loan application
  - Content - Loan application amount greater than or equal to 500k, and loan type is Home
  - Contract - Loan product is Luxury
- ▶ PolicyName - StandardLoan
  - Context - Loan application
  - Content - Loan application amount is less than 500k, and loan type is Home
  - Contract - Loan product is Standard

This model easily articulates the business policies that are necessary to describe a Luxury and a Standard loan, effectively and declaratively expressing the business intent.

Another example we used was the XtraBigLoan, where the business made a decision to outsource the underwriting of home loans over 900k to a third party. These loans are considered XtraBigLoan Products.

This business policy looks like:

- ▶ PolicyName - XtraBigLoan
  - Context - Loan Application
  - Content - Loan application amount greater than 900k, and loan type is home
  - Contract - Loan product is XtraBig

The business could decide that certain customer loans will remain in house regardless of the loan amount. The policy might state that all premium customers use standard or luxury loan products.

Two business policies would easily represent this business policy:

- ▶ PolicyName - Preferred Customer Standard Loan
  - Context - Loan Application and Customer are Preferred
  - Content - Loan application amount less than 500k, and loan type is Home
  - Contract - Loan product is Standard
- ▶ PolicyName - Preferred Customer Luxury Loan
  - Context - Loan Application and Customer are Preferred
  - Content - Loan application amount greater than or equal to 500k, and loan type is Home
  - Contract - Loan product is Luxury

Additional metadata can also become associated with a business policy, such as an effective date or expiration date, for instance, say the use of third-party provider XtraBigLoan is not effective until January 1st, 2010. The policy effective date of 01/01/2010 is updated, and now this policy is not considered until the specified date.

This XtraBigLoan policy now looks like:

- ▶ PolicyName - XtraBigLoan
  - Context - Loan Application and calendar date greater than or equal to 01/01/2001
  - Content - Loan application amount greater than 900k, and loan type is Home
  - Contract - Loan product is XtraBigloan

## Business service endpoint assertions

Before we can explain how the business policy is applied, we need one more piece to complete our scenarios, which is the actual services implementations that support our Loan Rating business services. For this example, we have three rating service implementations and their respective capabilities:

- ▶ Standard Loan rating endpoint, supports Standard loan products.
- ▶ Luxury Loan rating endpoint, supports Luxury loan products.
- ▶ XtraBig Loan rating endpoint, supports XtraBig loan products.

These endpoints are represented in the Business Service Repository business services model and metadata assertions express their capabilities. A metadata assertion, in our case, is a statement of fact or a declaration of capability. In this case, each of our loan rating endpoints will have a metadata assertion that declares the loan product they support, for instance, the endpoint that supports luxury loan products will have a metadata assertion labeled “loan product” with a value of “Luxury”.

## Applying business policies

At execution time, the Fabric’s runtime engine, Dynamic Assembler, will:

- ▶ Gather all policies that are applicable to the context (loan application) and create a composite policy, which is also called the selection policy. This policy articulates the situation and facts that must be met for the given circumstances.
- ▶ Using the selection policy, Dynamic Assembler finds all candidate endpoints that match that specific selection policy.
- ▶ The endpoint implementation that best supports the selection policy is chosen.

Let us apply the business policies to our example. Say a request has come in from a preferred customer for a home loan. The loan amount that is requested is 450k. The facts are:

- ▶ Customer type is Preferred.
- ▶ Loan Type is Home.
- ▶ Loan Amount is 450k.

Dynamic Assembler recognizes that only one policy applies to this request, which in this case is the Standard Loan policy, where the context is a loan application, and the loan amount (content) is less than 500k.

The Dynamic Assembler now considers this policy as the selection policy. Remember that this policy has a contract assertion stating that the loan product is Standard. Using this, the Dynamic Assembler selects candidate endpoints that

support the Loan Rating business service and also loan products. Three endpoints are in the list:

- ▶ Standard Loan rating endpoint, when the loan product is Standard
- ▶ Luxury Loan rating endpoint, when the loan product is Luxury
- ▶ XtraBig Loan rating endpoint, when the loan product is XtraBig

The Standard Loan endpoint is the only endpoint that matches the selection Policy and states that the loan product must be Standard. So the Standard loan endpoint is chosen.

### **Business policies: a summary**

We noted that:

- ▶ The Fabric business policies are represented and housed in the Business Services Repository.
- ▶ The BSR represents and stores business services' metadata and subscriber information.
- ▶ The metadata is realized using a business services model, or ontology. The ontology provides the means for the business policies, business services and subscriptions to share the same basic terms and understanding and provides for correlations to one another.

As an example, we used the terms, “loan product”, “loan type”, and “loan application amount” consistently throughout our examples. These terms are also represented and provided consistently within the BSR business services model and made available for use by additional policies and business services. Changes to the terms affect all resources equally, which provides us with a very consistent and manageable representation of the business and technical intent.

## **3.2.6 Service provisioning and enrollments**

A critical aspect to delivering business services and composite business applications is consideration for entitlement to the business services and solutions. After it is created, the introduction of solutions to the enterprise and ecosystem partners must be delivered and managed in a flexible manner.

Having composed a solution is not enough to realize the intended value. You must make the service visible and available for consumers to act upon it. Typically, you do not make the solutions and business services available to anonymous users and organizations; instead, they are purposely provisioned to internal and external entities for specific business purposes.

It is common for services to be introduced incrementally, over time to groups, organizations, and geographies as opposed to being made generally available to

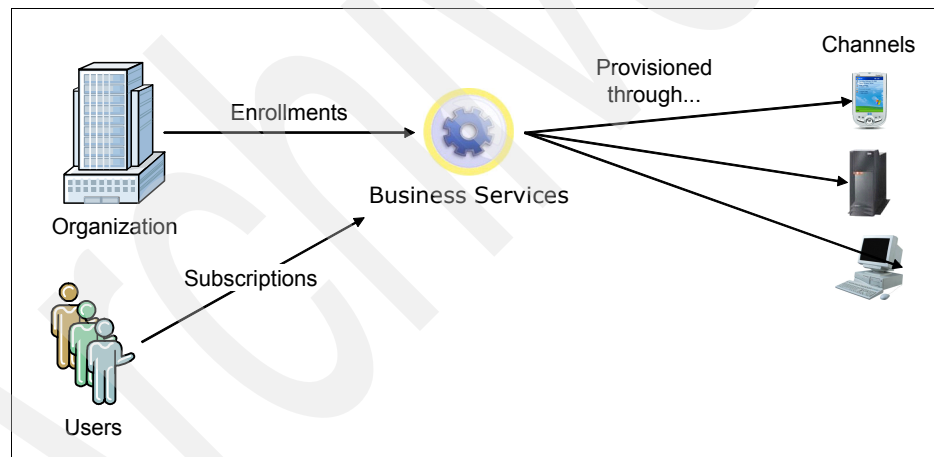
anyone or any organization. At times, you might need to remove or restrict services entitlements to various groups.

Fabric provides capabilities to effectively provision and set forth entitlements to business services and composite business applications.

Fabric considers three aspects important to the provisioning and entitlement of services, Subscribers, Enrollments, and Organizations:

- ▶ Subscribers are people, roles, organizations, or machines who are consumers of business services.
- ▶ Enrollments is an organizational association to one or more business services.
- ▶ Organization enrolls into a business service, so the organization's groups and users can subscribe to those business services.

Organizations enroll with business services, and users subscribe to business services, and business services are provisioned through channels, as we show in Figure 3-11.



*Figure 3-11 Provisioning and entitlement of services*

A subscription model is maintained with the BSR that collects and stores subscriber information. Additionally, because this information is maintained within the BSR, it realizes the advantages of visibility to the business services, composite business application, and business policies.

Fabric enables flexible business service provisioning through either self-service or assisted provisioning and supports auto provisioning and de-provisioning for a business service portfolio.

Fabric provides support for authentication and authorization systems through industry-standard interfaces and integrates seamlessly with identity management systems and LDAP through WebSphere Virtual Member Manager (VMM).

As we discussed, the notion of a business service includes not only the functional capabilities it delivers but also who might consume it and how it is made available. Fabric's unique provisioning and entitlement functionality are included in the technologies that are necessary to deliver composite business applications in an BPM-enabled SOA.

### **3.3 Fabric key benefits**

Delivering timely and effective solutions to an organization or enterprise are the driving reasons to embrace technology. Regardless of the technology chosen, a bottom-line benefit must be realized. That benefit can be the accumulation of many complex activities that work in unison to achieve a given task or a simple collection of steps that are available when needed. Regardless of the complexity, sophistication, or simplicity, the bottom line benefit and value is always easily recognized. Easily stated, it gets the job done or makes a task possible.

Fabric contributes to the success and benefit of solutions in a number of ways, some are extremely obvious while others might seem slightly more hidden. Of course the visibility of the benefits are predicated upon your perspective and vantage point, but nonetheless exist and have been proven many times over.

In this section, we discuss a few of the key benefits and values that are realized through Fabric so that as you consider a design or solution you can reflect upon them.

#### **3.3.1 Business process simplification**

Fabric, when combined with decision-intensive business processing, provides a unique means to extract decision-logic from the process and aggregate it with the business services that it helps to define. This capability reduces the sophistication and complexity that is associated with a decision-intensive business process.

The extracted decision logic now becomes declarative knowledge that is aggregated into a meaningful and consistently represented manner. With the decision logic extracted, represented, and managed in one place, Fabric creates a unique opportunity to improve flexibility and agility of that process, particularly with respect to changes and modifications.

This unique capability of Fabric allows us to use the metadata and the policies that are contained with the BSR as a means to drive the business processes.

Let us examine our Loan Application process example from the perspective of the business process.

The process starts with a simple series of steps that consist of customer verification, credit check, and loan rating and underwriting. Figure 3-12 illustrates this process.

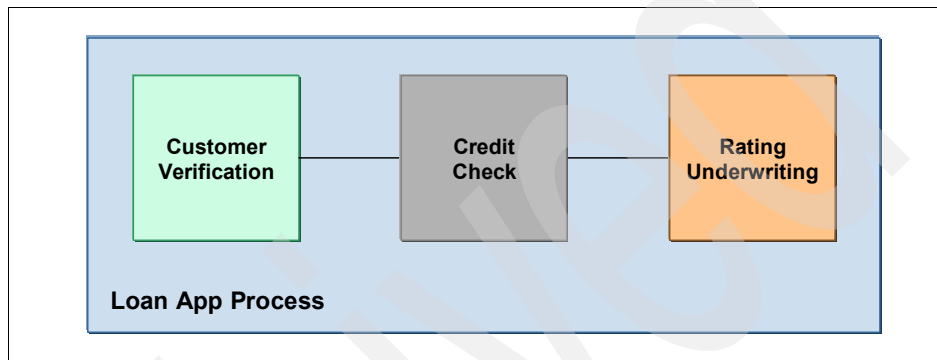


Figure 3-12 Loan application process - 1

Focusing just on the loan rating steps, we recognize that a decision must be made on the loan size, where loans less than 500k are considered standard loans and loans greater than or equal to 500k are considered luxury loans. Figure 3-13 illustrates this process.

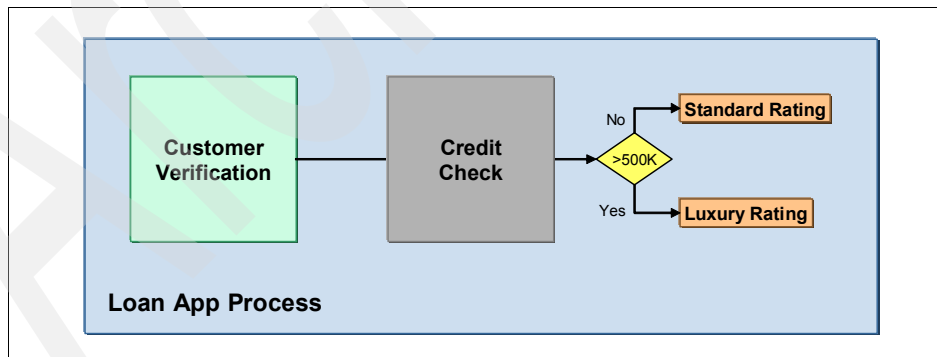


Figure 3-13 Loan application process - 2

Our example did not stop there because the business wanted to outsource all loans that are greater than or equal to 900k to a third party, XtraBigLoans, unless

the customer was a preferred customer, in which case they would like to keep the loan.

Considering these additional business requirements, the process now looks like, Figure 3-14.

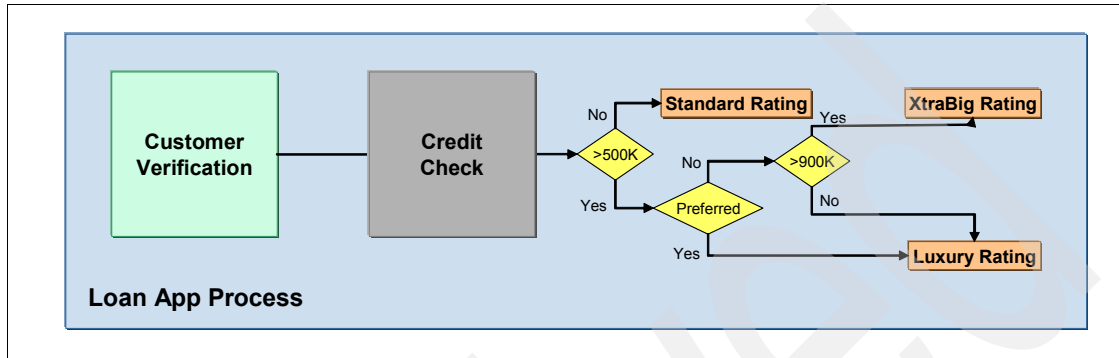


Figure 3-14 Loan application process - 3

Further, we must consider that each of the modifications that we added to the process occurred over time and each required that the process change, which lead to additional development and testing.

In contrast, if we use Fabric to implement this solution, the process remains as originally stated, with just the three business activities of customer verification, credit check, and loan rating and underwriting. But now the process steps are created and composed as business services and business policies.

The changes per the business requirements included adding luxury rating and adding XtraBigloans. Special considerations for preferred customers can now be included without changing the business process, which significantly reduces the development and testing time that is required to get these changes to market.

We extracted business process decision logic from the process and consolidated it into one place, which allows the process to behave dynamically based upon business context. Process information is now located in once place, the Business Service Repository and available for use by other processes, as illustrated in Figure 3-15 on page 43.



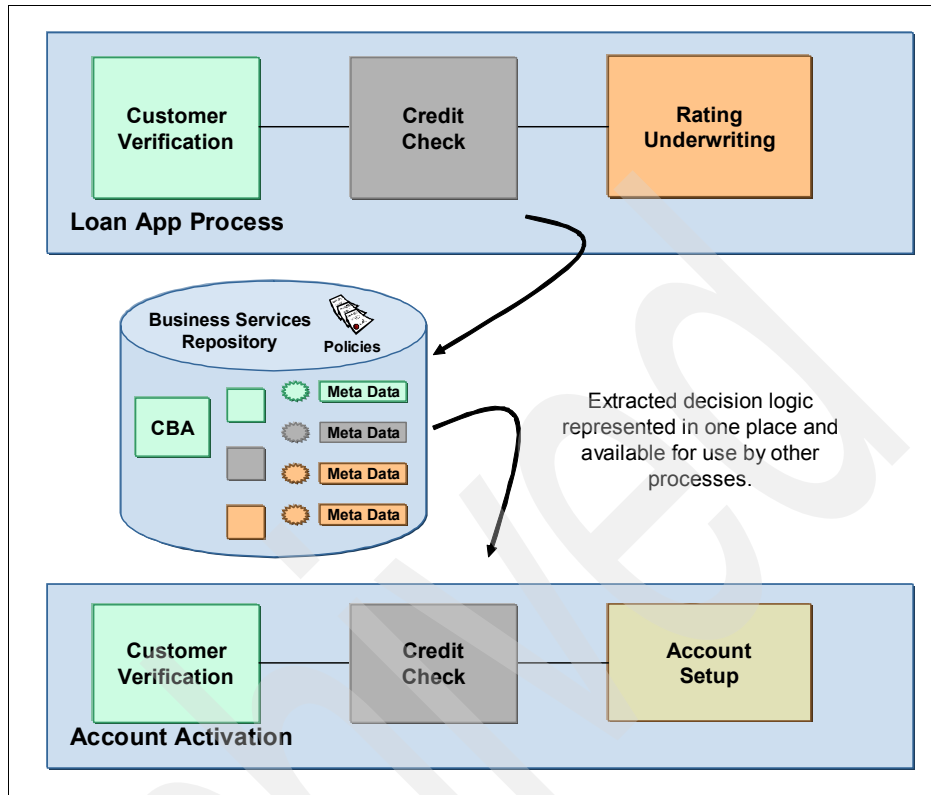


Figure 3-15 Loan application process - 4

### 3.3.2 Business process decision logic consolidation

Fabric creates the opportunity for us to consolidate the business process decision logic in one place, the BSR, in a consistent and understandable representation as declarative knowledge. This declarative knowledge is now in the form of business level policies and business service metadata instead of buried as hard coded endpoint bindings or process choreography logic.

The aggregated business knowledge is now available for reuse and can be managed constantly and deliberately, which allows the process to stay in step with the business intent.

### 3.3.3 Provides a business services metadata model

Core to Fabric is the BSR, which creates and establishes a consistent model of business services and business level policies. It is a central repository for

business service metadata, domain ontologies, policies, and subscribers. It creates a place to capture business constraints around service delivery and maps core business functions to supporting IT capabilities.

The Fabric BSR creates benefit and value in providing a place to house and store the information and by providing a consistent model and understanding that goes beyond a services list of capabilities.

A core benefit to an SOA is reuse, and to embrace reuse requires a comprehensive and clear understanding of the service itself and the information suggesting where it is used, by, and over what channels.

A bank would never loan money without tracking to who, under what terms, and how much. Likewise, this same understanding and tracking must also be provided for business services.

The Fabric Business Service Repository, through an ontology, provides a business services model of metadata that describes a business service and also understands the relationship a business service has to other business services, service implementations, subscribers, and consumption channels, which provide correlations of information that are contained within the model. Now the metadata is more than assertions and capabilities, it also provides us with an additional level of understanding.

Now as designers and solution developers wish to reuse a service, information about the business services current usage can quickly and easily be obtained and made available. This avoids possible go-live production issues, such as realizing after the fact that a service component is beyond 100 percent utilization or not entitled to your target deliver audience.

The ability to respond to simple questions, such as which users are consuming that service and over what channel, might seem obvious but without a proper business services understanding it can be impossible. The ability to answer this question and others like it is a tremendous benefit that embracing the Fabric provides.

### **3.3.4 Enables business service portfolio management**

Fabric's unique entitlement and provisioning capabilities create opportunities to better manage and deliver developed solutions to consumers. You can gather and represent business services and composite business applications using the BSR as portfolios of business capability.

Introducing these solutions to consumers can be challenging because a solution is usually not made available to the entire population of consumers at once, for

instance, an insurance application exposes a new capability to its agents; however, this capability is only supported for one region of the territory. This service must be exposed to the region it is built for first, while others in the territory are added later.

Fabric provides the capability to support this type of phased or incremental delivery of services. You can make the business services available through entitlements, and if special circumstances exist, you can also add business level policies. Because Fabric represents complex organizational hierarchies and captures metadata that is related to subscribers and roles, very granular provisioning and entitlement of business services and composite business application portfolios is possible.

You can manage a portfolio using an administrator or through the consumer themselves. Fabric supports administrated, delegated, or self-service provisioning of services that are made available.

### **3.3.5 Provides business services visibility**

Understanding how a business is performing, who is using it, and over what channel is critical information that all enterprises that deliver business solutions need. All solutions that are delivered with Fabric can take advantage of the business service and reporting capabilities that Fabric makes available.

When a business process includes the Fabric Dynamic Assembler to select an implementation at run time, information about that service invocation is maintained and made available as reporting statistics. The information provided by Fabric is unique and interesting because it includes both subscriber or user data and consumption channel information. This data is easily and readily available through administrative console capabilities, or you can extract it using business intelligence tooling.

Using this information, administrators and business owners can have access to relevant information, such as who is consuming the service, over what channels, and what service levels are they witnessing. This is critical information to know when managing a portfolio of business services.

Additionally Fabric makes information available to other instrumentation, such as Tivoli Composite Application Manager for SOA.

### 3.3.6 Provides business services metadata governance

Establishing a life cycle and governance of the metadata that is contained within the BSR is a core Fabric capability. Fabric provides capabilities to control all aspects of metadata that are contained within the BSR.

In the traditional sense, Fabric provides a means by which changes to metadata are submitted, tracked, approved, and published, which is a requirement that is absolutely necessary in any collaborative environment.

Additionally, Fabric provides a unique capability that validates meta-models to ensure accuracy and correctness before they are published. This capability is partly driven by the use of ontologies and descriptive logic and then reinforced by the metadata life cycle management process.

The value of the maintained information, which is provided and maintained within the Fabric BSR, is critical to the success of a BPM-enabled SOA. Carefully guarding and managing that data is a service that is provided with Fabric through comprehensive governance, which includes tracking changes among collaborative teams, ensuring accuracy and completeness of the ontological models, and also by providing a method and means by which this is delivered.

# Technical overview of WebSphere Business Services Fabric

Delivering on the promise of service-oriented architecture (SOA) through Composite Business Applications (CBAs) could be a difficult journey without the necessary tools and guidance to lead the way and to assist with the details. The ability to quickly construct new applications through composing independent business services creates new and exciting ways of delivering core business solutions. Leveraging a platform that is specifically designed to deliver composite business applications is critical for success.

WebSphere Business Services Fabric (Fabric) is an end-to-end SOA platform to model, assemble, deploy, manage, and govern composite business applications. Fabric provides the design-time tooling and runtime environment for the composition and deployment of flexible and agile composite business applications. Fabric includes WebSphere Process Server (runtime environment) and WebSphere Integration Developer (design time tooling).

In this chapter, we begin with an overview of the Fabric base products, followed by details on each of the respective modules, and then a discussion about Fabric Dynamic Assembler and Fabric policies.

## 4.1 Product overview

Fabric is a rich product with many benefits and features. The following list highlights some Fabric key features:

- ▶ Highly scalable runtime engine, *Dynamic Assembler* that enables business services to be delivered based on content, context, and contract.
- ▶ End-to-end platform that is used to source, assemble, deliver, and manage business services for BPM-enabled SOA applications.
- ▶ Extends and enhances IBM-core BPM-enabled SOA products that include WebSphere Process Server, WebSphere Integration Developer, and WebSphere Service Registry and Repository.
- ▶ Contains a standard-based business metadata repository that stores and manages business services, policies, and business service entitlements.
- ▶ Eclipse-based and Web-based tools that enable collaborative assembly and management of business services for SOA applications.
- ▶ Visual assembly environment for creating and managing industry-specific business service metadata models and policies.
- ▶ Inter-operates with external monitoring systems.
- ▶ Governs across the product life cycle, which includes the creation of versions and migration across environments.

The WebSphere Business Services Fabric also includes optional *industry content packs*. Industry content packs contain prebuilt industry assets that contain industry-common services and industry SOA models. Industry content packs accelerate business service development.

## 4.2 WebSphere Business Services Fabric product positioning

Fabric includes the tools and the runtime model that can assemble, deploy, manage, and govern business services, which are the reusable building blocks of composite business applications, as illustrated in Figure 4-1 on page 49.

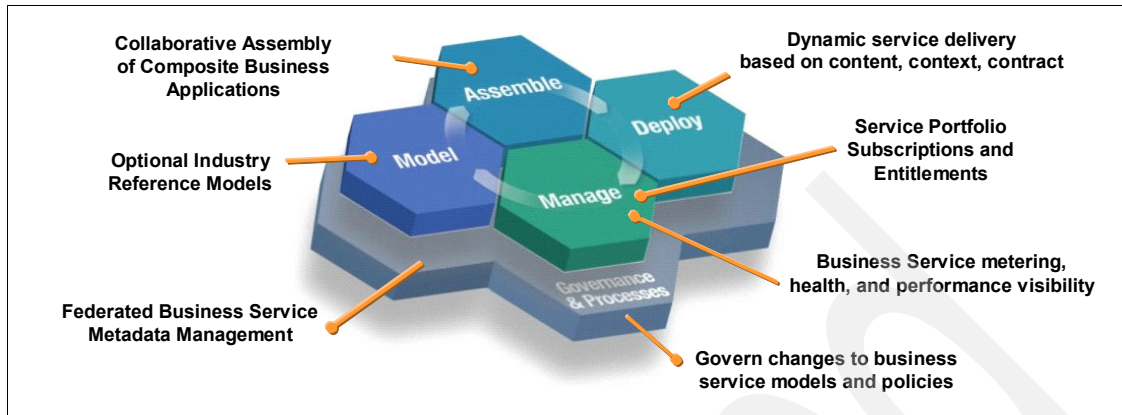


Figure 4-1 SOA Foundation life cycle

## 4.2.1 The reusable building blocks of composite business applications

In this section, we discuss the reusable building blocks of composite business applications: model, assemble, deploy, manage, and govern.

### Model

The model building block of composite business applications:

- ▶ Enables code-free discovery of services from process models, service registries, code repositories, and file systems.
- ▶ Models top-down and bottom-up policy-driven relationships between subscribers and business services that are sourced from internal and external applications.

### Assemble

The assemble building block of composite business applications:

- ▶ Facilitates the simplified, metadata-driven assembly of business services into flexible processes and applications.
- ▶ Assembles solutions from an inventory of business services.
- ▶ Leverages prebuilt, customizable industry reference models.

## **Deploy**

The deploy building block of composite business applications:

- ▶ Enables role-based, contextual assembly and delivery of business services across multiple communication channels, such as the World Wide Web, business-to-business (B2B), and interactive voice response (IVR).
- ▶ Delivers dynamic, context-aware business processes by replacing hard-coded service bindings with dynamic endpoint selection that is based on content, context, and contract.

## **Manage**

The manage building block of composite business applications:

- ▶ Automates and controls multi-entity and multi-domain entitlement of business services for service consumers.
- ▶ Enables application administrators to create, control, and manage role-based on-demand service packages to subscribers across the business ecosystem.

## **Govern**

The govern building block of composite business applications:

- ▶ Provides architectural visibility and control by applying consistent design-time, run time, and change time policies.
- ▶ Governs multi-author, multi-geography collaborative service design, assembly, and delivery.
- ▶ Leverages out-of-the-box integration with the Business Services Repository, and open APIs for seamless integration with change configuration systems and Web services management systems.

### **4.2.2 WebSphere Business Services Fabric product components**

Fabric consists of three basic components, as shown in Figure 4-2 on page 51.



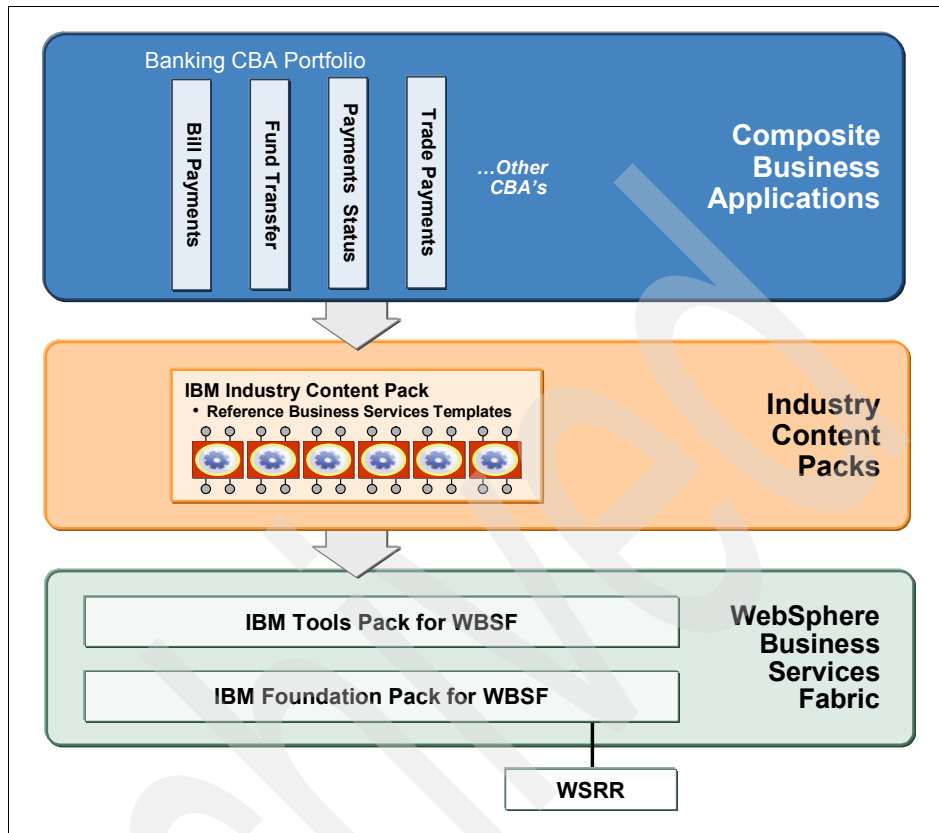


Figure 4-2 Fabric product components

Fabric is categorized into three basic buckets:

- ▶ **WebSphere Business Services Fabric:**  
End-to-end platform for composite business services, which includes WebSphere Processing Server and WebSphere Integration Developer.
- ▶ **Industry Content Packs:**  
Pre-built, industry-specific SOA assets that speed time-to-market and instills industry standards and best practices. Industry content packs are optional extensions to the Fabric.
- ▶ **Composite Business Applications:**  
Flexible, adaptable, loosely coupled business applications and processes. They may be created as customer solutions or solution frameworks available for purchase from IBM Global Business Services (GBS), System Integrators (SIs) or Independent Software Vendors (ISVs).

The Fabric modules include two software packs and four optional industry content packs.

The two software modules are:

- ▶ *IBM Business Services Foundation Pack*
  - Provides a highly scalable, runtime policy definition and enforcement engine that enables dynamic service assembly and service behavior adaptation based on content, context, and contract.
  - Controls and automates entitlement of business services for subscribers, which enables creation, control, and management of service packages to subscribers across the ecosystem. Integrates with leading security and identity management products.
  - Provides visibility and monitoring of service-oriented processes and applications. Includes multi-perspective views and enables drill-down analysis of events and exceptions.
  - Provides end-to-end governance of business services through design-time, run time, deploy time, and manage time.
  - Stores and manages business services, policies, and business service entitlements.
  - Enables business service adaptation and delivery based on content, context, and contract.
- ▶ *IBM Business Services Tool Pack*
  - Provides user tools to maintain, manage, and govern services metadata, subscribers, policies, and governance processes.
  - Enables domain-experienced software architects to model, create, publish, and manage industry-specific, service-metadata models and policies around data, processes, resources, policies, and domains.
  - Enables a visual assembly environment for creating and managing industry-specific, business-service metadata models and policies.

The four optional industry content packs are:

- ▶ IBM Banking Payments Content Pack
- ▶ IBM Healthcare Payor Content Pack
- ▶ IBM Insurance Property and Casualty Pack
- ▶ IBM Telecom Operation Content Pack

The Fabric Foundation Pack includes WebSphere Process Server, and the Fabric Tools Pack includes WebSphere Integration Developer.

## Federation of external resources

Enterprises use a variety of systems to store and search their critical data. The ability to federate from different data sources is key to realizing the full value of the data they contain. Fabric supports federation from external data sources, such as *Lightweight Directory Access Protocol (LDAP)* directories and the *WebSphere Service Registry and Repository (WSRR)* product.

Fabric considers WebSphere Services Registry and Repository (WSRR) a prerequisite. However, WSRR is not included with the Foundation Pack nor the Tools Pack. You must separately purchase WSRR.

## Supported platforms

The Fabric Foundation Pack provides support for AIX®, Linux Red Hat Enterprise Edition (RHEL), Linux SUSE Linux Enterprise Server (LHEL), Windows® Server 2003 Enterprise Edition SP2, Windows® XP Professional SP2, and z/OS®.

The Fabric Tools Pack provides support for Windows® XP Professional SP2 and Microsoft® Vista (Business & Enterprise).

**Note:** More details on supported platforms are at:

[http://www-306.ibm.com/software/integration/wbsf/sysreqs/?S\\_CMP=rnav](http://www-306.ibm.com/software/integration/wbsf/sysreqs/?S_CMP=rnav)

### 4.2.3 Business Services Foundation Pack

The Business Services Foundation Pack builds upon and extends the capabilities that the WebSphere Process Server provides. The Foundation Pack includes the *Business Services Dynamic Assembler*, *Fabric Administrative Console*, and the *Fabric Software Development Kit*. Refer to Figure 4-3 on page 54.

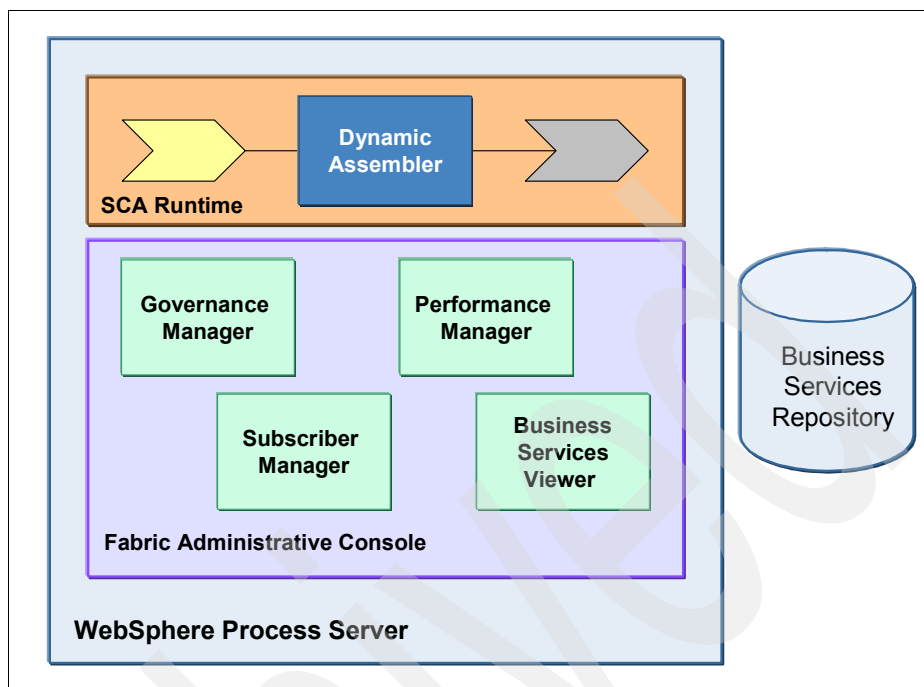


Figure 4-3 WebSphere Business Services Foundation Pack

► Business Services Dynamic Assembler (Dynamic Assembler)

The Foundation Pack extends the WebSphere Process Server runtime by introducing a metadata-driven dynamic service assembly engine. The Dynamic Assembler is packaged as a Service Component Architecture (SCA) component and is managed by the SCA runtime within the WebSphere Process Server.

► Fabric Administrative Console

A Web-based administrative console that provides the management and governance of business services and composite business applications. The Fabric Administrative console contains the following five modules:

– Business Services Governance Manager

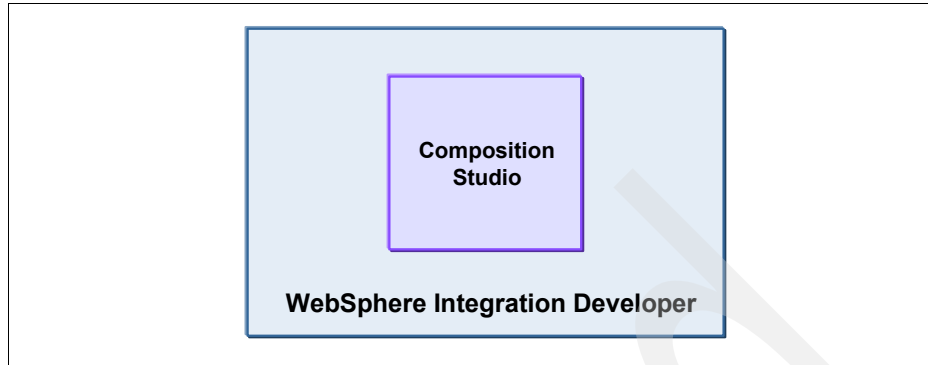
Provides the administrative capability to control access and visibility to business service models and policies within the IBM Business Services Repository component.

- Business Services Repository  
An ontology-based business service model repository that captures information about business services, business policies, and service entitlements.
- Business Services Subscriber Manager  
Enables you to create, control, and manage service portfolios for subscribers across a business ecosystem and interoperates with leading security and identity management products.
- Business Services Performance Manage  
Provides visibility and monitoring of business services-based solutions.
- Business Services Viewer  
A tool that provides search and view capabilities into the contents of the Business Services Repository.
- Fabric Software Development Kit  
A set of Web service APIs that give applications programmatic access to the business services metadata and policies.

**Note:** The Foundation Pack includes WebSphere Process Server. For those existing customers of WebSphere Process Server a separate Foundation Pack product is available that does not include WebSphere Process Server.

#### 4.2.4 Business Services Tools Pack

The IBM WebSphere Business Services Tool Pack provides the assembly environment for business services. The Tool Pack extends WebSphere Integration Developer with the tooling that is required to create and manage business services metadata. Business services related tooling is packaged as an Eclipse plug-in that adds an additional perspective to WebSphere Integration Developer called the Business Service perspective, as shown in Figure 4-4 on page 56.



*Figure 4-4 WebSphere Business Services Tools Pack*

The Tool Pack Includes:

- ▶ **WebSphere Integration Developer**  
Enables you to assemble and integrate business services across WebSphere Process Server. With the WebSphere Integration Developer's flexible user interface, you can generate and customize user interfaces that can be used for human interactions within a business process.
- ▶ **Composition Studio**  
Allows a developer to assemble business services, create business policies, and simulate dynamic assembly of business services. Composition Studio is packaged as an Eclipse plug-in and adds an additional perspective to WebSphere Integration Developer called Business Service.
- ▶ **Unit Test Environment**  
Allows you to build and execute business services on your workstation. The Fabric Unit Test Environment (UTE) contains all of the Foundation Pack components.

### 4.2.5 Industry content packs

A Fabric solution development life cycle can be accelerated for specific industries through the application industry content packs that contain prebuilt SOA assets. These assets are made of business service templates and are based on industry standards and prevalent industry best practices.

Industry content packs can speed time-to-market for industry-specific service-oriented business solutions.

The following list identifies the benefits of using the industry content packs:

- ▶ Designed for specific industry participants to facilitate more rapid service-oriented business solution assembly.
- ▶ Optimized for industry best practices, which ensures consistency and reuse across geographical locations, product lines, and processes.
- ▶ Aligned with top-down decomposition of an industry domain into business capabilities and processes, which provides visibility into reuse across the organizational ecosystem.
- ▶ Help speed time-to-market using business service templates with associated business metadata that ensures consistency and reuse.
- ▶ Help simplify interoperability by defining common language in the form of prepackaged industry-specific vocabularies, such as ACORD, HL7, ISO 20022, and NGOSS SID, to facilitate interoperability between disparate IT assets.
- ▶ Help speed time-to-market with prebuilt, frequently reused, and industry standards-specific service implementations for common transactional functions, such as validation and error identification.
- ▶ Enable standardized connectivity to disparate applications in the IT ecosystem with prebuilt service interfaces that are based on industry standards.
- ▶ Help speed time-to-market using prebuilt business metadata that ensures consistency and maintains the standard compliance during the definition and development of business service policies.
- ▶ Expands the use within an enterprise or enhances the security of publishing to a broader partner ecosystem.
- ▶ Enables clients and partners to plug in their SOA assets into predefined business service templates.

Fabric industry content packs are specialized to industry and also specialized to the solutions they can accelerate within an industry. If an industry content pack exists for your industry, you should review and match the content that is contained within the pack against current business objectives. If they match, an industry content pack can greatly accelerate the development of business services and the delivery of composite business applications.

## 4.2.6 Software development kit

Composite business applications that are built with Fabric have access to a suite of services through the WebSphere Business Services Fabric Software Development Kit (SDK). The Fabric SDK provides a collection of APIs that are

exposed as Web services to access and customize various components and their functionality. All APIs are made available as Web services.

The SDK offers APIs for performing operations that are related to the following modules:

- ▶ The Business Services Repository provides the following functionality for subscribers and resources:
  - Create users, subscriptions, and organizations.
  - Update user preferences.
  - Query for application suites, applications, and services.
  - Query for granted service actions.
- ▶ The Dynamic Assembler provides the following functionality for policies:
  - Read and manipulate (customize) policies.
  - Send 'broadcast' events that are delivered to recipients based on catalog metadata, for example, policies.

## 4.3 Product component overview

In this section, we discuss:

- ▶ 4.3.1, “Project setup, project development, and project deployment” on page 58
- ▶ 4.3.2, “Fabric administrative console” on page 61
- ▶ 4.3.3, “Composition Studio” on page 67
- ▶ 4.3.4, “Business Services Repository” on page 69
- ▶ 4.3.5, “Dynamic Assembler” on page 71

### 4.3.1 Project setup, project development, and project deployment

The Foundation Pack and Tools Pack modules work together to deliver composite business applications. The steps are divided into two basic activities: project setup and project development through deployment.

The tasks and associated Fabric modules for project setup are represented in Figure 4-5 on page 59, and we describe the tasks after the figure.



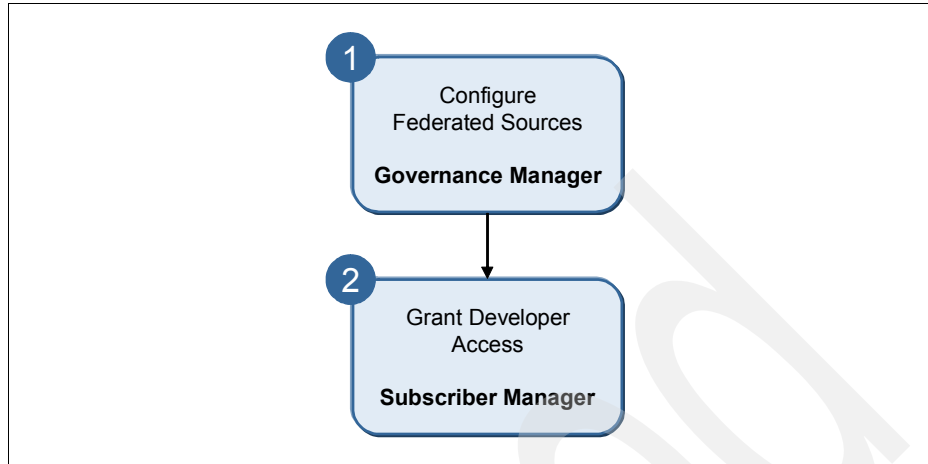


Figure 4-5 Fabric project setup

1. Configure federated sources.

Fabric can federate to other sources, such as WSRR or LDAP. Governance Manager makes the necessary configurations.

2. Grant developer access.

Before a developer or user can view a Fabric project, they must be granted access. Subscriber Manager administrates the access.

Figure 4-6 on page 60 illustrates the tasks and associated Fabric modules for project development to deployment are represented. Following Figure 4-6 on page 60, we explain the tasks.

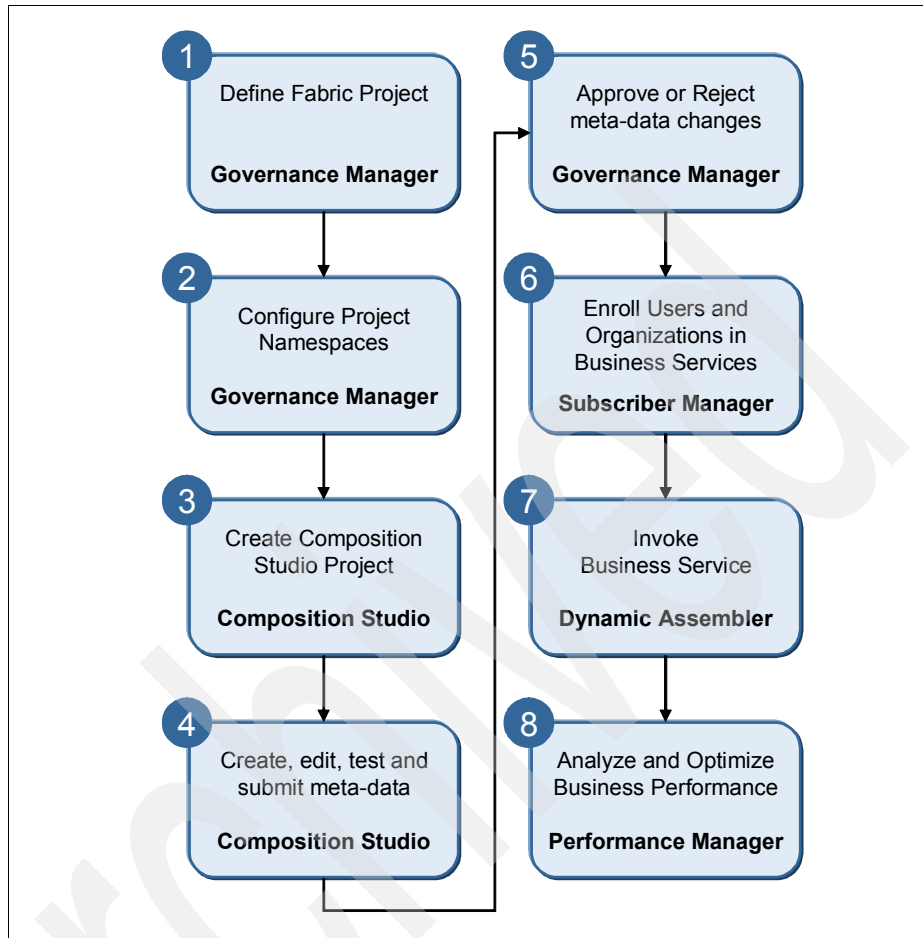


Figure 4-6 Fabric project development to deployment steps

1. Define Fabric project.

A Fabric project is created to manage the metadata for a set of applications and services. Governance Manager defines the project.

2. Configure project namespaces.

A Fabric project consists of one or more namespaces. The namespaces need to be created and configured. Governance Manager configures the namespaces.

3. Create Composition Studio project.

A Composition Studio project is specific to a developer, where as a Fabric project is exposed to a broader user community. Composition Studio projects are created in the Composition Studio tool.

4. Create, edit, test, and submit metadata updates.

Composition Studio completes the composition of business services, composite business services, and other metadata entries.

5. Approve or reject metadata change.

As developers create and modify a Composition Studio project, the changes must be approved prior to being published to the Fabric project. Governance Manager provides the change-list management activities to accept or reject metadata changes.

6. Enroll users and organizations into business services.

Before a business service can be invoked, it must be provisioned. Subscriber Manager provides the administrative tooling to create and manage business service entitlements and subscriptions.

7. Invoke business service.

Business services are invoked at runtime using the Dynamic Assembler component.

8. Analyze and optimize business performance.

Performance Manager provides business visibility and monitoring of service-oriented processes and composite business applications.

### 4.3.2 Fabric administrative console

The Fabric administrative console includes the following components:

- ▶ “Governance Manager” on page 61
- ▶ “Subscriber Manager” on page 64
- ▶ “Performance Manager” on page 66

#### **Governance Manager**

The Business Services Repository stores metadata that is used to create business services in an SOA system. The metadata describes the services, policies, relationships among them, and entitlements to subscribers. Using Composition Studio, you can create or modify the metadata.

A multi-authoring environment controls the changes that are published to the Business Services Repository. To ensure data integrity, monitor and resolve any conflicts to the changes that you apply to the metadata. Using Governance

Manager, you can manage the metadata change process through the following capabilities:

- ▶ Configure projects.

Governance Manager provides the mechanism to create Fabric projects. All projects are associated with a *team*, which is an organization that is managed through Subscriber Manager. Only users on the team can use a given project. Fabric supports the following project types:

- Business service project

All CBA metadata is stored in name spaces, which is owned by this project type.

- Ontology extension project

This project contains extensions to the Fabric business service model for new Assertion Types, Role Types, and Channel Types.

- Ontology project

This project contains the business services model.

- External WSRR project

Web services metadata federated from WSRR registries belong to this type of Fabric Project.

- External LDAP project

All user data that is federated from LDAP belongs to this type of Fabric Project.

- ▶ Configure resource federation.

Fabric supports federation with LDAP and WSRR. The external resource configurations are created using Fabric projects of the respective type, that is external WSRR project or external LDAP project.

- ▶ Configure namespaces.

- ▶ Configure and manage namespaces for projects.

- ▶ Configure environments and repository.

Configure a repository to a specific environment, such as Test, Development, or Production. Manage changes.

- Import and export.

Provides the ability to move data between projects and systems. Users can export some or all of the data for a project to a Fabric Content Archive file, and then import or update this project on to another system.

- Manage changes.

To view, approve, or reject metadata changes, and to publish approved changes to the Business Services Repository.

A key capability of Governance Manager is to manage changes that are submitted for publication to the BSR. A developer begins working on a project by replicating the Fabric project to their respective Composition Studio project as a local copy. As work is performed on the studio project, changes are saved to the local copy. When a developer is satisfied with their changes, they can then submit them to Governance Manager for approval and subsequent publication to the BSR. The changes are submitted to Governance Manager as changelists. A *changelist* is a list of changes to the local copy. An administrator, using Governance Manager, reviews the changes and approves or rejects them. The developer is notified by e-mail of the decision.

Rejected changelists stop at this point. Approved changelists are now presented to the BSR for publication. One last action takes place. The model is checked to be sure that the new changes do not create an integrity breach in the current model. This check is similar to the integrity check that is performed in Composition Studio.

Composition Studio does not allow a developer to submit changes that create integrity failures on their local copy. So you may ask why the integrity check is necessary. The reason that the integrity check is necessary is because collaborative development does not happen in a serialized manner, that is each developer submits changes as they are ready. The changes of one developer might not include changes that another developer makes, and in a collaborative environment, this must be given consideration.

Fabric does provide the ability for developers to update their work and realize the most current model with up-to-date changes. However the reality of collaborative teams is that it is not always possible for developers to work in a serialized manner, regardless of how big or small a development team is. The governance model of Fabric recognizes this and compensates with the model integrity check.

As a caveat, we must also mention that the Fabric integrity check only confirms that the model is in fact sound and complete. It does not and cannot confirm that the latest changes do not impose implications to other changes, for instance say how a policy is realized; therefore, you should review this type of situation as an approval step and confirm it through system testing.

After you pass the integrity check, the updates are published to the BSR. If the updates fail the integrity check, they are rejected.

Figure 4-7 provides a basic illustration of the Governance Manager process flow.

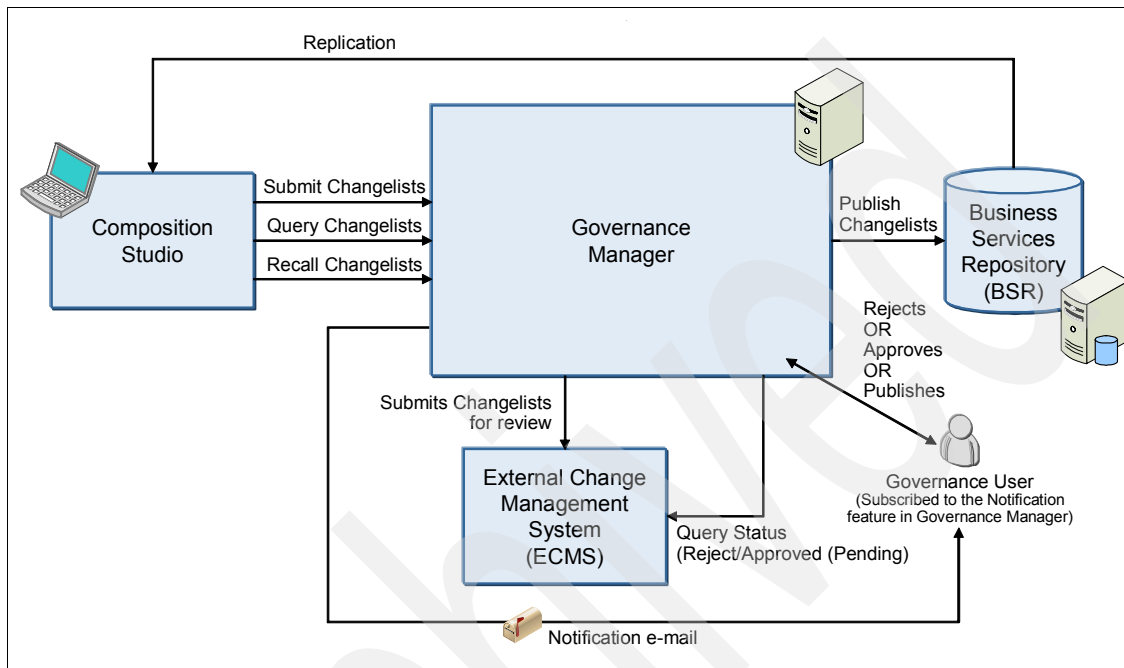


Figure 4-7 Governance Manager process flow

## Subscriber Manager

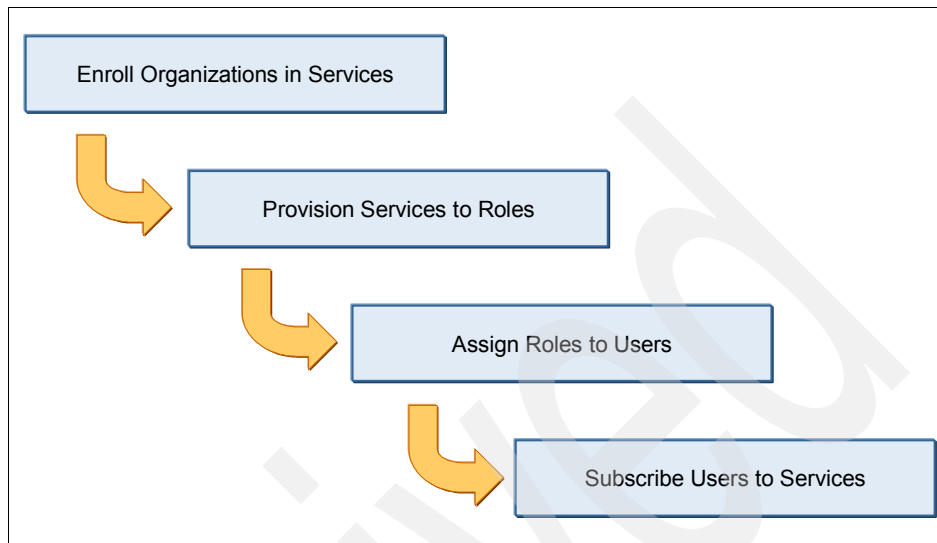
Subscriber Manager administrates and manages business service entitlements using an organizational enrollment and subscription model to assign, create, control, and manage on-demand service packages. A *subscriber* is a person, role, organization, or a machine who is a consumer of a business service.

Subscriber Manager enables business service portfolio provisioning through either self service or assisted provisioning.

Using Subscriber Manager, a service consumer or customer support representative can register subscribers, manage subscriber profiles, and enroll or dis-enroll from business services. Subscriber Manager enables business service portfolio provisioning through either self-service or assisted provisioning and supports auto provisioning and de-provisioning of service entitlements.

The Subscriber Manager model has three basic themes: enrollments, subscribers, and subscriptions. Accenting this theme are organizations, users

and roles. Figure 4-8 shows the typical flow for creating business service entitlements.



*Figure 4-8 Typical flow for creating business service entitlements*

As Figure 4-8 illustrates, the basic process is to:

1. Enroll an organization to a business service.
2. Provision the business service to roles.
3. Assign the roles to users.
4. Subscribe users to services.

Subscription Manager provides administration capabilities to manage enrollments, manage subscribers, and manage subscriptions.

Through federation of external resources, Fabric leverages support of authenticating and authorization systems, such as identity management systems, LDAP, and WebSphere Virtual Member Manager (VMM).

VMM provides a secure facility within WebSphere Process Server V6.1 to support customers' basic organizational entity management needs. Organizational entities refer to entities that are common to most organizations, such as people, login accounts, business units, security roles, and business roles. VMM provides a model of the organizational entities that serves the basic needs of a variety of applications and hides from them the disparate repositories that can be used underneath.

## **Subscriptions**

Subscription IDs are created using Subscriber Manager or by an application that uses the Subscriber Manager's application programming interfaces (API), which are in the SDK. The Subscription ID is the uniform resource identifier (URI) of the user's subscription of the business service.

As business services are invoked, a subscription is provided. The subscription communicates the context of this invocation with respect to the user, their role, organization, affiliation, and channel.

The subscription is key to identifying the context of an invocation and a prerequisite to Performance Manager reporting.

## **Performance Manager**

Performance Manager provides business visibility and monitoring of service-oriented processes and applications for business analysts and IT administrators. It includes multi-perspective views and enables drill-down analysis of events and exceptions in a loosely coupled, service-oriented business ecosystem. Performance Manager can correlate runtime service invocation data with the metadata from the Business Services Catalog to produce valuable metrics on operational and strategic business performance.

Performance Manager includes a set of visibility services that enable administrators to monitor the behavior and performance of their business services.

The following reports are made available with Performance Manager:

- ▶ **Service invocation summary report.**  
Use the service invocation summary report to view the response times of Web services and endpoints that are invoked in the context of a single business-service transaction. You can view the service invocation summary report for business services that are associated with all applications, all application suites, or a type of business service.
- ▶ **Web service performance report.**  
Use to monitor a Web service's capacity to handle transaction volumes, its response times, and availability for business transactions.



- ▶ **Endpoint service report.**  
Use to monitor an endpoint's capacity to handle transaction volumes, its response times, and availability for business transactions.
- ▶ **Service utilization report.**  
Using the service utilization business service, you can analyze business service use in an enterprise. The report displays key details about the usage of a business service, such as its volume, cost per unit, cost modifier, and the total cost in a particular period.

### ***Performance Manager reports prerequisite***

Performance Manager requires that a subscription ID is included with the business services invocation.

Performance Manager tracks business service enrollments and other transaction data that is required for reporting through an identifier that is supplied to the system. Fabric uses a super-set of this information, termed as subscription, to indicate to its components the use of an enrolled business service on a specific channel by a particular user playing one or more roles within an organization. Using the subscription ID, Performance Manager extracts the data that is required for reporting from the received subscription information and stores it in a database. Without a subscription ID, an invocation cannot be tracked, and the Performance Manager cannot provide performance details about the invocation.

## **4.3.3 Composition Studio**

Using the Eclipse-based authoring tool, Composition Studio, you can model, publish, and manage business service models, interfaces, policies, and service portfolios into service-oriented business processes and composite business applications.

Fabric business service models provide the domain for modeling composite business applications. By using Composition Studio and the Fabric Administrative Console, you can create business service model instances and store them in the Business Service Repository as metadata solutions. After the metadata is available in the Business Service Repository, other developers can use the metadata to understand solution's components, reuse solution's components, and refine solution's behavior.

As the metadata authoring tool, Composition Studio is the starting point for Fabric's comprehensive metadata governance life cycle, which you can use to submit changelists against the Business Services Repository.

Composition Studio is installed as a set of plug-ins to the WebSphere Integration Developer environment and provides a Business Service perspective.

Composition Studio provides a comprehensive view of a composite business application, which includes the application suite, business services, composite services, interfaces and endpoints, and policies. Figure 4-9 provides a view of a Fabric project as seen through the business services perspective of Composition Studio.

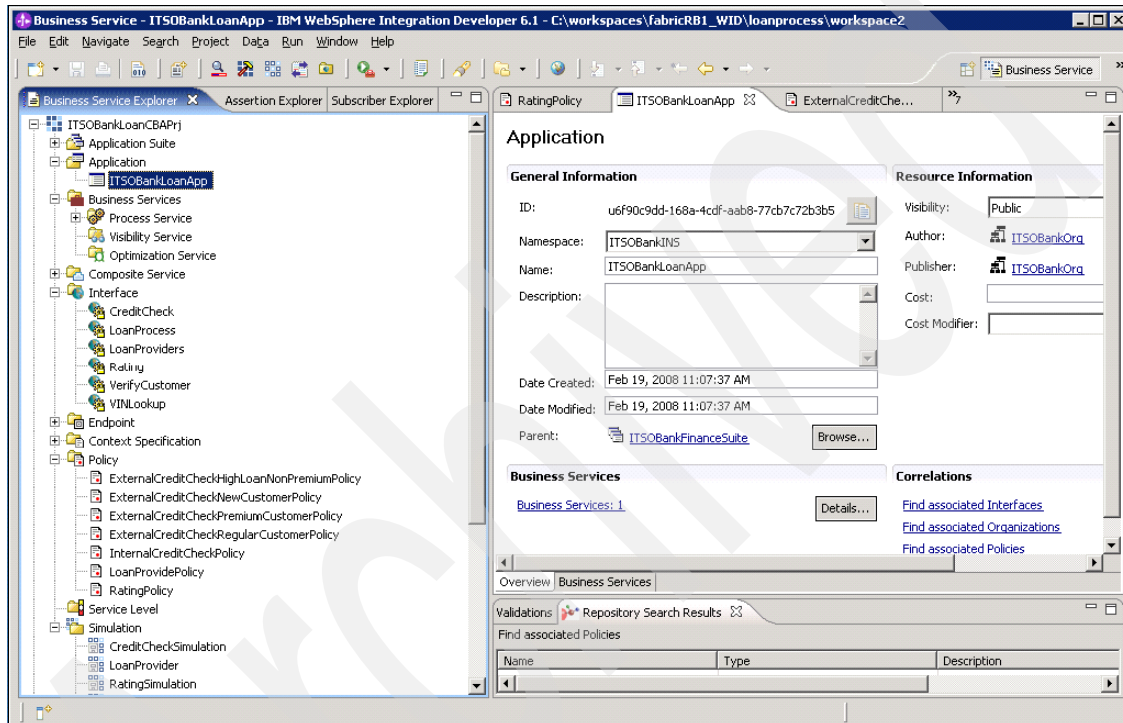


Figure 4-9 Composition Studio

A developer who is working with Composition Studio does not interact directly with the Business Services Repository; instead, the developer works with a replicated copy of the Business Services Repository models. This technique allows Fabric to support collaborative development in a very scalable and manageable manner.

A developer starts by first searching, locating, and then replicating a model to their respective development environment. After the replication is complete, all work that developer does in Composition Studio is saved to the local copy. The updates are saved as changelists. The developer can submit these changes for approval and subsequent publication to the business services model. After which

the developer can update and synchronize their copy with the core and base business services model.

After the developer replicates a project, they can begin to edit and author the metadata. The Fabric business service repository represents the data in a prebuilt model, and this model provides the basis for the Business Service perspective.

WebSphere Integration Developer is an Eclipse-based tool. Eclipse tools consider a perspective as the visual container for a set of views and editors that are made available for a designer or a developer. An eclipse tool has many perspectives, and when you use Fabric, Composition Studio provides the Business Service perspective.

The Business Service perspective provides an Explorer type view of a project and includes views and editors for all of the resource types, such as applications, applications suites, business services, policies, endpoints, composite services, interfaces, and so on.

A Fabric designer can also create and execute *correlations queries* and *policy simulations*. Correlations queries look at the business service model and return a result set that indicates the relationships of items that are specified within the query, for example, a simple correlation can be to identify all of the endpoint implementations that are associated with business service specification. A correlation query is executed and the results are returned for inspection. A number of correlation queries are pre-established and provided with Fabric. A developer can also build specific correlations queries and add them to their Business Service perspective. These custom correlations can offer significant benefit when to understanding current service usage.

Policy simulations are invaluable tooling that allow a Fabric developer to enter simulated context and content and then witness through simulation which policies would have been chosen and used to create the selection or composite policy. From the selection, policy developers can see which endpoints were considered a candidate list and ultimately chosen.

Policy simulations can be saved and stored for later retrieval and shared with other developers.

#### 4.3.4 Business Services Repository

The BSR is an ontology-based business service model repository that captures information about business services, business policies, and service subscribers. It supports discovery and federation of data from WSRR and LDAP systems.

The BSR is a core component of Fabric and provides for the storage and modeling of all Fabric metadata, policies, and subscription information. The BSR's unique and comprehensive information store is built as an ontology. The ontology describes the vocabulary, structure, constraints, relationships, and behaviors and provides a more thorough representation and understanding of a problem domain. The BSR is primarily a repository of business services metadata only. The BSR does not store artifacts, such as Web services description language (WSDL) files, and the BSR is persisted to a relational database management system (RDBMS).

Other characteristics of the BSR are:

- ▶ Full support for versioning and configuration of deployment environments.
- ▶ Conflict detection during collaborative development.
- ▶ Powerful search, dependency, and impact analysis through correlation queries.

The BSR is modeled using Resources Definition Framework (RDF) and Web Ontology Language (OWL), which makes it easily extensible.

## Namespaces

The internal model that the BSR uses is called the business service model. A namespace provides a means of grouping objects in the business service model, which makes the system more modular. All objects that are in the business service model must belong to a namespace. A namespace is identified by its Uniform Resource Identifier (URI).

The BSR is not limited to a single model; instead, it supports storage and retrieval of models under a given namespace.

In addition to basic namespace partitioning, the BSR also adds the concept of namespace type. A *namespace type* determines which type of information is stored under a particular namespace in the system, for example, a namespace of type *schema* contains only classes and properties that define the business service model, while a namespace with the type *instance* contains only objects of those classes.

Various types of namespaces are supported, and a user can create some namespaces, while others are system generated.

Namespaces that you can create are:

- ▶ Instance  
Used to store business service model instances.

- ▶ Schema  
Used to store schema information.
- ▶ Enrollment  
Used to store business service entitlement information. The system creates this namespace at the time of configuring an external LDAP project.

Namespaces that the system creates are:

- ▶ Sponsored  
Stores supporting instance information for federating objects. The system creates this namespace at the time of configuring an external WebSphere Service Registry and Repository project.
- ▶ External  
Used to federate objects from external sources.

### 4.3.5 Dynamic Assembler

The Dynamic Assembler is packaged as an SCA component and is managed by the SCA runtime within the WebSphere Process Server. Dynamic Assembler determines which endpoints are to be used at runtime based on the incoming request and current circumstances or the context of the request.

Dynamic Assembler looks to the business service that is requested, relevant policies per the context, and from that assembles one policy, the composite policy or the selection policy. After the selection policy is determined, candidate endpoints are chosen that match the criteria that is established by the selection policy. The candidates are ranked and tiered. The most appropriate candidate is selected as the binding for the current request. Figure 4-10 on page 72 presents a sequence diagram of events that are taken during the Dynamic Assembler process.

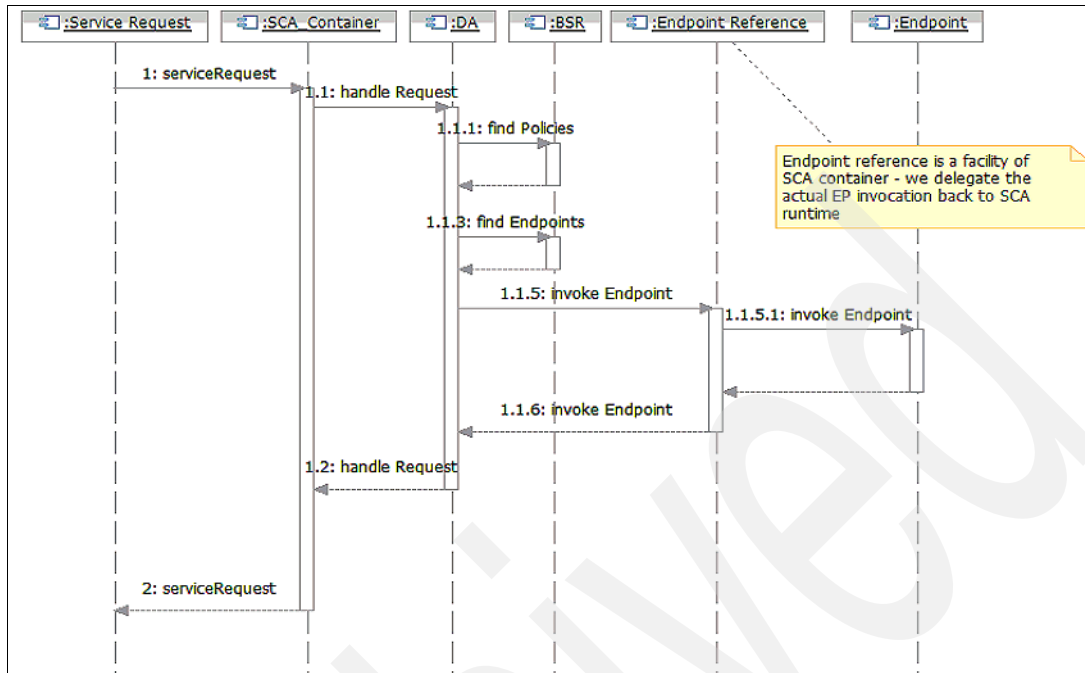


Figure 4-10 Dynamic Assembler sequence diagram

The typical usages of Dynamic Assembler include making endpoint decisions that are based upon:

- ▶ The interface that is requested and the cost of the endpoints that are associated with the interface.
- ▶ The delivery channel from which the request was received.
- ▶ The invocation request context, such as the loan type that was requested.
- ▶ A time sensitive feature of the request, for instance, endpoint availability.

Endpoint selection might be simulated at design time using the policy simulation capabilities of Composition Studio. You can log and review the runtime decisions that Dynamic Assembler makes.

## Dynamic Assembler and SCA

Fabric utilizes the Service Component Architecture (SCA) programming model. SCA was developed to simplify the integration between business applications and development of new services. SCA separates application business logic and the implementation details, providing a model that defines interfaces, implementations, and references in a technology neutral manner.

This programming model is a natural foundation for Fabric's approach to building composite applications. Dynamic Assembler is a first class SCA primitive and is declared in the context of an SCA module as a custom component, as shown in Figure 4-11.

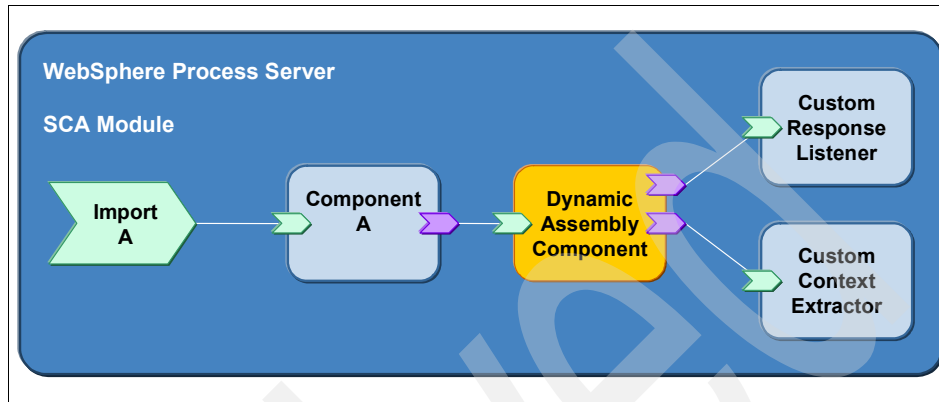


Figure 4-11 Dynamic Assembler SCA first class primitive

### **Dynamic Assembler extensions**

You can modify the behavior of Dynamic Assembler at different stages of the process using extensions or plug-ins. In the next section, we provide a list of extensions and their intended uses.

#### **Context extractor**

In many cases, the data that the Dynamic Assembler needs to make decisions is in the actual request or response message body. However, the Dynamic Assembler cannot operate directly upon the message body; instead, it only uses data that is in the context. Thus, there is a type of plug-in that is known as a context extractor. These plug-ins are invoked early in the life cycle of a Dynamic Assembler request, whose job is to look through the request message body and insert (or update) data items in the context based upon what is found there.

Use a context injector when message content is considered a critical aspect of the endpoint selection process. It must be extracted from a message and injected into the context.

#### **Policy guard**

A policy guard plug-in can completely stop the Dynamic Assembler from continuing to locate and invoke an endpoint for a request, which causes a request to fail, which means not invoke an endpoint.

Policy guards are invoked with the original request and the selection policy. They can look through the request and the selection policy and decide whether or not to allow the request to continue.

Use a policy guard for exception processing. Also, before you implement a policy guard, investigate the possibility of using a reject-always-policy as an alternative. A reject-always-policy offers similar functionality without the need to modify Dynamic Assembler behavior.

### ***Endpoint filter***

An endpoint or candidate filter plug-in is invoked after the Dynamic Assembler performs all of its policy matching and has given a policy matching score to all of the endpoints, but before an endpoint is selected. The endpoint is invoked with the entire list of all of the candidates that were considered. For each such endpoint, the following items are also provided:

- ▶ The cost (a configuration item)
- ▶ The policy matching score (calculated)
- ▶ An integer tier, such as a numeric bucket into which this endpoint falls (set by the plug-in)

Whether or not the endpoint is rejected (set by the plug-in), the plug-in is then free to take the following actions:

- ▶ Reject an endpoint (with a string that describes the reason).
- ▶ Assign a tier for the endpoint.

If an endpoint is rejected, it is no longer considered as a potential handler for the request; otherwise, if the plug-in assigns tiers, then all other scoring is ignored. Every endpoint in the same tier is considered equivalent for the purposes of finding an endpoint to handle the request.

Use endpoint filter to change the tiering behavior of Dynamic Assembler, for instance, Dynamic Assembler selects the lowest cost endpoint, perhaps there is a reason to reverse this and select the highest cost endpoint.

### ***Response listener***

To allow possible modification of the response, response listener is called after an endpoint responds.

If this plug-in makes no changes, then it either returns the OutboundResponse that it passed or it returns a null. Otherwise, it makes modifications to the InboundResponse and then returns the modified OutboundResponse object.



Use response listener to modify the returned message or possibly trace a returned message with means other than say typical Dynamic Assembler debugging capabilities.

### ***Assembly event listener***

Assembly event listener is invoked after all ResponseListeners are run or if an endpoint is not found, so that notifications are sent to other frameworks or applications. The Assembly event listener has two methods: `handleInvocation()` and `Endpoint NotFoundEvent()`.

The `handleInvocation` method is called when an endpoint responds and all ResponseListeners are called.

This `handleEndpointNotFound` method is called when no endpoint is found to handle the request.

Both methods return nothing, nor can they affect Dynamic Assembler processing.

Use when additional Dynamic Assembler debug information is required.

### ***Extension execution***

Dynamic Assembler extensions are processed at different times during the endpoint selection process. See Figure 4-12 on page 76, for a Dynamic Assembler extensions activity diagram.

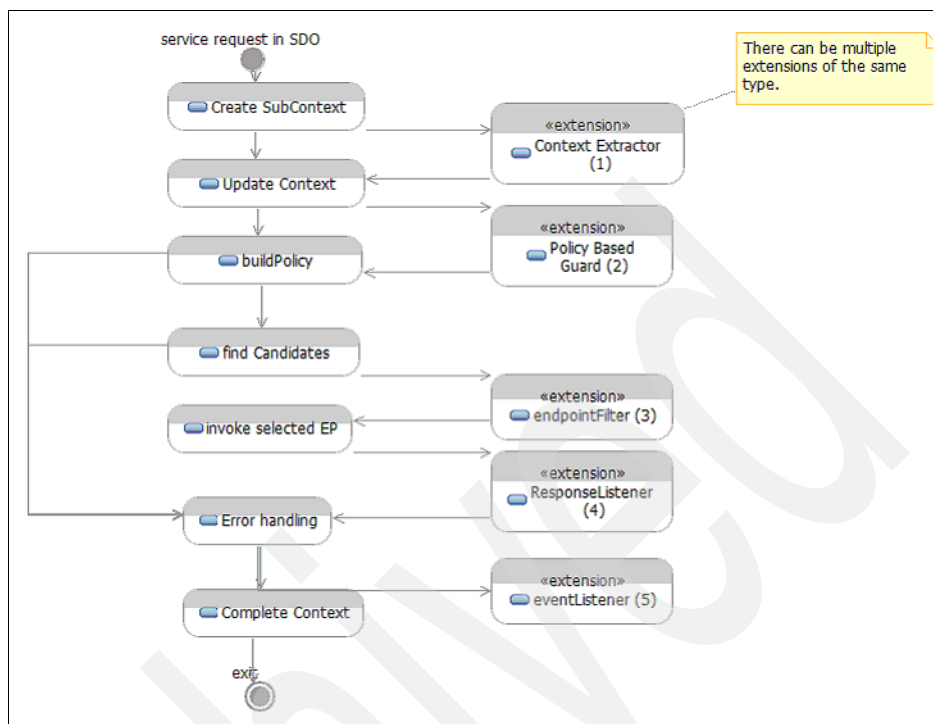


Figure 4-12 Dynamic Assembler extensions activity diagram

Policy plug-ins are not necessarily called for every request, for instance, if result caching is turned on, then the only time that policy plug-ins are called is when there is a result cache miss. This is for performance because it allows a significant portion of Dynamic Assembler processing to be bypassed for frequently-invoked endpoints. If result caching is turned off, policy plug-ins are called every time.

## Dynamic Assembler configuration

When using Dynamic Assembler, the developer has a few options to set regarding the runtime configuration:

### ► Enable Result Caching

Controls whether or not Dynamic Assembler component is required to recalculate the selected endpoint each time.

- ▶ **Fire Invocation Events**  
Controls whether the Dynamic Assembler component imits events that summarize each invocation.
- ▶ **Enable Verbose Logging**  
Controls whether or not the Dynamic Assembler component sends detailed debug information to the application server logs. This setting only turns logging on for an individual Dynamic Assembler component, if you want to turn on verbose logging for all Dynamic Assemblers, you can do this using settings in the integration console.

## 4.4 Fabric polices

You compose Fabric policies using the Composition Studio policy editor. You create the policies in a Composition Studio project, and they are an integral part of the business services model, which is the core model of the Business Services Repository.

Policies key attributes:

- ▶ **Target**  
The initial context that is relevant to a policy.
- ▶ **Start Date**  
The date on a policy that becomes effective and starts being enforced.
- ▶ **End Date**  
The date a policy expires and is no longer enforced.
- ▶ **Priority**  
The priority given to a policy. If a policy conflict exists the policy with the highest priority, it is given first consideration.

### ***Context, content, contract***

The policy editor models policies under three basic conditions: context, content, and contract, as shown in Figure 4-13 on page 78.

**Policy**

**General Information**

ID: u2861f9d8-cea0-4570-aa75-e5cd15d2b8ec

Namespace: ITSOBankINS

Name: ExternalCreditCheckHighLoanNonPremiumPolicy

Description:

Date Created: Feb 19, 2008 11:07:37 AM

Date Modified: Feb 19, 2008 11:07:37 AM

**Resource Information**

Visibility:

Author: ITSOBankOrg

Publisher: ITSOBankOrg

**Policy Information**

Target: CreditCheck

Start Date: Feb 21, 2008

End Date: Feb 28, 2010

Priority: 0

**Context**  
Contexts: 0

**Content**  
Contents: 2

**Contract**  
Assertions: 1

**Policy Conditions**

Figure 4-13 Policy conditions: context, content, and contract

► Context

The context specifies the perspective condition to be applied, such as this policy applies only for users of role type agent, or this policy only applies if the incoming channel request was EDI. A context specification can be complex and requires that a group of conditions are met, such as the role must be agent, and the channel must be EDI, or it can list a number of conditions and only require that one of them be present.

► Content

The content condition specifies information that be must be considered and is part of the business service request, for example, if the business service was loan rating and a policy is based on say the loan amount. A loan amount condition is specified within the policy model. The loan amount is expected to accompany the loan service request.

► Contract

The contract condition articulates facts about the situation when the circumstances match to the context and content that are stated in the policy. That is to say, if the context and content conditions are met, the policy is to meet the items specified by the contract. An example of a contract condition for a loan rating service might specify that the loan service that is requested is

a Luxury “loan product”, when the content condition “loan amount” is greater than 500,000.

### **Natural ordering**

Within a Fabric project there is generally more than a few policies. A policy also has a scope, which is labeled as the policy target. You can use the target to further qualify a policy, if a conflict arises. The target is a form of a context condition. The scope is defined using what is called *natural ordering* within Fabric. Natural ordering within Fabric takes on a hierarchy of very general to very specific. Where a very general target might be a Fabric project or application suite, and a very specific target might be an organization or role, as shown in Figure 4-14.

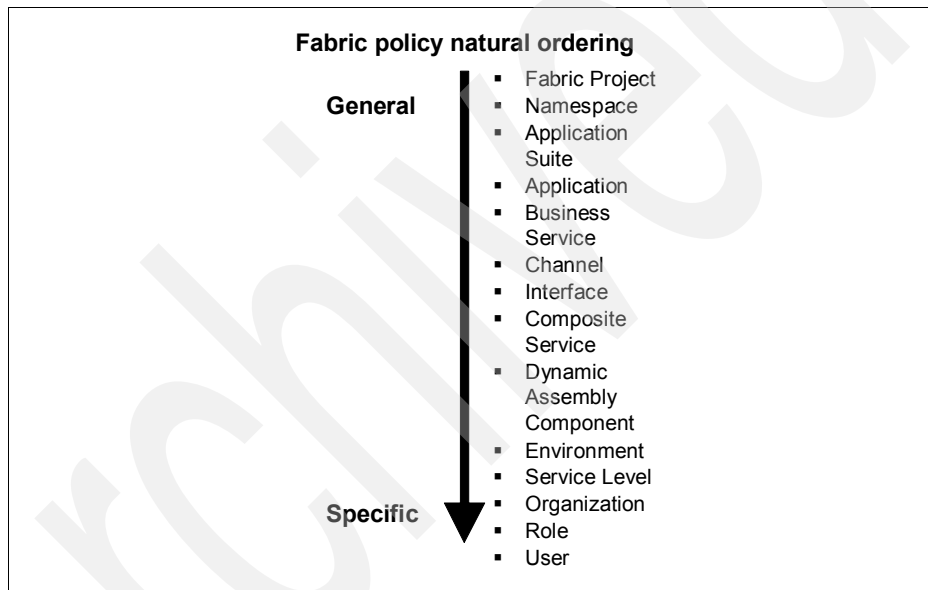


Figure 4-14 Fabric policy natural ordering

Now let us consider that two policies exist that declare that a loan is of type luxury, if the loan amount is greater than some value. In this case, the scope or target of the policy determines which policy to consider. The policy with the most specific target is used. If Policy-A has a target of, say, application, and Policy-B has a target of organization, policy-B is chosen because organization is a more specific scope than application.

If natural order cannot resolve a policy conflict, then locks and priority can also be considered.

A *policy lock* allows a designer to set a very general target scope on a policy and declare that the policy cannot be subject to being overridden by another policy at a more specific level. It locks the policy in regardless of other policies. This capability is useful to declare a new corporate policy that has, say, no exceptions.

A policy priority can also be specified, and in the case of conflict, the policy with the highest priority is chosen.

If a policy conflict still cannot be resolved, a policy conflict exception occurs and the endpoint selection fails.

### ***Policy simulation***

Avoiding policy conflicts might not be possible but understanding how and why they could happen is easily understood using the *policy simulation* capabilities that are provided in Composition Studio. Refer to Figure 4-14 on page 79.

The policy simulation allows a designer to simulate the behavior of a policy as applied by the Dynamic Assembler.

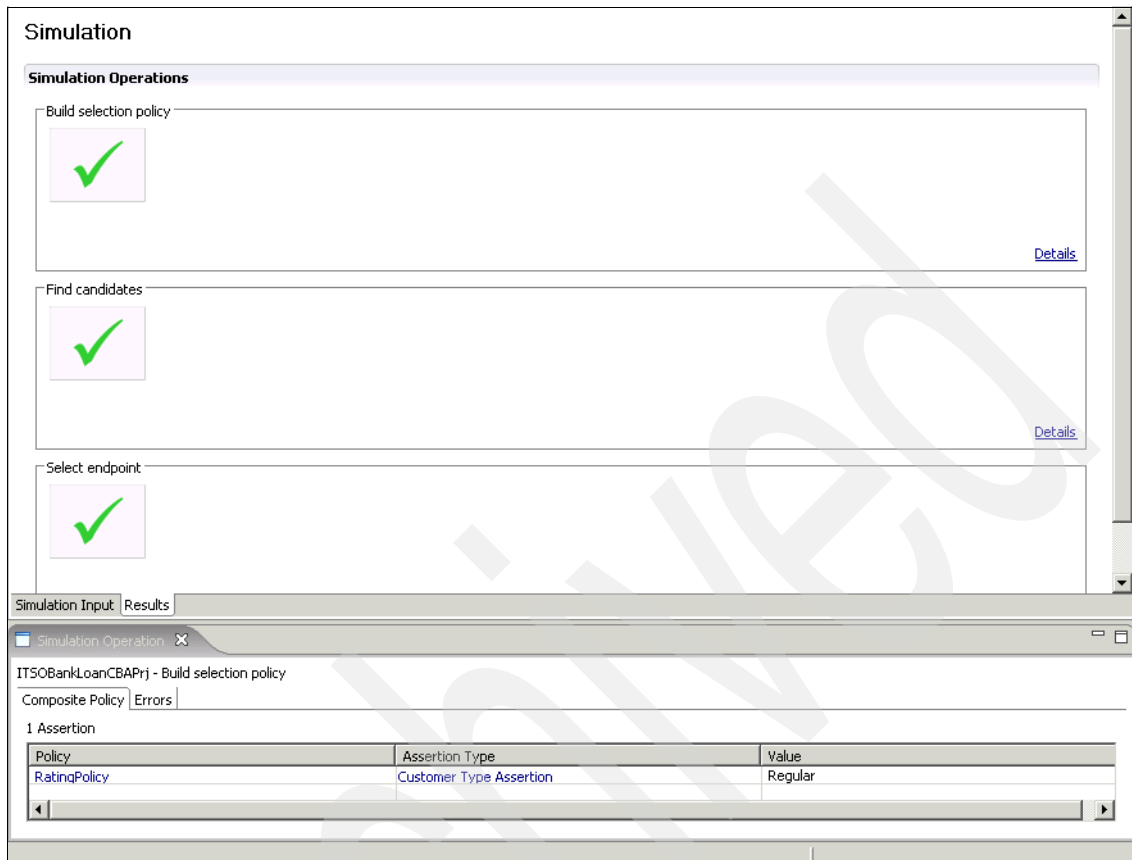


Figure 4-15 Policy simulation

Dynamic Assembler performs the following four steps as it makes an endpoint determination:

1. Build a selection context.

A context is determined based on context and content dimensions.

2. Build a selection policy.

The selection policy is composed from all applicable policies based upon the determined context.

3. Find candidate endpoints.

All endpoints that meet the contract conditions as specified by the selection policy are selected and ranked.

4. Select an endpoint.

The active endpoint with the highest ranking is selected.

The policy simulator executes these steps based on conditions that are set by the developer who requests the simulation. Information about the results of each step is made available to the developer that explains how Dynamic Assembler makes its endpoint decisions. Information gathered here can greatly assist designers in creating policies that match the business or technical intent.

You can save results from policy simulations for future reference.

## 4.5 Business Services Fabric Modeling Tool

The *Business Services Fabric Modeling Tool* provides a modeling environment that allows business service architects and developers to use industry standards and models to create Fabric BSR ontology model extensions and other content. The tool provides a repeatable approach to model Fabric concepts from an industry business glossary. The Business Services Fabric Modeling tool is provided as a set of plug-ins for Rational Software Architect V7.

You can use the Business Services Fabric Modeling Tool to:

- ▶ Transform industry standard schemas to business glossary terms in UML notation.
- ▶ Define Fabric assertions, channels, and roles using UML notation.
- ▶ Define business capabilities and process map instances using UML notation.
- ▶ Transform UML models to OWL files that WBSF can import.

The Business Services Fabric Modeling Tool provides the following functions:

- ▶ UML profiles that provide stereotypes for Fabric concepts.
- ▶ XSD to UML import wizard for industry standard concepts.
- ▶ UML to UML transformation to convert glossary terms to ontology classes.
- ▶ UML to OWL transformation to generate OWL files that define Fabric assertions.
- ▶ Channels, roles classes, business capabilities, and process map instances.





## Part 2

# Building Fabric solutions

*Archived*



## Business scenario for this book

In this chapter, we introduce a fictional bank, the ITSOBank, whose business scenario is used as a working example throughout this book. This financial organization deals in functional areas, such as customer account management, electronic fund transfer, loan processing, and so on. For the subsequent chapters in this book, we consider vehicle loan processing as a sample business scenario.

This chapter contains the following topics:

- ▶ ITSOBank's current business scenario for the Vehicle Loan processing system
- ▶ Analyzing the business scenario
- ▶ The business and IT challenges that the bank currently faces

In subsequent chapters, we show how ITSOBank addresses the business and IT challenges that they face through the implementation of a WebSphere Business Services Fabric solution.

## 5.1 Introduction

ITSOBank is a fictional financial institution in North America that offers a variety of products that are related to account management (deposits) and loan processing. As part of ITSOBank's account management portfolio, ITSOBank provides saving account and checking account capabilities to the customer. ITSOBank currently specializes in providing loan products, such as personal loans and vehicle loans. For this book, we consider the vehicle loan processing system as a business scenario.

## 5.2 Business scenario overview

As a step towards modernizing the IT infrastructure based on SOA, ITSOBank decides to implement a Loan Processing System (LPS). In this section, we elaborate on the use case model that depicts the business use cases and the actors that are involved in the system. We also illustrate the business process model.

### 5.2.1 Use case scenario

Figure 5-1 on page 87 depicts the use case model for the simplified vehicle loan processing system. A Customer or Loan Officer applies for the loan. After the loan application is received, ITSOBank performs the customer verification, credit check, Vehicle Identification Number (VIN) lookup, and calculates the risk rating. Based on the risk rating score, an appropriate loan provider is selected to furnish the loan. We depict and explain the business process in 5.2.2, "AS-IS business scenario" on page 89.

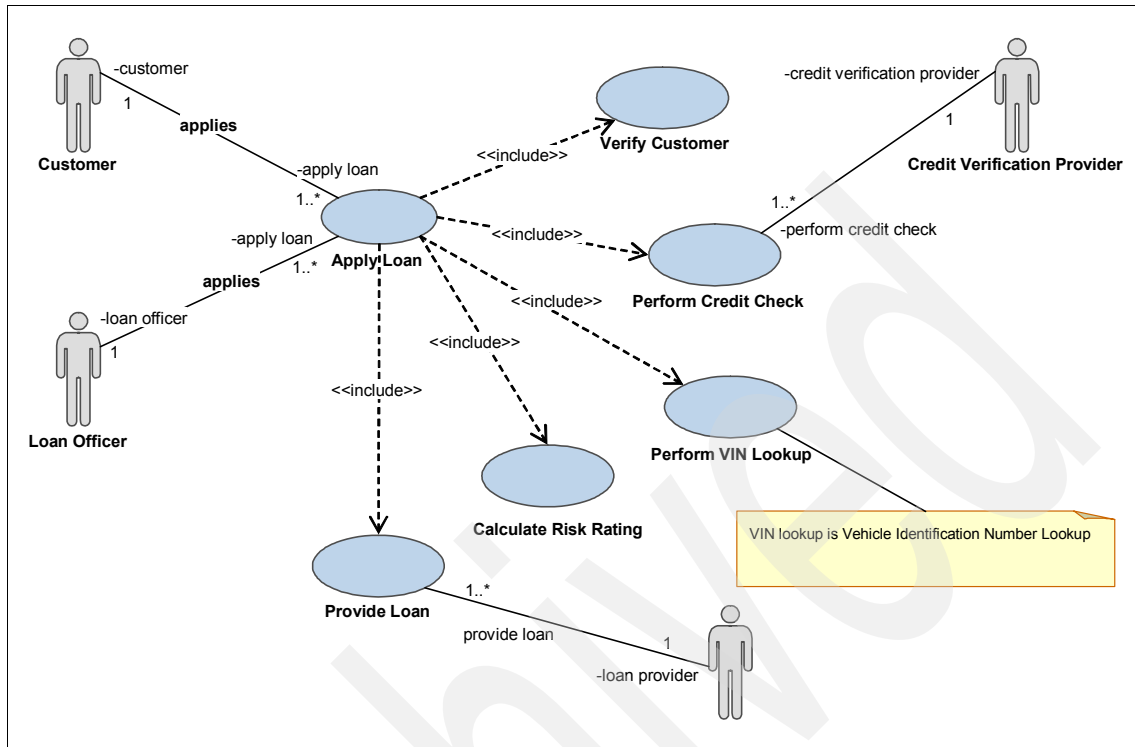


Figure 5-1 Loan Processing Use Case Model

The following list contains the use cases that constitute the loan processing system:

- Apply Loan
- Verify Customer
- Perform Credit Check
- Perform Vehicle Identification Number (VIN) lookup
- Calculate Risk Rating
- Provide Loan

The actors that participate in the loan processing system are:

- Customer

Performs the financial transaction with the bank for applying for a vehicle loan. ITSObank classifies a customer in three categories: Premium, Regular, and New. Premium and Regular customers are the existing customers who have either a loan account or a savings account with the bank.

- ▶ **Loan Officer**  
Applies for loans on behalf of the customer.
- ▶ **Credit Verification provider**  
External system that provides the credit check business functionality.
- ▶ **Loan Provider**  
Entity that finances the loan. There are situations where the external banks (lenders) are interested in providing the loan to the customer through ITSOBank. Such loan providers are known as external loan providers.

As shown in Figure 5-2, the bank provides the internet as a channel for customers to perform financial transactions with the bank. A Loan Officer in the bank can access the loan processing system through the internet as well as ITSOBank's internal loan application interface—the business-to-business (B2B) channel.

The loan processing system uses the credit verification provider to provide the credit score of the customer who applies for the vehicle loan. Based on the credit score and rating score (high/medium/low rating), the loan processing system selects an appropriate loan provider to finance the loan.

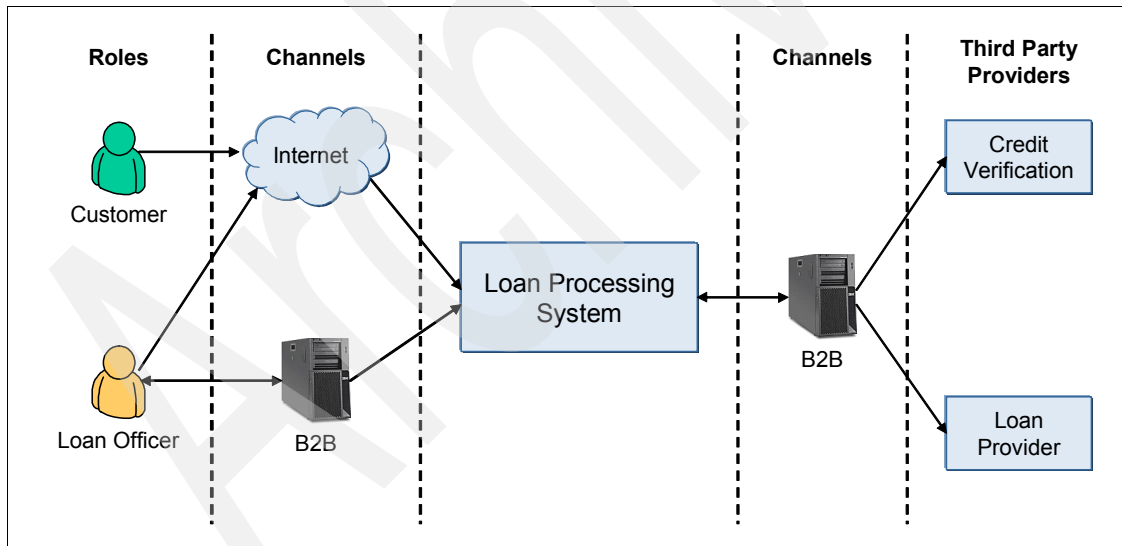


Figure 5-2 High-level business context diagram

### 5.2.2 AS-IS business scenario

ITSOBank's business analyst, Joe, and his team define a business process model for the Vehicle Loan processing system, as shown in Figure 5-3 on page 90:

1. The customer, through a self-service portal (online), can apply for the vehicle loan or can request that a Loan Officer apply for the vehicle loan on behalf of the customer. The Loan Officer can apply for the loan through the self-service portal (online) or through the bank's internal loan application interface (B2B).
2. After the loan application is populated, the vehicle loan processing process is initiated by performing the customer verification. One of the tasks involved in customer verification is to identify the type of customer: Premium, Regular, or New.

Figure 5-3 on page 90 illustrates the Loan Processing Business Process model.

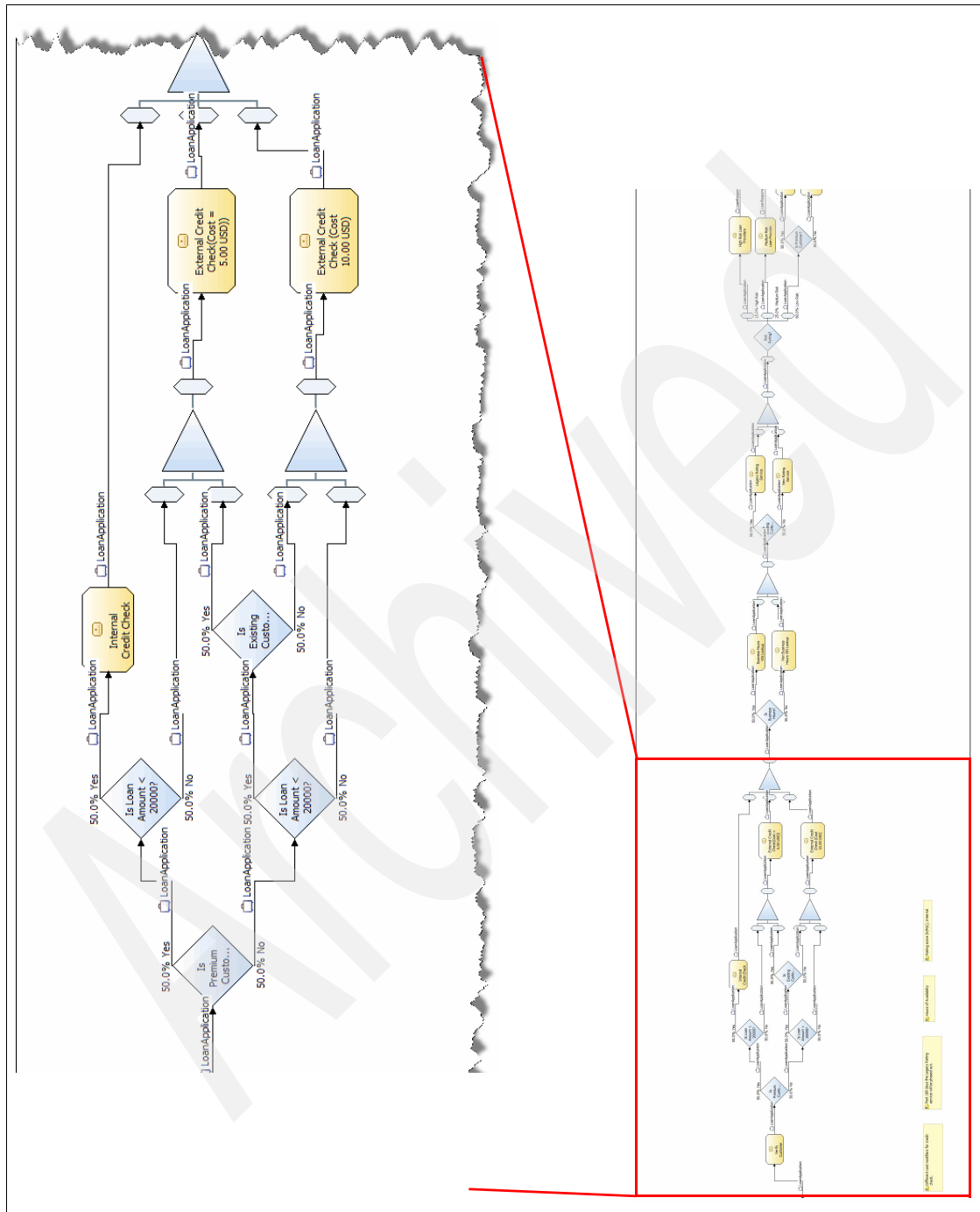


Figure 5-3 Loan Processing Business Process Model - Part I



3. The credit check (credit verification) is performed. Currently there are two types of credit checks: internal and external. The internal credit check service is provided by ITSOBank, and the processing cost is minimal (2 USD per transaction). ITSOBank also has agreements with third-party credit verification agencies for credit verification. Each of these agencies has different processing costs, for example, one external agency charges \$5 United States Dollars (USD) for a credit check, and another agency charges \$10 USD. Based on the type of customer and loan amount, an appropriate credit check service is invoked.

Table 5-1 enumerates the decision logic for selecting an appropriate credit check service, based on the customer type and loan amount.

Table 5-1 Decision logic for credit check

Customer type	Loan amount	Credit check service
Premium	< 20000 USD	Internal (Processing Cost = 2.00 USD)
Premium	>= 20000 USD	External (Processing Cost = 5.00 USD)
Regular	< 20000 USD	External (Processing Cost = 5.00 USD)
Regular	> =20000 USD	External (Processing Cost = 10.00 USD)
New	< 20000 USD	External (Processing Cost = 10.00 USD)
New	> =20000 USD	External (Processing Cost = 10.00 USD)

4. As shown in Figure 5-4 on page 92, the Vehicle Identification Number (VIN) Lookup is performed. The process involves invoking an appropriate VIN Lookup service that is based on the time of request - business or non business hours. The VIN Lookup service that is available with the bank can be accessed only in business hours (9:00 AM to 5:00 PM). To maintain business continuity, ITSOBank uses an external VINLookup service that is available in non business hours (5:00 PM to 9:00 AM and weekends).
5. The results from the VIN Lookup are sent to the Rating service to calculate the risk rating of a customer. Currently ITSOBank uses a COBOL-based rating system and faces issues in terms of maintaining it. Hence, the ITSOBank's CIO made a decision that the bank will introduce a new Rating system that is more efficient and easier to maintain. A decision was made to have the existing customers of the bank routed through the established rating system, and new customers are routed to the new rating service. After 90 days, existing customers are also routed to the new rating system, and the established rating system is phased out.

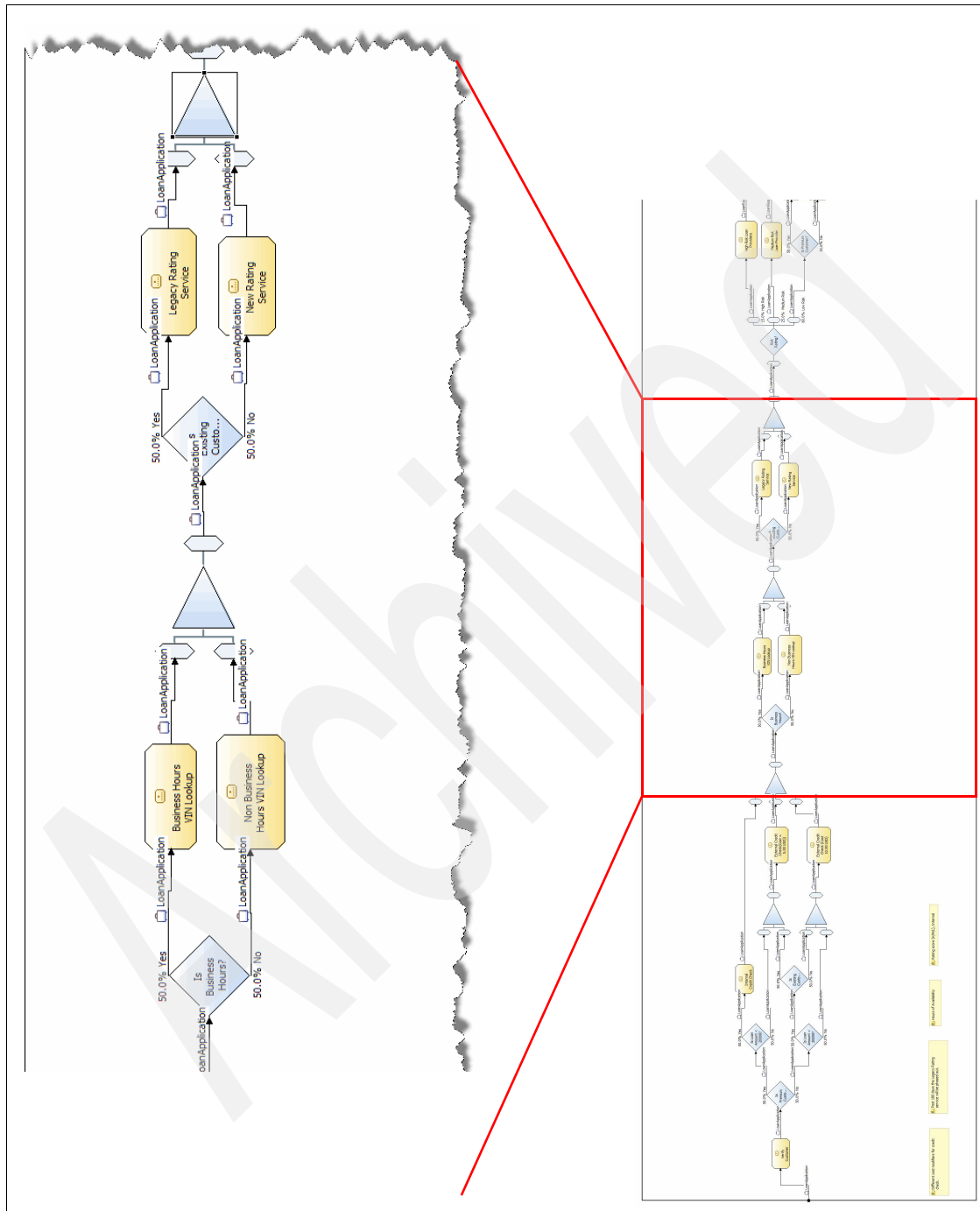


Figure 5-4 Loan Processing Business Process Model - Part II

6. As shown in Figure 5-5 on page 94, based on the rating score that is obtained from the rating system, an appropriate loan provider is selected. Typically, a rating score is a combination of letters and digits, but for simplicity we have the rating score as high, medium, and low. Based on the rating score, the loan provider's interest rates will differ. Premium customers with a low risk rating receive the lowest rate of interest on a loan. Table 5-2 provides sample interest rates that are based on the rating score and customer type.

*Table 5-2 Rate of interest based on the rating score and customer type*

Rating score	Customer type	Rate of interest (%) per annum
Low	Premium	4.565
Low	Regular OR New	6.850
Medium	Any (Premium, Regular OR New)	8.585
High	Any (Premium, Regular OR New)	10.545

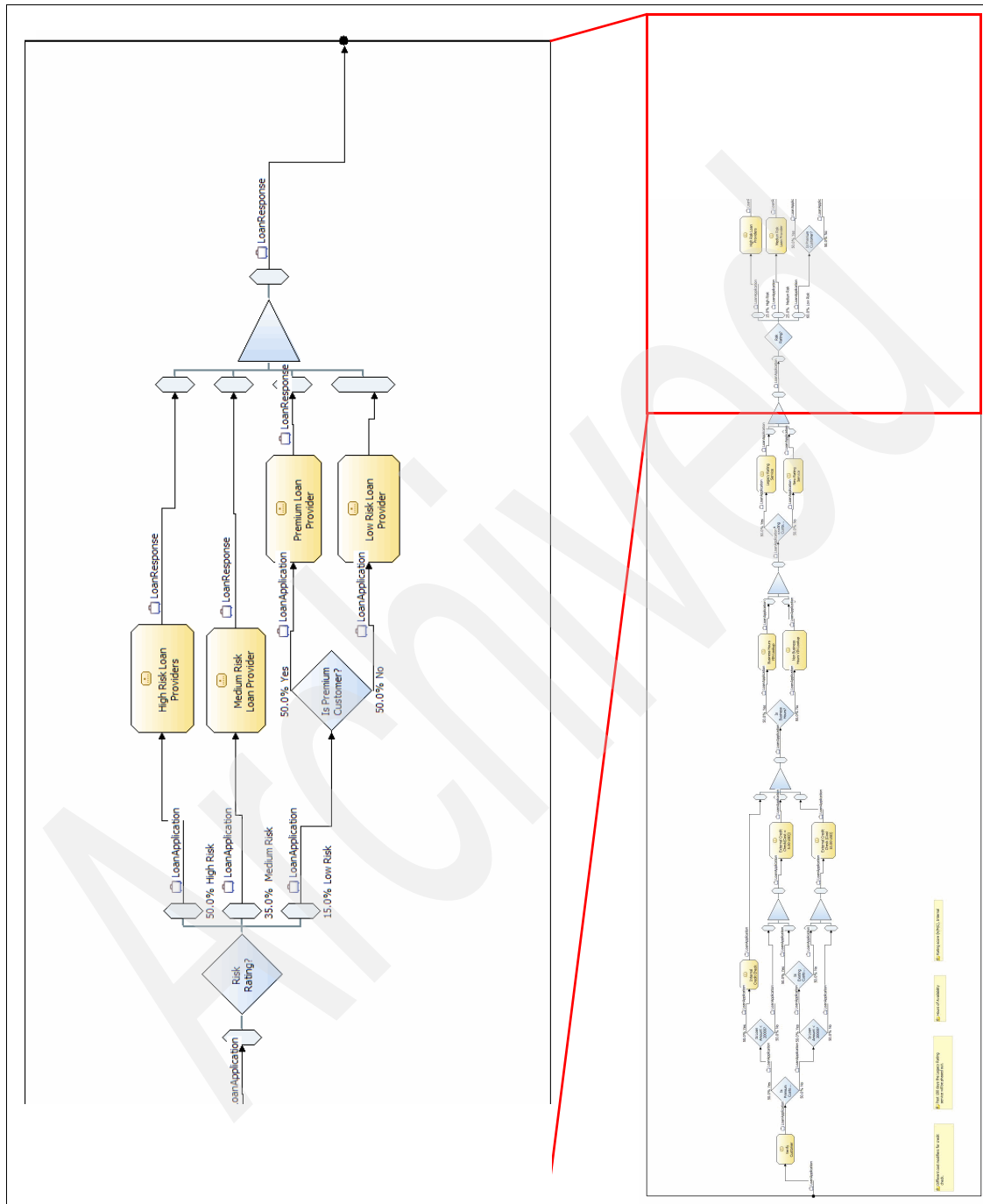


Figure 5-5 Loan Processing Business Process Model - Part III

The response from the loan provider containing the loan status, loan amount sanctioned, and the rate of interest is communicated to the customer and the Loan Officer.

**Note:** For the purposes of this book, we assume that the interface definition for some of the services, such as Credit Check, VIN Lookup, and Loan Provider, are similar. However, in reality, different available techniques should be used to map the service interface definitions, as exposed by different service providers, to a common interface definition.

## 5.3 Business challenges

ITSOBank is faced with many business challenges to keep the bank competitive. The following list contains a few key business challenges that the bank currently faces:

- ▶ Rapid changes in business

ITSOBank's business experiences frequent changes to maintain their competitive edge in the market. A key question is how the bank can launch new and differentiated products into the market? The bank frequently has to consider changes, such as the addition of new channels for customers to apply for loans or the inclusion of new loan products, such as housing loans.

- ▶ Simplify the process changes and increasing business agility

The current loan processing system, as described in 5.2.2, "AS-IS business scenario" on page 89, has a lot of variabilities that are embedded in the process. Changes in the process require that the IT team builds a new process model, for example, two weeks ago the Business Analyst team introduced a new service that differentiates between Premium and Regular customers. The IT team had to make the changes in the business process to accommodate the business requirement.

ITSOBank needs the ability to extract and separate business-level information from the process to keep the process simple and flexible.

- ▶ Increase business agility and flexibility

ITSOBank needs to make their business model more agile and flexible to absorb the change in business demand, for example, ITSOBank needs a mechanism to specify a policy that will allow a Premium customer to have priority over the Regular customers of the bank.

- Innovate the business process to remain competitive

Since the ITSOPBank is making its way in the lending market, one of the challenges is to survive in the market and to offer innovative products to the customer. ITSOPBank is in the process of exposing the loan application through multiple channels, such as hand held devices (PDA), Interactive Voice Response (IVR), and so on. To do so, the AS-IS process needs to be modified with the implementation logic, specific to the different channels.

- Provide incremental business changes

ITSOPBank plans to introduce the loan application to the various groups of users that can be based on either the geography, branch offices, and so on. ITSOPBank needs the ability to manage the subscriptions (entitlements) to new sets of users that are based on roles, geography, and so on.

- Execute growth strategies using merger and acquisition activity

ITSOPBank announced an acquisition of a mortgage lender to accelerate its market share, increase its number of innovative products, and capitalize on the acquired company's state-of-the-art lending platform. The acquisition becomes final within nine months.

## 5.4 IT challenges

The following list contains a few key IT challenges that the ITSOPBank faces:

- Costly and time consuming business process redeployment - increasing the complexity and costs associated.

Whenever there is a change in the business, the corresponding IT systems need to be reconciled to implement the change. All of the business challenges that we discussed in the 5.3, “Business challenges” on page 95, account for changes in the disparate IT systems. Similarly, ITSOPBank’s decision to introduce the concept of a premium customer cost the IT department to change their process to bind to different credit check and loan provider services. This set of changes accounted for about three calendar months from impact analysis, design, coding, testing, and re-deployment.

Business Process Management (BPM) enabled SOA applications that can extend and enhance the existing IT capabilities as needed. There is also a need to have greater flexibility and responsiveness through dynamic service selection instead of creating and deploying hard coded service bindings.

- Business logic is embedded in multiple locations and IT systems.

ITSOPBank faces a key challenge of having core business logic spread out across different IT systems, for example, the decision logic to select a particular credit check service is embedded in the portal application (J2EE™

application), hence any change request involves effort and cost for development and deployment.

ITSOBank needs the ability to consolidate the business logic (business level information) in a centralized federated repository for easy discovery and change management.

- ▶ Ability to monitor and manage the performance of business services.

ITSOBank has a third-party provide some of the business functionality, such as credit check and loan provider service. ITSOBank has to keep track of billing and metering of the service invocation because the transaction cost needs to be paid to the third-party service provider. Currently there is no provision in the loan application, which allows the bank to record the service invocation. In the current process, the service invocation is managed and monitored at the IT level, rather than at the business level.

- ▶ Service versioning

ITSOBank is the AS-IS process that we described in 5.2.2, “AS-IS business scenario” on page 89. The AS-IS process has two types of customer rating services—a heritage and a new rating service. A challenge that the bank is currently facing is how to migrate to the new rating service from the heritage-based service such that the heritage rating service is phased out. Proper versioning of the credit rating service needs to be in place within the bank, so as to enable business applications (consumers) to use the appropriate credit rating service that is based on customer classification.

- ▶ Service entitlements

With the number of IT systems that are being used and exposed as services, ITSOBank needs to manage the service entitlements to their users. They need an infrastructure that will allow them to manage the service entitlements at a business level.

- ▶ Normalized service interfaces, based on industry standards

ITSOBank faces challenges in terms of integrating a new rating engine because it is based on a proprietary interface and needs to have a normalized and consistent service interface that is based on industry standards.

ITSOBank needs a framework that allows them to build industry-specific SOA solutions that can be easily extended.

- Impact on the services utilization (service reuse)

In its path towards a mature SOA infrastructure, the bank wants to encourage different lines of businesses to use exposed enterprise services within the organization. Further, the bank wants to meter and measure the extent of service reuse, for example, the customer verification service can be re-used by different Lines of Business (LOB), and it gets difficult to maintain the correlation between the service reuse in various business processes.

- Impact of assimilating merger and acquisition systems and applications

All of the above IT challenges are further exacerbated by the need to rationalize with an entire new IT department and extract maximum capability from the new lending platform. They need an infrastructure to rapidly maximize and extract the appropriate value from both entities in a manner that is seamless to constituents.

## 5.5 Summary

In this chapter, we provided an overview of ITSOBank's vehicle loan processing system and detailed the business process that is associated with it. In this chapter, we also highlighted a few key business and IT challenges that the ITSOBank currently faces, and the pressing need to resolve these challenges in a cost effective and efficient way.

The subsequent chapters in the book demonstrate (using the vehicle loan processing use case as an illustration) how WebSphere Business Services Fabric along with other IBM Business Process Management (BPM) products help to build the new loan processing system, yet effectively addresses the current business and IT challenges.



# Installing and configuring Fabric

In this chapter, we provide an overview of how to install and configure WebSphere Business Services Fabric. It contains the following sections:

- ▶ 6.1, “Overview of products” on page 100
- ▶ 6.2, “Typical deployment topology for the Business Services Foundation Pack” on page 100
- ▶ 6.3, “Development topology for the Business Services Tool Pack” on page 102
- ▶ 6.4, “Installing the pre-requisites” on page 103
- ▶ 6.5, “Installing the Business Services Foundation Pack” on page 105
- ▶ 6.6, “Verifying the Business Services Foundation Pack installation” on page 111
- ▶ 6.7, “Installing the Business Services Tool Pack” on page 126
- ▶ 6.8, “Verifying the Business Services Tool Pack installation” on page 130
- ▶ 6.9, “Installing Fabric on z/OS” on page 132
- ▶ 6.10, “Known limitations” on page 133
- ▶ 6.11, “Clustering and failover” on page 134

## 6.1 Overview of products

WebSphere Business Services Fabric consists of the *Business Services Foundation Pack* and the *Business Services Tool Pack* to help simplify the business, technology, governance, and process interoperability challenges that are associated with business services in an SOA. Typically developers install the Tool Pack, and the Foundation Pack installs a real runtime environment.

WebSphere Business Services Fabric is an integrated product offering that includes WebSphere Process Server for the composition and deployment of flexible, service-oriented business processes.

WebSphere Service Registry and Repository is a prerequisite for WebSphere Business Services Fabric. WebSphere Service Registry and Repository stores, accesses, and manages technical information about services and service endpoint descriptions in an SOA.

WebSphere Business Services Fabric, as part of the Foundation Pack, provides a Business Service Repository that contains business-related meta data and industry semantic models for use in dynamic service selection and assembly. The Business Service Repository includes business-related policies, semantics, meta data, and subscriptions. Business Services Fabric leverages and extends the WebSphere Service Registry and Repository to source technical service meta data information and to provide business context through the use of Business Service Repository.

## 6.2 Typical deployment topology for the Business Services Foundation Pack

Figure 6-1 on page 101 provides a typical deployment topology for the Business Services Foundation pack.

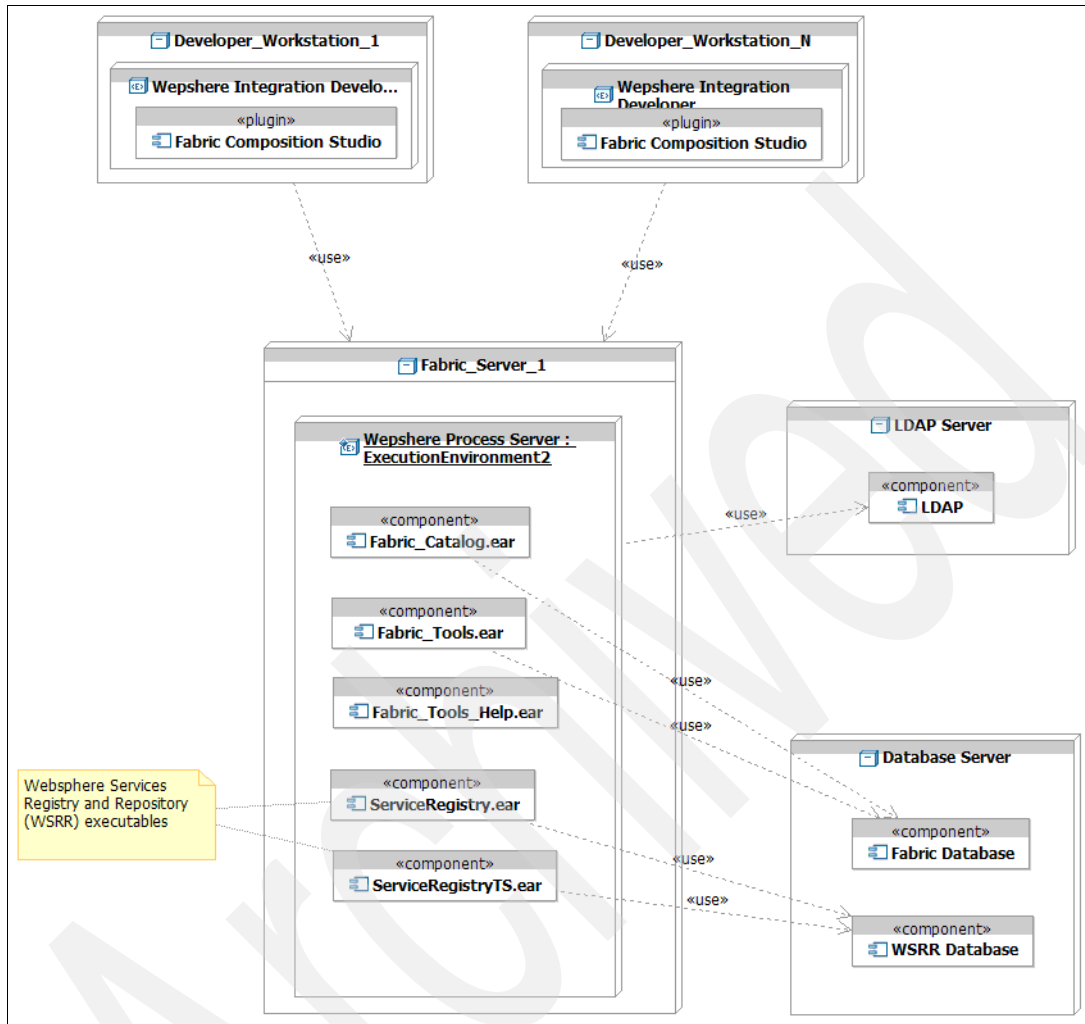


Figure 6-1 Typical deployment topology using the Foundation Pack

As shown in Figure 6-1, the WebSphere Business Services Fabric components, fabric-catalog.ear, fabric-engine.ear, fabric-tools.ear, and fabric-tools-help.ear, are deployed as enterprise applications on a WebSphere Process Server instance. You must enable global security on the instance, which is required for fabric-tools.ear because only authenticated users can access WebSphere Business Services Fabric Tools applications. The Business Services Fabric components use the Fabric Database.

The WebSphere Process Server instance can also contain WebSphere Service Registry and Repository components, ServiceRegistry.ear and

ServiceRegistryTS.ear. The WebSphere Service Registry and Repository components use the WSRR Database.

**Note:** You can also deploy WebSphere Service Registry and Repository on a separate WebSphere Application Server.

WebSphere Business Services Fabric integrates with an external repository, such as LDAP through WebSphere Process Server VMM (Virtual Member Manager). LDAP stores user profiles, which you can source in WebSphere Business Services Fabric Tools application to assign subscriptions as part of the Business Service Subscription process. For more information about this, refer to Chapter 17, “Integrating with Lightweight Directory Access Protocol” on page 481.

Figure 6-2 illustrates the Fabric Database.

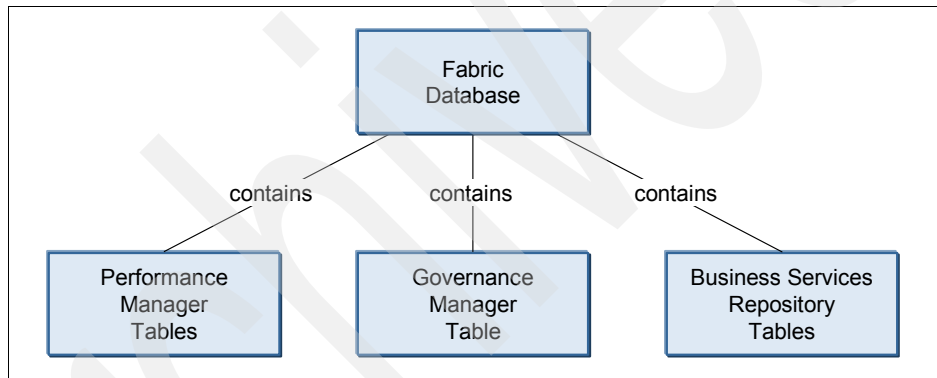


Figure 6-2 Fabric Database

The Fabric Database contains Performance Manager tables for storing dynamic assembler transaction records, Governance Manager tables for strong governance related transactions records, and Business Service Repository tables for storing Business Service Meta data.

## 6.3 Development topology for the Business Services Tool Pack

Figure 6-2 provides a typical development environment for the Business Services Tool Pack.

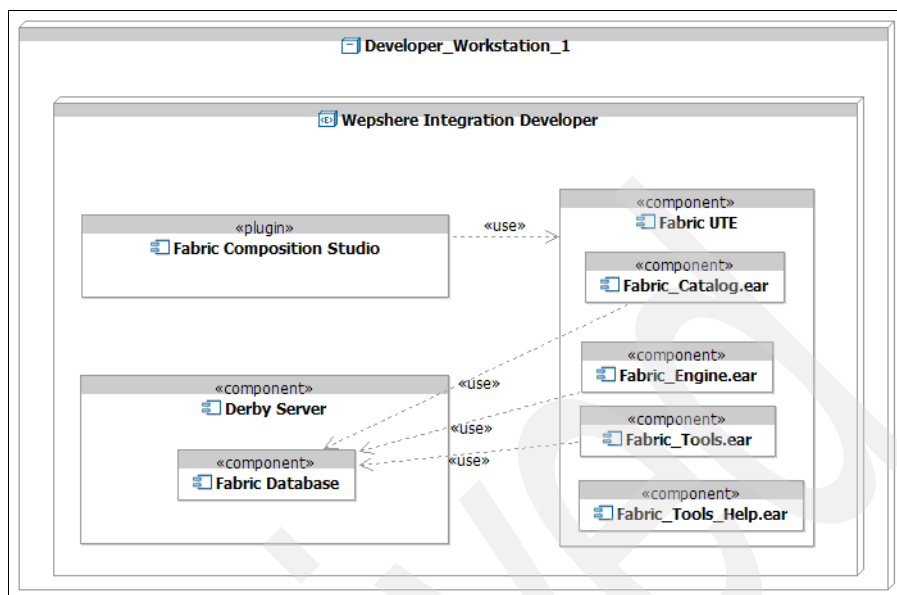


Figure 6-3 Development workstation topology using the Tool Pack

The developer workstation consists of WebSphere Integration Developer with the Business Services Tool Pack.

While installing the Tool Pack, you have an option of selecting a Unit Test Environment. The Unit Test Environment provides an embedded WebSphere Process Server instance that allows you to deploy and test composite business applications (CBA). The Unit Test Environment deploys the Business Services Foundation Pack on that WebSphere Process Server profile and uses a Derby database instead of DB2® or Oracle®.

Developers can model, assemble, deploy, and test the CBA applications and then migrate those CBA applications to a WebSphere Process Server environment that has the Foundation Pack installed.

## 6.4 Installing the pre-requisites

WebSphere Business Services Fabric includes WebSphere Process Server and WebSphere Integration Developer. The installation of WebSphere Process Server and WebSphere Integration Developer is a separate process. Before you run the Foundation Pack and Tool Pack installer, you should have WebSphere Process Server 6.1 and WebSphere Integration Developer 6.1 already installed.

**Note:** For the latest information about platform-specific disk space requirements, supported operating systems, and the operating system fixes and patches that you must install to have a compliant operating system, see WebSphere Business Services Fabric's detailed system requirements at the following Web address, and select the link to your version of WebSphere Business Services Fabric:

<http://www-306.ibm.com/software/integration/wbsf/sysreqs/index.html>

Apart from WebSphere Process Server and WebSphere Integration Developer, you must install the following software:

► WebSphere Service Registry and Repository V6.1

For instructions on how to install WebSphere Service Registry and Repository, refer to:

[http://publib.boulder.ibm.com/infocenter/sr/v6r1/topic/com.ibm.sr.doc/twsr\\_installn02.html](http://publib.boulder.ibm.com/infocenter/sr/v6r1/topic/com.ibm.sr.doc/twsr_installn02.html)

► Databases

The following databases are supported with the Foundation Pack:

- DB2 Enterprise Server Edition V8.2 Fix Pack 6 or Fix Pack 7
- DB2 Enterprise Server Edition V9.1 Fix Pack 1 or Fix Pack 2
- Oracle 10g Enterprise Edition Release 2 - V10.2.0.2

**Note:** For the ITSOPBank application, we used DB2 Enterprise Server Edition V9.1.

► LDAP servers

The following LDAP servers are supported with the Foundation Pack:

- IBM Tivoli Directory Server V6.0
- Microsoft Windows Active Directory® 2003
- IBM z/OS Security Server V1.7

**Note:** For the ITSOPBank application, we used Tivoli Directory Server 6.0.

- ▶ IBM Rational Software Architect V7.0.0.3 or later

Rational Software Architect is required to install the WebSphere Business Services Fabric Modeling Tool, which is provided as a set of plug-ins.

- ▶ Business Services Fabric Modeling Tool

The IBM Business Services Fabric Modeling Tool provides a modeling environment that allows business service architects and developers to use industry standards and models to create content for Fabric. The tool provides a repeatable approach to model Fabric concepts from an Industry Business Glossary. To install the Business Services Fabric Modeling Tool, refer to 8.7.1, “Installing plug-ins and importing profiles” on page 165.

**Note:** For the ITSObank applications in this book, we used a topology as described in 6.2, “Typical deployment topology for the Business Services Foundation Pack” on page 100 on a Windows XP environment with 3 GB of RAM.

## 6.5 Installing the Business Services Foundation Pack

The Business Services Foundation Pack provides the integrated run-time and management environment for CBA deployment.

**Note:** Before commencing the Foundation Pack installation, make sure you have WebSphere Process Server V6.1 installed and you have created a WebSphere Process Server profile which has security enabled. A WebSphere Process Server server should be running before starting the Foundation Pack installation.

To install the Business Services Foundation Pack:

1. From the installation directory where you downloaded WebSphere Business Services Fabric Foundation Pack, run **launchpad.exe**.
2. Click **IBM Business Service Foundation Pack installation**, and then click **Launch the installation wizard for IBM Business services Foundation Pack for Multiplatforms**, which opens the Language selection panel. Select your language, and click **Next**.
3. On the Welcome panel, click **Next**.
4. On the License Agreement panel, select **I accept the terms in license agreement**, and click **Next**.

- From the Install Set pull-down list, select **Custom**, and select all of the packages, as shown in Figure 6-4. Click **Next**.

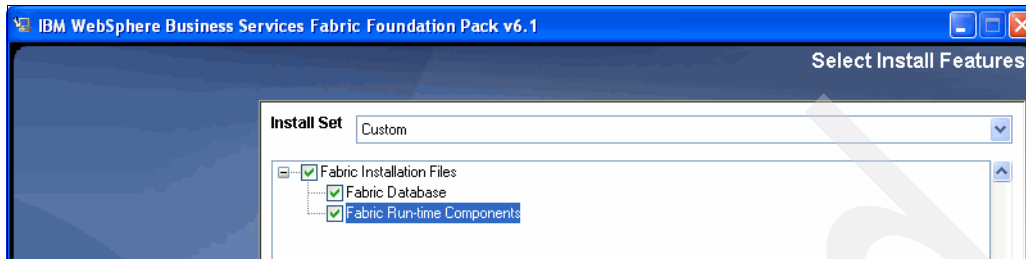


Figure 6-4 Select install packages

- Select the directory to install the installation files, as shown in Figure 6-5, and click **Next**.

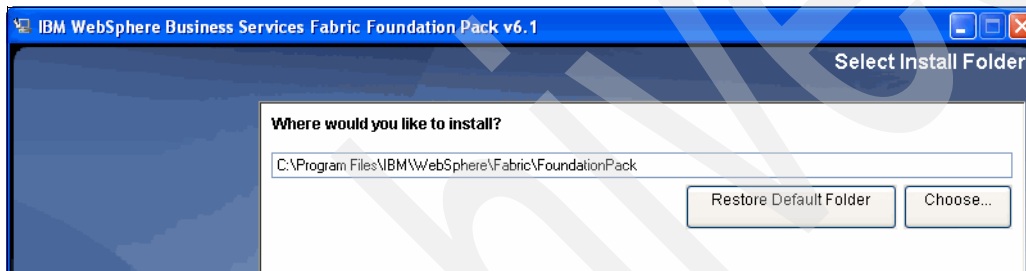


Figure 6-5 Select install folder

- Select the drive where the database will reside, for example C:, as shown in Figure 6-6 on page 107, and click **Next**.



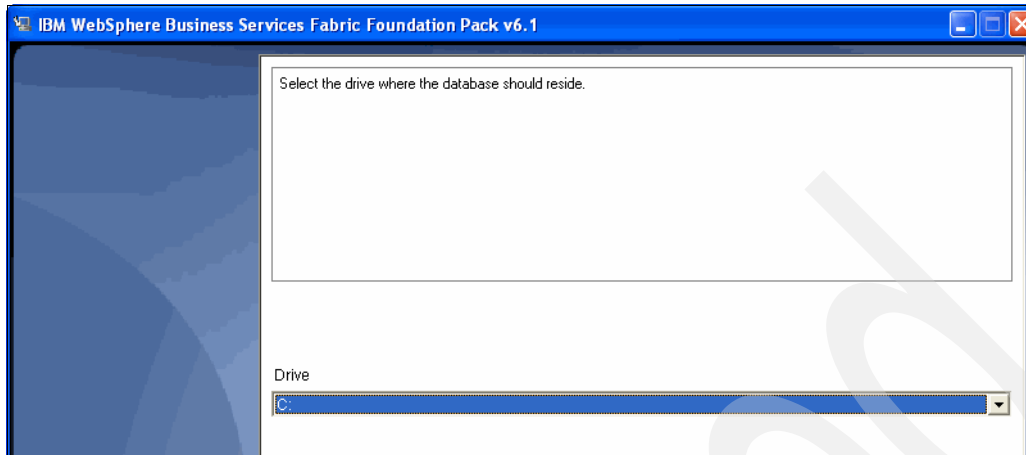


Figure 6-6 Specify database directory

8. Specify the user name and password for the database administrator, as shown in Figure 6-7. The database administrator is required to create the WebSphere Business Services Fabric database. In this example, we use db2admin as the username and password. Click **Next**.

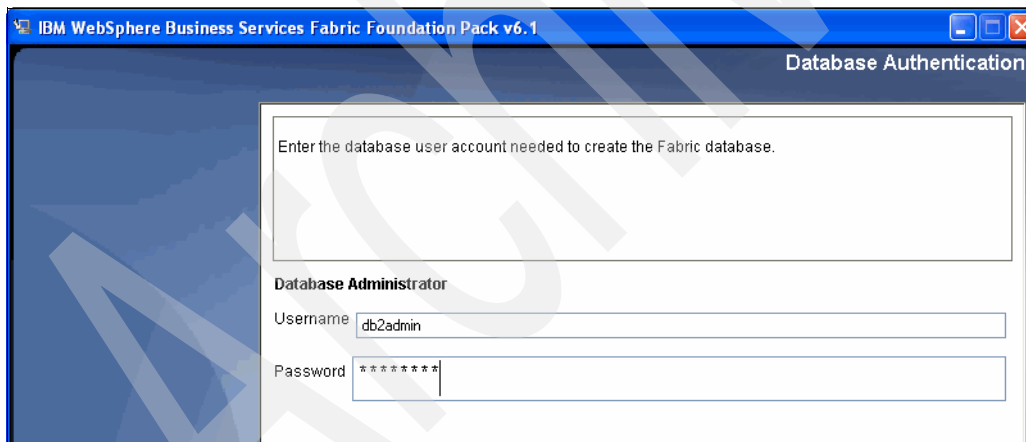


Figure 6-7 Specify Database Authentication

**Note:** Currently the installer does not check if the database username and password are valid and proceeds with the installation.

9. Select the WebSphere Process Server profile where you want to install the Foundation Pack, as shown in Figure 6-8. The WebSphere Process Server profile must have security enabled. Click **Next**.

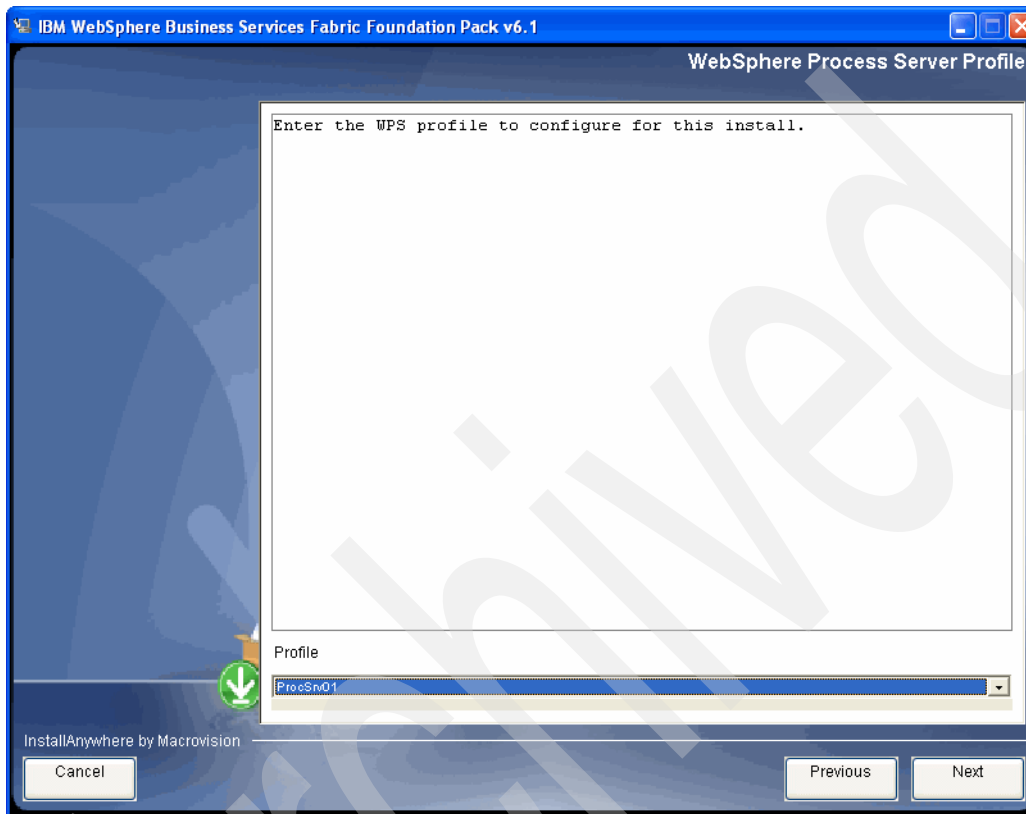


Figure 6-8 Specify WebSphere Process Server Profile

10. Enter the username and password that has access to the WebSphere Process Server, as shown in Figure 6-9 on page 109. Click **Next**.

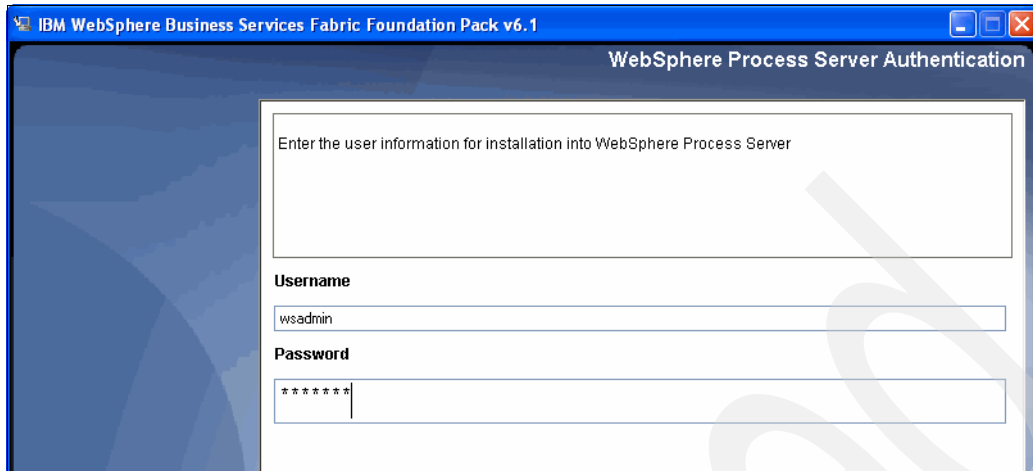


Figure 6-9 Specify WebSphere Process Server Authentication

**Tip:** You can use the same username and password that you use to log on to the WebSphere Process Server Administration Console. If you do not use any user credentials to log on to the WebSphere Process Server Administration Console, that implies that security is not enabled on the profile or that you might have turned it off. Security must be configured to install the Foundation Pack. For the ITSOBank scenario, when we created the profile, we set the user ID and password as wsadmin.

11. Enter the JDBC™ settings to connect to the database, as shown in Figure 6-10 on page 110. Provide the JDBC Driver Path, for example, for the DB2 database, the path is [DB2PathInstall]\SQLLIB\Java. Make sure that you provide the right path. The installer does not create the Fabric Database if the path that you supply is invalid. Click **Next**.

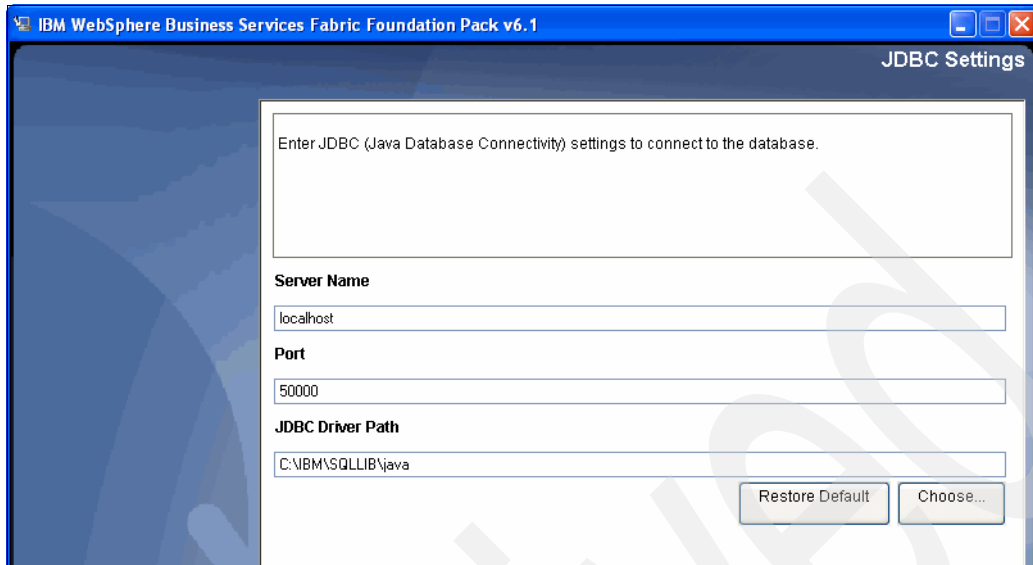


Figure 6-10 Specify JDBC settings

12. Configure the host name for the E-mail server. Configure the E-mail username and password. WebSphere Business Services Fabric uses this information to send users e-mail notifications (Figure 6-11). Click **Next**.

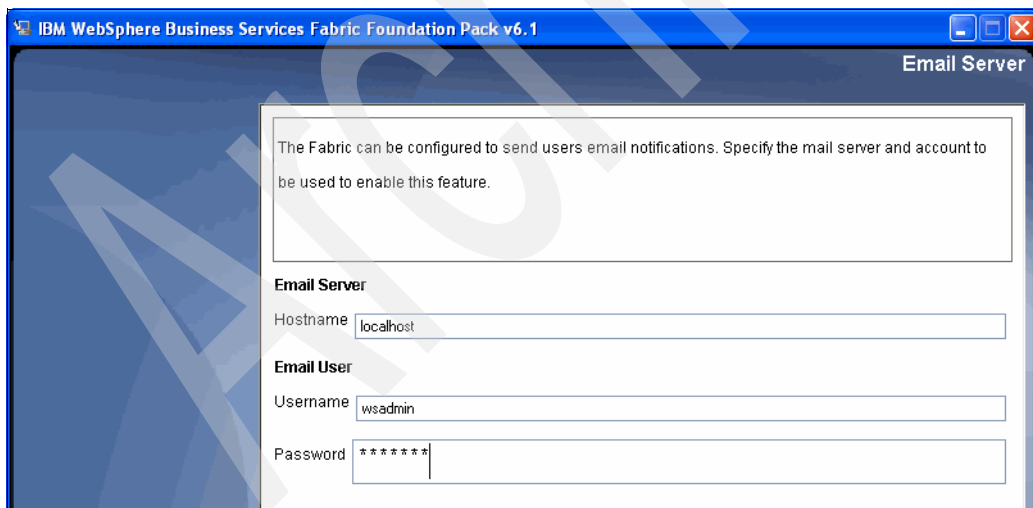


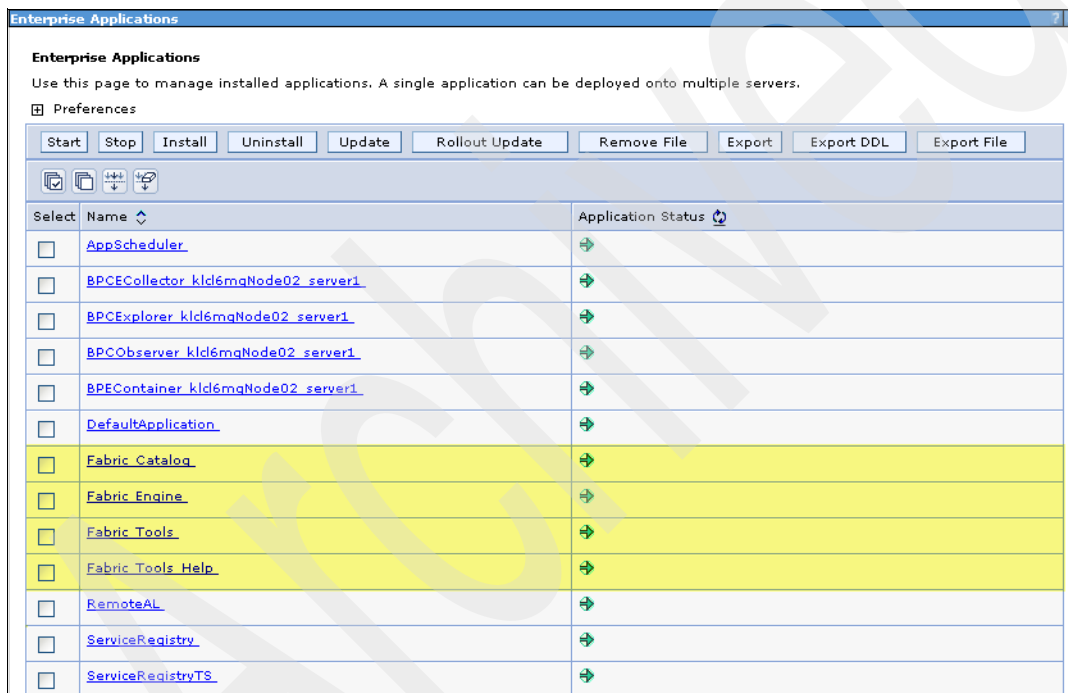
Figure 6-11 Configure E-mail Server

13. Click **Next** on the Pre Installation Summary panel.

14. You will see a successful confirmation message after WebSphere Business Services Fabric is successfully installed.

## 6.6 Verifying the Business Services Foundation Pack installation

The Foundation Pack installs the EAR files that we show in Figure 6-12 (highlighted in yellow), which you can view in the Enterprise Applications panel of the WebSphere Process Server Administration Console. Logon to the Administration Console, on the menu on the left, click **Enterprise Applications**.



Select	Name	Application Status
<input type="checkbox"/>	<a href="#">AppScheduler</a>	Running
<input type="checkbox"/>	<a href="#">BPCECollector.kld6mqNode02_server1</a>	Running
<input type="checkbox"/>	<a href="#">BPCEExplorer.kld6mqNode02_server1</a>	Running
<input type="checkbox"/>	<a href="#">BPCEObserver.kld6mqNode02_server1</a>	Running
<input type="checkbox"/>	<a href="#">BPCEContainer.kld6mqNode02_server1</a>	Running
<input type="checkbox"/>	<a href="#">DefaultApplication</a>	Running
<input type="checkbox"/>	<a href="#">Fabric Catalog</a>	Running
<input type="checkbox"/>	<a href="#">Fabric Engine</a>	Running
<input type="checkbox"/>	<a href="#">Fabric Tools</a>	Running
<input type="checkbox"/>	<a href="#">Fabric Tools Help</a>	Running
<input type="checkbox"/>	<a href="#">RemoteAL</a>	Running
<input type="checkbox"/>	<a href="#">ServiceRegistry</a>	Running
<input type="checkbox"/>	<a href="#">ServiceRegistryTS</a>	Running

Figure 6-12 Enterprise Applications in the Administrative Console

If all four enterprise applications are running, this implies that the Foundation Pack is successfully installed. If all four enterprise applications are not running, look at the System.out.log files for errors related to the startup of enterprise applications. Refer to 6.6.2, “Troubleshooting your installation” on page 121 to resolve any installation issues.

## 6.6.1 Accessing the Fabric Tools Console

The Fabric\_Tools enterprise application can only be used by an Administrator role. To use Fabric\_Tools, you first need to map users or a group to the Administrator role. The users and groups can come from an external registry, such as LDAP. If you already have an LDAP configuration, you can source users and groups from it and map it to an Administrator role, as described in 17.2.1, “Foundation Pack administrator configuration” on page 485.

**Note:** LDAP configuration support is provided through the VMM feature in WebSphere Process Server, which WebSphere Business Services Fabric uses.

For verification purposes, you can also use a File Repository. While you create a WebSphere Process Server profile, if you enable security, WebSphere Process Server by default configures a federated repository as the user account repository and configures a File Repository to store the user information.

To verify that the File Repository is configured:

1. Navigate to the left menu of the Administration Console, and select **Security → Secure administrations, applications and infrastructure**. You should see that the user account repository is set to federated repositories, as shown in Figure 6-13 on page 113.

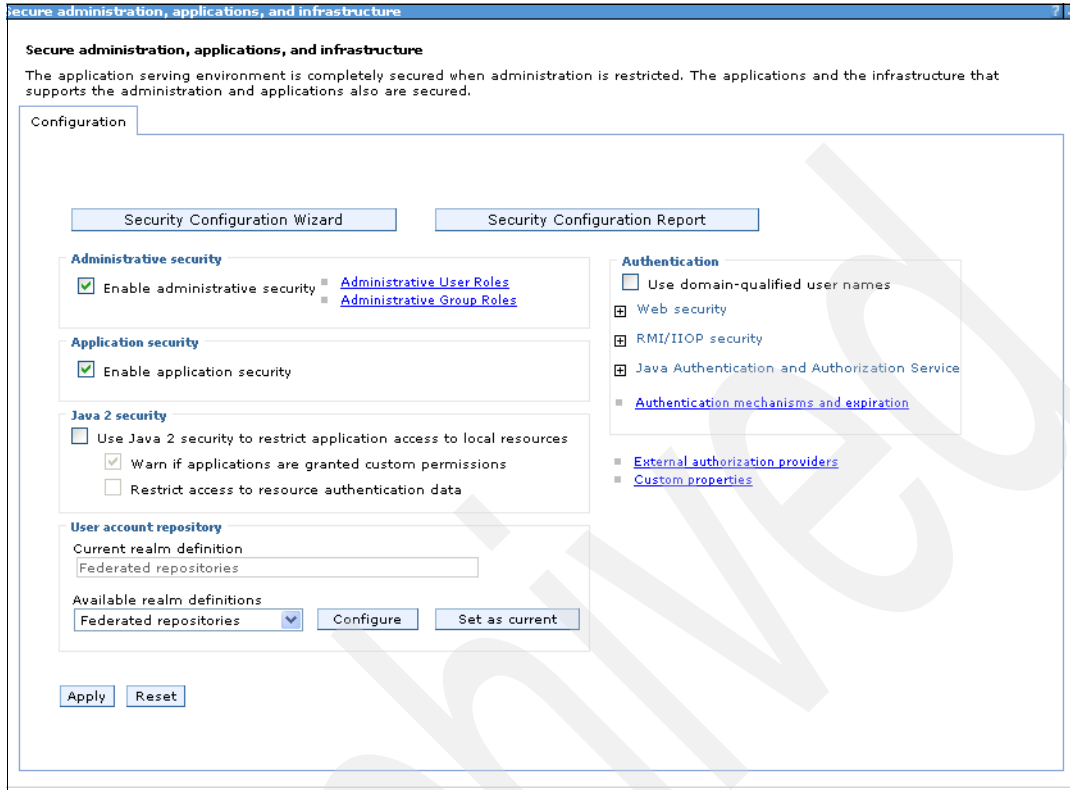


Figure 6-13 Federated repository as user account repository

2. Click **Configure**, and you will see a configured file repository, as shown in Figure 6-14 on page 114. You can add multiple repositories here.

Secure administration, applications, and infrastructure

[Secure administration, applications, and infrastructure](#) > **Federated repositories**

By federating repositories, identities stored in multiple repositories can be managed in a single, virtual realm. The realm can consist of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in repository and one or more external repositories.

Configuration

**General Properties**

\* Realm name  
defaultWIMFileBasedRealm

\* Primary administrative user name  
wsadmin

**Server user identity**

☒ Automatically generated server identity

☐ Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

☒ Ignore case for authorization

Repositories in the realm:

Select	Base entry	Repository identifier	Repository type
<input type="checkbox"/>	o=defaultWIMFileBasedRealm	InternalFileRepository	File

Figure 6-14 File Repository configuration

You can create and add users and roles to this file -based repository. We will create a group called FabricAdminGroup, and add a user to it. You can use the same username that you use to login to the WebSphere Administration console, or create a new user, and add the new user to the FabricAdminGroup.

## Creating groups and users

1. To create a group, navigate to **Users and Groups** → **Manage Groups**, and click **Create**. In the next panel, create a group name called FabricAdminGroup, and click **Create**, as shown in Figure 6-15 on page 115. On the next panel, click **Close**.



**View:** All tasks

- Welcome
- Guided Activities
- Servers
- Applications
- Resources
- Security
- Environment
- Integration Applications
- System administration
- Users and Groups
  - Administrative User Roles
  - Administrative Group Roles
  - Manage Users
  - Manage Groups
- Monitoring and Tuning
- Troubleshooting

**Manage Groups**

**Create a Group**

\*Group name  
FabricAdminGroup

Description  
Fabric administrations group

Create Cancel

Figure 6-15 Create a group

- To create a user, navigate to **Users and Groups** → **Manage Users**, and click the **Create** button. On the Create user panel, create a user ID name, fabricadmin, and provide the required details, as shown in Figure 6-16.

**View:** All tasks

- Welcome
- Guided Activities
- Servers
- Applications
- Resources
- Security
- Environment
- Integration Applications
- System administration
- Users and Groups
  - Administrative User Roles
  - Administrative Group Roles
  - Manage Users
  - Manage Groups
- Monitoring and Tuning
- Troubleshooting

**Manage Users**

**Create a User**

\*User ID  
fabricadmin **Group Membership**

\*First name  
Fabric

\*Last name  
Admin

E-mail

\*Password  
\*\*\*\*\*

\*Confirm password  
\*\*\*\*\*

Create Cancel

Figure 6-16 Create a user

3. Click the **Group Membership** button, and in the Group Membership panel, enter FabricAdminGroup in the Search for text field, as shown in Figure 6-17. Click **Search**, and add the **FabricAdminGroup** to Current Groups, and click the **Close** button.

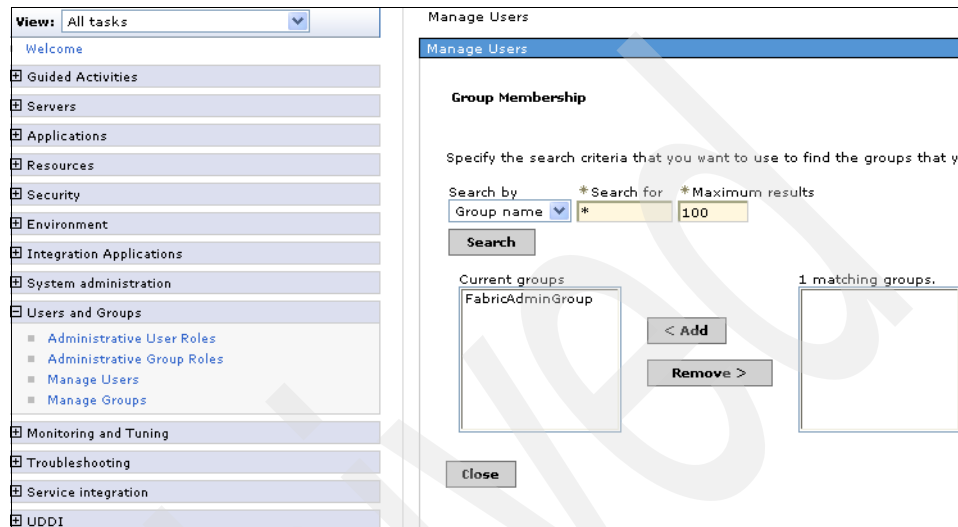


Figure 6-17 Add Group Membership

4. On the Create a user panel, click **Create** to create the user and to link it to the FabricAdminGroup group.
5. When the group is created, you can map the group to the Tools J2EE Role Administrator role so that the user from the FabricAdminGroup group can access the Tools console. Click **Applications** → **Enterprise Applications**, and click **Fabric\_Tools** in the WebSphere Process Server Administration Console. On the next panel, click **Security role to user/group mapping**, as shown in Figure 6-18 on page 117.

Enterprise Applications

**Enterprise Applications**

[Enterprise Applications](#) > [Fabric\\_Tools](#)

Use this page to configure an enterprise application. Click the application or its modules.

Configuration

**General Properties**

\* Name  
Fabric\_Tools

Application reference validation  
Issue warnings

**Detail Properties**

- [Target specific application status](#)
- [Startup behavior](#)
- [Application binaries](#)
- [Class loading and update detection](#)
- [Remote request dispatcher properties](#)
- [Security role to user/group mapping](#)
- [View Deployment Descriptor](#)
- [Last participant support extension](#)

**References**

Figure 6-18 Fabric\_Tools Enterprise Applications

- On the Security role to user/group mapping panel (Figure 6-19), select **Administrator**, and click **Look up groups**.

Enterprise Applications

[Enterprise Applications](#) > [Fabric\\_Tools](#) > [Security role to user/group mapping](#)

Security role to user/group mapping

Each role that is defined in the application or module must map to a user or group from the domain user registry.

Select	Role	Everyone?	All authenticated?	Mapped users	Mapped groups
<input type="checkbox"/>	Administrator	<input type="checkbox"/>	<input type="checkbox"/>		FabricAdminGroup Administrator
<input type="checkbox"/>	QA Engineer	<input type="checkbox"/>	<input type="checkbox"/>		QA Engineers
<input type="checkbox"/>	Developer	<input type="checkbox"/>	<input type="checkbox"/>		Developers
<input type="checkbox"/>	Architect	<input type="checkbox"/>	<input type="checkbox"/>		Architects
<input type="checkbox"/>	Business Analyst	<input type="checkbox"/>	<input type="checkbox"/>		Business Analysts
<input type="checkbox"/>	Business Architect	<input type="checkbox"/>	<input type="checkbox"/>		Business Architects

Figure 6-19 Lookup groups

7. On the Look up users or groups panel, enter FabricAdminGroup as the search string, and click **Search**. Click >> to add it to the selected group, as shown in Figure 6-20. Click **OK**.

The screenshot shows the 'Enterprise Applications' window with the breadcrumb path: Enterprise Applications > Fabric\_Tools > Security role to user/group mapping > Look up users or groups. The page title is 'Look up users or groups'. Below the title, it says 'Specifies whether to look up users or groups.' and 'The following roles are mapped to the items in the selected list.' There is a list box containing 'Administrator'. Below this, there is a search section with a 'limit (number of items)' set to '20' and a 'Search String' field containing 'FabricAdminGroup'. A 'Search' button is next to the search string. Below the search section, it says 'Select users or groups in the Available list. Move them to the Selected list by clicking >>.' There are two list boxes: 'Available:' containing 'FabricAdminGroup' and 'Selected:' containing 'Administrator' and 'FabricAdminGroup'. Between the list boxes are '>>' and '<<' buttons. At the bottom are 'OK' and 'Cancel' buttons.

Figure 6-20 Map role to groups

8. On the Security role to user/group mapping panel, you will see that the group is mapped to the Administrator role, as shown in Figure 6-21 on page 119. Click **OK** to save the changes. When you click **OK**, you will see a message to save it to master configuration. Click **Save** to make the changes effective, which restarts the **Fabric\_Tools** enterprise application with the modified configuration.

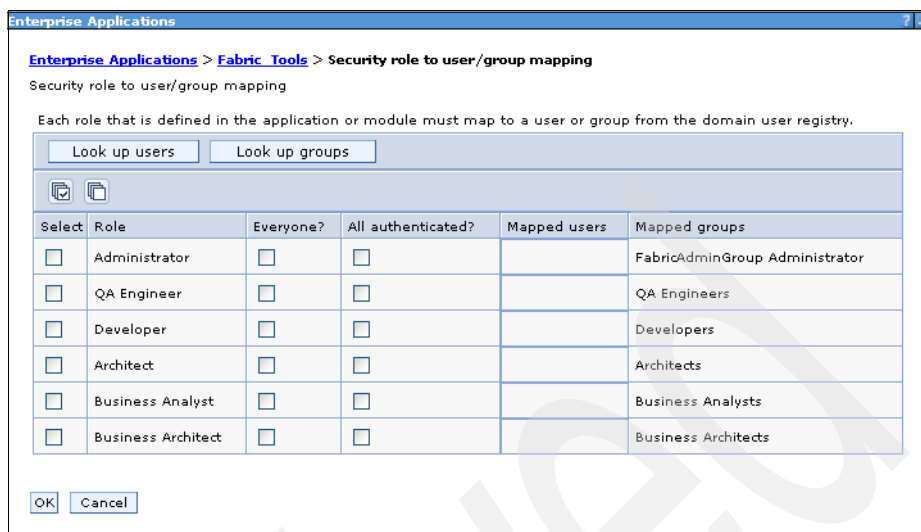


Figure 6-21 Role to Group mapping

You can now access the Fabric Console.

## Accessing the Fabric Console

To access the Fabric Console:

1. In a browser, type the Web address of the WebSphere Business Services Fabric Console, for example, `http://localhost:9080/fabric/app`. Replace the host name and port name based on your WebSphere Process Server configuration. Enter the username as **fabricadmin** and the password.

**Note:** We configured the File Based Repository using VMM for verification. Later, if you plan to use LDAP, and if the user resides on the repository, you need to remove the File Based repository because the user id cannot be common across repositories. If you used a File Based Repository user id to logon to the Fabric Administration Console, before you switch to the LDAP repository, you must grant one user from LDAP Administrator privileges in Fabric, and subscribe that user to all the services.

2. After you login, you will see the panel, as shown in Figure 6-22 on page 120. Click all of the service subscriptions to subscribe to all services, and then click **Save**.

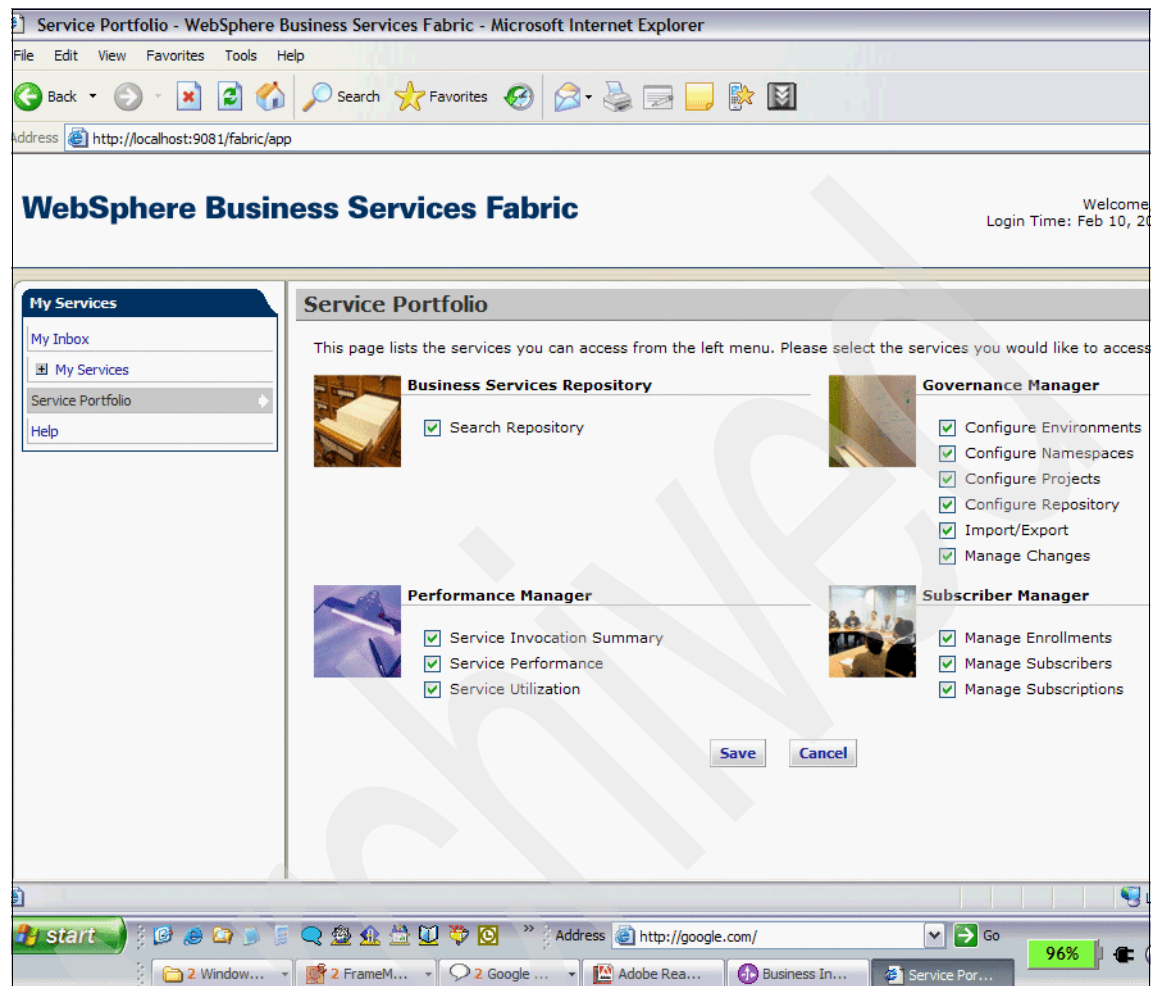


Figure 6-22 Service Portfolio window

**Note:** The WebSphere Business Services Fabric system grants administrative access to the first VMM-authenticated user who logs in. This user can then grant privileges to others. So if you plan to switch to another repository, such as LDAP, after you configure the File Based Repository and are using the File Based Repository user id by logging into WebSphere Business Services Fabric Console, you first need to logon with the File Based Repository user id, source the LDAP users in Subscriber Manager, assign them to System Organization, and subscribe the user to all of the Fabric services. You also need to map the LDAP group to the Administrator J2EE Role that Fabric\_Tools requires, as discussed earlier. After doing that, you can remove the File Based Repository configuration from WebSphere Process Server VMM configuration. Note that the same user id should not exist in multiple repositories.

## 6.6.2 Troubleshooting your installation

To help troubleshoot your installation, it is important to understand the resources that the installation process creates. The installation process should have created the following:

1. Created a FabricDB database.

To verify if the database is properly installed, open the DB2 Control Center, and look for the FabricDB database. Verify that there are 23 tables that belong to the database user id that you used during installation.

If the database is not created, an error was most likely encountered during database creation. To work around this, manually create the database and use the scripts that are provided in the database folder for your database. Refer to the *FabricInstallGuide\_language.pdf* (for English *FabricInstallGuide\_en.pdf*), *Chapter 8 Getting Started with Manual Installation - Creating and Configuring Databases* section for instructions. This manual is located in the documentation\InstallGuide folder of the Foundation Pack installable.

The database is generally not created due to an incorrect database user name and password that you supplied during the installation. The installer does not verify the database credentials that you entered at installation time.

**Note:** As part of the installation process, the WebSphere Business Services Fabric application bootstrap process loads the IBM Core Services Pack into the FabricDB database. This occurs when you first start up the application server with the EAR files deployed. The system automatically creates the necessary database tables and loads the necessary reference data that consists of IBM Core Services Pack and the WebSphere Business Services Fabric life cycle services.

2. Installed Fabric enterprise application EAR files.

To verify if the EAR files are successfully installed, logon to the WebSphere Administration Console, and go to **Enterprise Applications**. You should see the following four enterprise applications, as shown in Figure 6-23 on page 123:

- Fabric\_Tools: Installs the Fabric Administration console
- Fabric\_Tools\_Help: Installs the Fabric Web Help files
- Farbric\_Catalog: Installs the catalog services like Remote FCA Imports
- Fabric\_Engine: Installs the Dynamic Assembler runtime application

If the application status is green for all of these enterprise applications, this implies that the enterprise applications were started successfully.



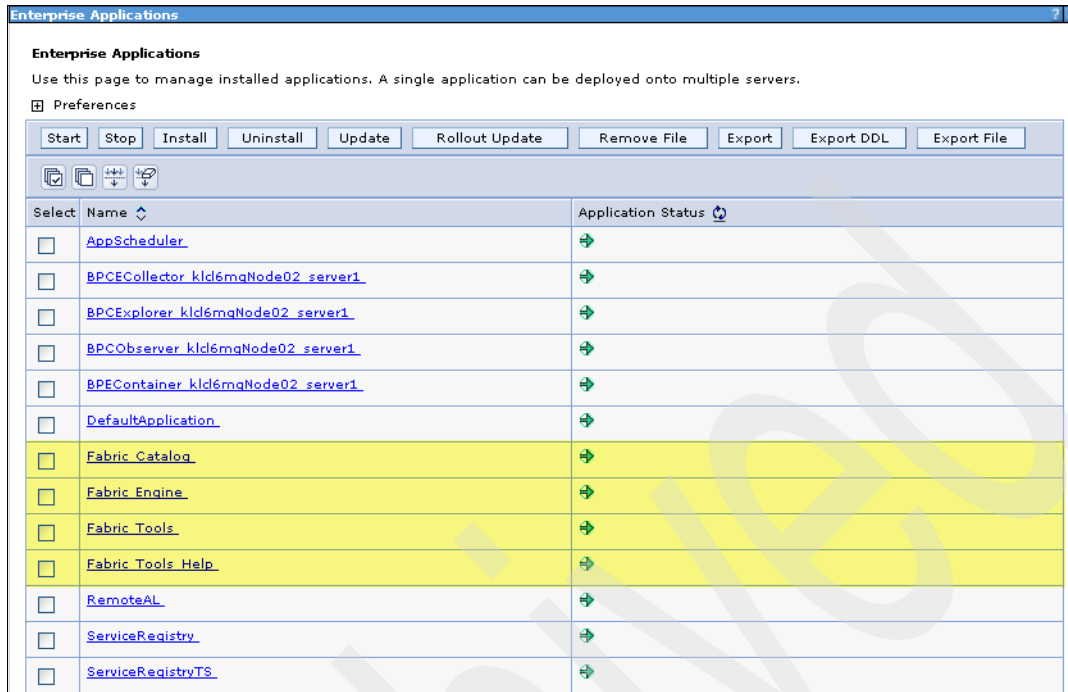


Figure 6-23 Viewing WebSphere Business Services Fabric Enterprise applications window

### 3. Created the data sources.

To verify if data sources were successfully created, perform the following:

- Logon to the WebSphere Administration Console, and click **Resources** → **JDBC** → **Data sources**.

You should see three data sources (Figure 6-24 on page 124):

- fabric\_bsr (managed Business Service Repository application connections)
  - fabric\_gm (manages governance application connections)
  - fabric\_pm (manages performance application connections)
- Select all three data sources, and click **Test connection**. If the connection works, this implies that the data sources are working fine. If the connection does not work, click the data source settings, provide the right database configuration, and save the settings. Make sure that the database username and password are valid.

<input type="checkbox"/>	<a href="#">fabric_bsr</a>	jdbc/fabric/bsr
<input type="checkbox"/>	<a href="#">fabric_qm</a>	jdbc/fabric/qm
<input type="checkbox"/>	<a href="#">fabric_pm</a>	jdbc/fabric/pm

Figure 6-24 Data source connections

4. Created a topic, a connection factory, and an activation specification for handling events that the WebSphere Business Services Fabric Dynamic Assembler emits.
  - a. To verify if the topic is successfully created, logon to the WebSphere Administration Console, click **Resources** → **JMS** → **Topics**, and you should see the DA Event Topic, as shown in Figure 6-25.

New Delete				
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>				
Select	Name	JNDI name	Provider	Description
<input type="checkbox"/>	<a href="#">CommonEventInfrastructure AllEventsTopic</a>	jms/cei/notification/AllEventsTopic	Default messaging provider	Common Event Infrastructure All Events Topic
<input type="checkbox"/>	<a href="#">DA Event Topic</a>	jms/fabric/DAEventTopic	Default messaging provider	
Total 2				

Figure 6-25 DA Event Topic configuration

- b. To verify that the connection factory is successfully created, logon to the WebSphere Administration Console,

click **Resources** → **JMS** → **Connection Factories**, and you should see the DA Event Connection Factory, as shown in Figure 6-26.

<input type="checkbox"/>	<a href="#">DA Event Connection Factory</a>	jms/fabric/DAEventConnectionFactory	Default messaging provider
--------------------------	---	-------------------------------------	----------------------------

Figure 6-26 DA connection factory configuration

- c. To verify that the activation specification is successfully created, logon to WebSphere Administration Console, click **Resources** → **JMS** → **Activation Specifications**, and you should see the DA PerfMon Activation, as shown in Figure 6-27.

<input type="checkbox"/>	<a href="#">DA PerfMon Activation</a>	jms/fabric/DAPerfMonActivation	Default messaging provider
<input type="checkbox"/>	<a href="#">HTMInternalActivationSpec</a>	eis/HTMInternalActivationSpec	Default messaging provider
<input type="checkbox"/>	<a href="#">HUB Event Activation</a>	jms/fabric/HubEventActivation	Default messaging provider
<input type="checkbox"/>	<a href="#">HUB Request Activation</a>	jms/fabric/HubRequestActivation	Default messaging provider

Figure 6-27 DA activation specifications

5. Created a mail session that sends notifications through e-mails.

To verify that the mail session was successfully created, logon to the WebSphere Administration Console, click **Resources** → **Mail** → **Mail Sessions**, and you should see the Fabric Mail, as shown in Figure 6-28. You can configure the required mail providers by clicking **Fabric Mail session**.

Select	Name ▾	JNDI name ▾
<input type="checkbox"/>	<a href="#">Fabric Mail</a>	mail/fabric

Figure 6-28 Mail provider

6. Created an SCA service integration bus.

To verify if the SCA service integration bus was successfully created, logon to the WebSphere Administration Console, click **Service Integration** → **Buses**, and you should see the fabricbus, as shown in Figure 6-29 on page 126. Click **fabricbus**, and select **Messaging engines**. You should see that the fabricbus messaging engine application status is green.

<input type="checkbox"/>	<a href="#">fabricbus</a>	Used by the WBSF JMS infrastructure	<a href="#">Disabled</a>
--------------------------	---------------------------	-------------------------------------	--------------------------

Figure 6-29 Fabric SCA integration bus

The installer copies the WebSphere Business Service Fabric Dynamic Assembler and SCA libraries (fabric-da-api.jar,fabric-da-sca.jar,fabric-da-scdl.jar) to the lib\ext directory of your WebSphere Process Server installation.

7. Verified that the libraries exist. This is required at run time when you are using the Dynamic assembler SCA component.

**Warning:** The installer starts and stops the server a couple of times. If you experience problems, make sure that there is no firewall blocking any ports, which interferes with the installation.

## 6.7 Installing the Business Services Tool Pack

The IBM Business Services Composition Studio and WebSphere Integration Developer comprise the Business Services Tool Pack, which provides the integrated design and assembly environment for CBA development.

**Note:** Before you install the Tool Pack, make sure that you have WebSphere Integration Developer V6.1 installed and that it is not running.

If you are planning to deploy and test your CBA applications in WebSphere Integration Developer, install the Fabric Unit Test Environment as part of the Tool Pack installation procedure. To install this test environment, ensure that you already have a WebSphere Process Server Unit Test Environment created.

To install the Business Services Tool Pack:

1. From the installation directory of the WebSphere Business Services Fabric Tool Pack, click **launchpad.exe** to start the installation.

The Launchpad for IBM Business Service Fabric Tool Pack opens up, as shown in Figure 6-30 on page 127.

2. Click the **IBM Business Services Tool Pack installation**.

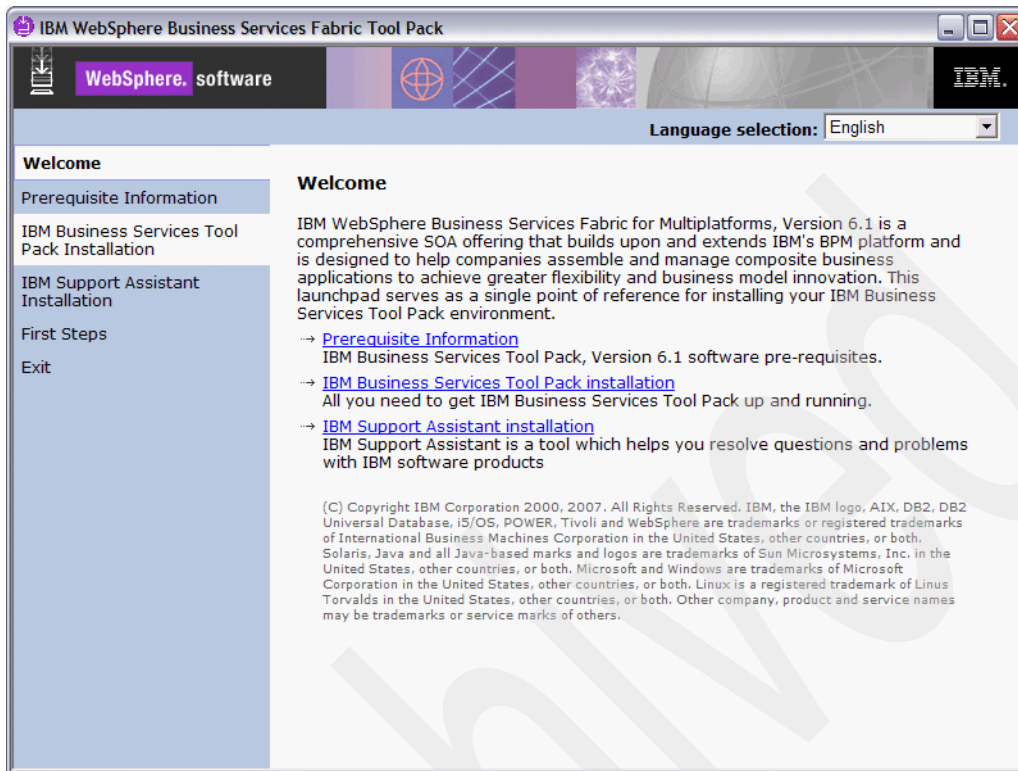


Figure 6-30 Welcome window

3. Click **Launch the installation wizard for IBM Business Services Tool Pack for Multi platforms**, as shown in Figure 6-31 on page 128.

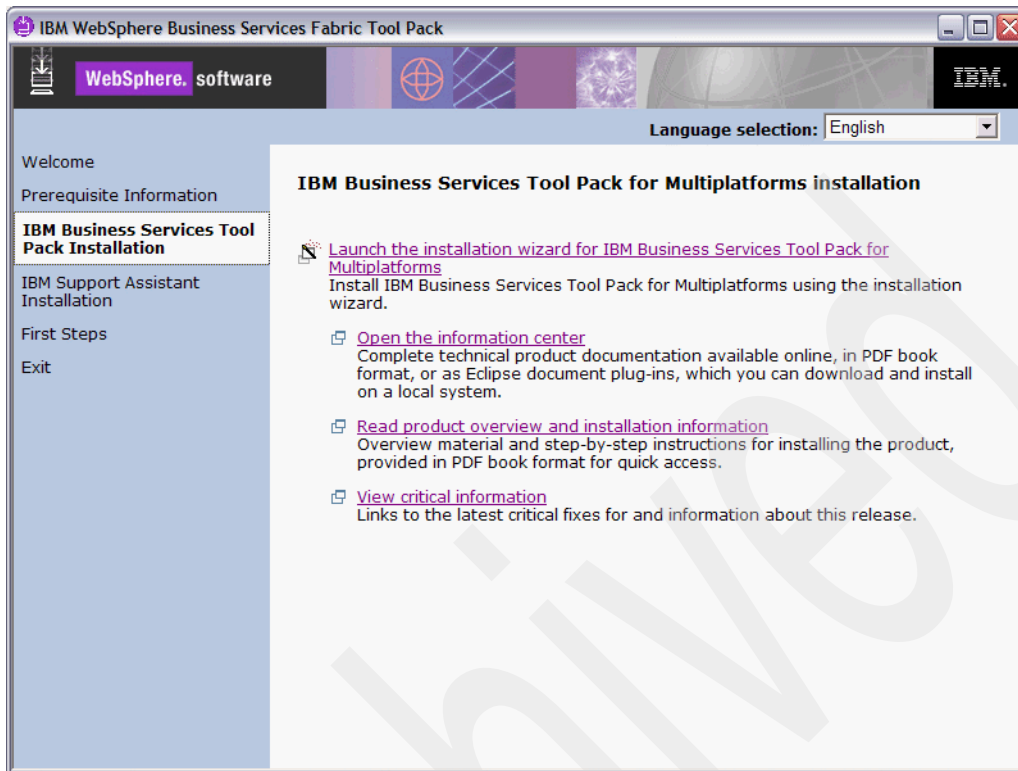


Figure 6-31 Launchpad for Tool pack installation

4. Select your language of choice, and click **OK**.
5. When the Install Anywhere Welcome panel opens up, click **Next**.
6. Review the Software License Agreement, and select **I accept the terms in license agreement**, and click **Next**.
7. From the Install Set pull-down menu, select **Custom**, and select **Install Composition Studio** and **Unit Test Environment**, as shown in Figure 6-32 on page 129.

The Unit Test Environment lets you deploy and test CBA applications using an embedded WebSphere Business Services Fabric server. The embedded WebSphere Business Services Fabric server is an extension to the WebSphere Process Server Unit Test Environment.

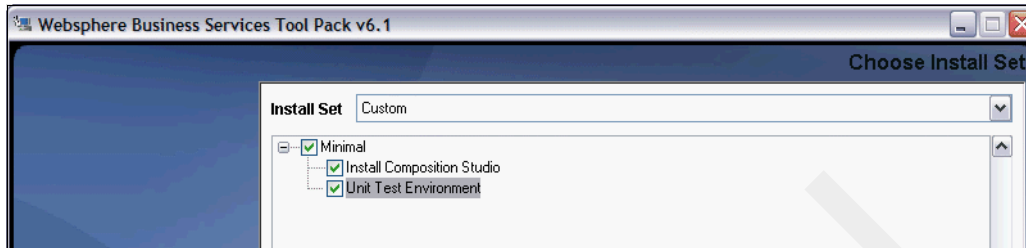


Figure 6-32 Custom installation

8. Enter the path where you want the Install Shield to copy the application files, as shown in Figure 6-33. Click **Next**.

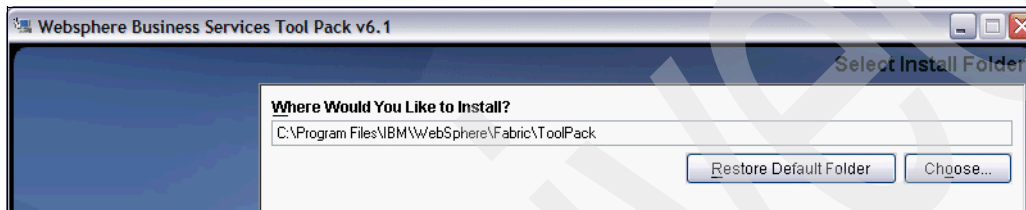


Figure 6-33 Specify install directory

9. Enter the WebSphere Integration Developer installation directory, as shown in Figure 6-34. The installer copies the Composition Studio into the plug-in folder of the WebSphere Integration Developer installation directory.

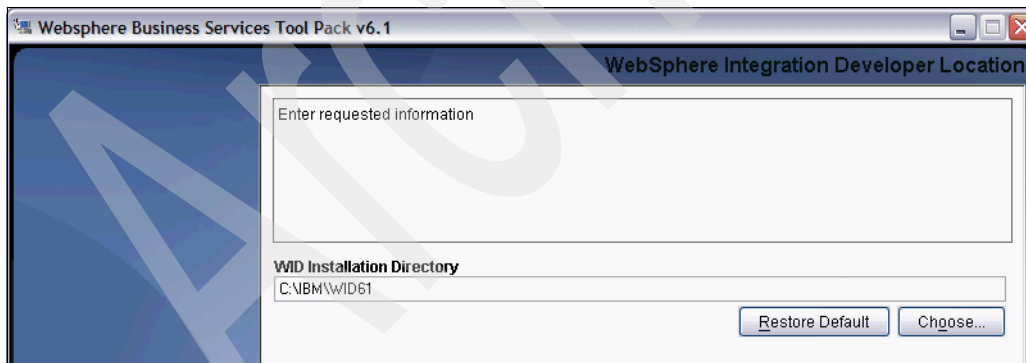


Figure 6-34 WebSphere Integration Developer installation directory

10. On the Preinstallation Summary panel, click **Install**. A successful confirmation message is displayed if the Tools Pack is successfully installed.

## 6.8 Verifying the Business Services Tool Pack installation

To verify the Tool Pack installation, perform the following:

1. Open WebSphere Integration Developer, and click **Window** → **Open Perspective** → **Other**, and select **Business Service**, as shown in Figure 6-35. The Tool Pack installs the Business Service perspective, which is used to assemble CBA applications.

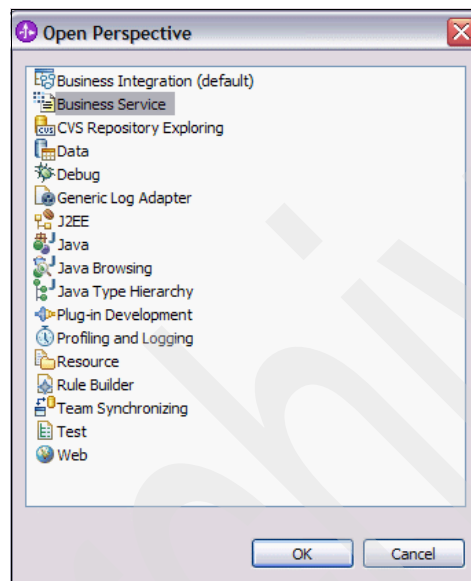


Figure 6-35 Perspective that displays Business Service

2. If you selected to install the Unit Test Environment during installation, you will see WebSphere Business Services Fabric Server V6.1 in the Servers view of the Business Services perspective.

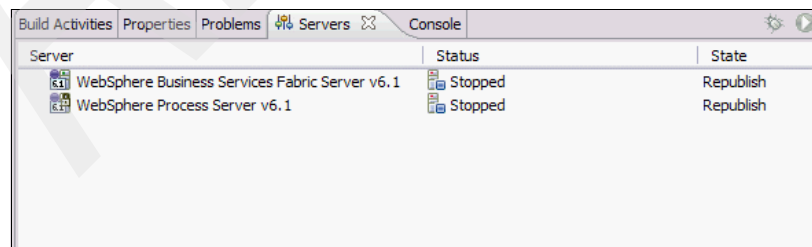


Figure 6-36 Servers view displaying WebSphere Business Services Fabric Server 6.1



In the Unit Test Environment, you can deploy and test CBA applications. The Fabric Unit Test Environment is similar to the WebSphere Process Server Unit Test Environment, but with some additional runtime libraries added that WebSphere Business Services Fabric requires.

When you install the Fabric Unit Test Environment, you also get the Foundation Pack that is installed on that server profile. The only difference between the Fabric Unit Test Environment and the Foundation Pack is that the Unit Test Environment uses a Derby database and has auto approve enabled, which skips the governance process.

**Note:** The installer copies a new version of the Derby libraries in [WIDInstallableDirectory]\runtimes\bi\_v61\derby and backs up the existing Derby libraries in [WIDInstallableDirectory]\runtimes\bi\_v61\derby.orig. Make sure that the Derby libraries are copied.

3. To verify if the Fabric Unit Test Environment is properly configured, start the Fabric server by right-clicking **WebSphere Business Services Fabric server v6.1**, and select **Start**.
4. After the server starts, logon to the WebSphere Business Services Fabric console using `http://localhost:port/fabric/app` (replace port with your WebSphere port configuration) using the user ID and password of admin.

**Note:** As part of the Unit Test Environment installation, the installer automatically creates an Administrator role and assigns the admin user id to it.

The Service Portfolio panel is displayed (also shown in the Foundation Pack verification section Figure 6-22 on page 120).

5. Select all services, and click **Save** to subscribe to all of the services. Click **Save** to assign all services to the admin user and to grant the admin user administrator access.

**Note:** The WebSphere Business Fabric database (fabricdb) is located in the [WIDInstallableDirectory]\runtimes\bi\_v61\derby folder.

**Note:** For the ITSOPBank scenario, we use the Tool Pack to develop the CBA application, and we use the Unit Test Environment to deploy and test CBA applications. We use the internal File Based Repository as the user repository. For testing the ITSOPBank application, we create a user called fabricadmin and a group called FabricAdminGroup, link the user to the group, and assign the FabricAdminGroup to Fabric\_Tools Role, as described in 6.6.1, “Accessing the Fabric Tools Console” on page 112.

After we test the CBA application, we export the WebSphere Business Service Fabric artifacts and enterprise applications to a Foundation Pack environment and use LDAP as user repository.

## 6.9 Installing Fabric on z/OS

WebSphere Business Services Fabric V6.1 does not provide an interactive installer on z/OS. You must manually install the WebSphere Business Services Fabric Foundation Pack on a WebSphere Process Server environment on z/OS.

The WebSphere Process Server profile that you need to install WebSphere Business Services Fabric on must have global security enabled. In a network deployment environment, typically LDAP is used for the user registry, which authenticates the users.

The WebSphere Business Services Fabric Tool Pack is not supported on z/OS. The Tool Pack is typically installed on a developer workstation.

**Note:** Before you install Fabric on z/OS, verify the detailed system requirements at:

<http://www.ibm.com/support/docview.wss?rs=36&uid=swg27010719>

For detailed manual installation steps for installing the Foundation Pack on to an existing WebSphere Process Server for z/OS V6.1 environment, refer to:

[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.ws.fabric.install.doc/fpi/concept/c\\_getting\\_started\\_manual\\_installation.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.ws.fabric.install.doc/fpi/concept/c_getting_started_manual_installation.html)

The following list contains the high-level steps of the manual installation process for the Foundation Pack on z/OS:

1. Create and configure the Fabric database.

2. Create the following WebSphere Process Server resources for the Fabric environment:
  - Java™ Database Connectivity Provider to connect to the Fabric database.
  - Three data sources for:
    - BSR (Business services Repository)
    - PM (Performance Manager)
    - GM (Governance Manager)
  - DA Event topic for handling Dynamic Assembler events
  - DA Performance activation specification for the DA Event topic
  - A Fabric Service Integration Bus
  - Mail server configuration for sending Fabric notifications
3. Install the four Fabric enterprise application EAR files:
  - Fabric\_Tools: Installs the Fabric Administration console.
  - Fabric\_Tools\_Help: Installs the Fabric Web Help files.
  - Fabric\_Catalog: Installs the catalog services like Remote FCA Imports.
  - Fabric\_Engine: Installs the Dynamic Assembler runtime application.
4. Save the WebSphere Process Server master configuration file, and stop and start the server.

## 6.10 Known limitations

Currently WebSphere Business Services Fabric does not support the Local Operating System as the user repository; however, in production deployment, you typically use an external registry, such as LDAP, as the user registry. WebSphere Business Services Fabric integrates with a federated repository through WebSphere Process Server Virtual Memory Manager. For more information about how to configure a federated repository for WebSphere Business Services Fabric to use, see Chapter 17, “Integrating with Lightweight Directory Access Protocol” on page 481.

## 6.11 Clustering and failover

Cluster is a grouping of one or more server instances that improve the availability and workload management for your server environment. Figure 6-37 shows a typical cluster configuration for WebSphere Business Services Fabric.

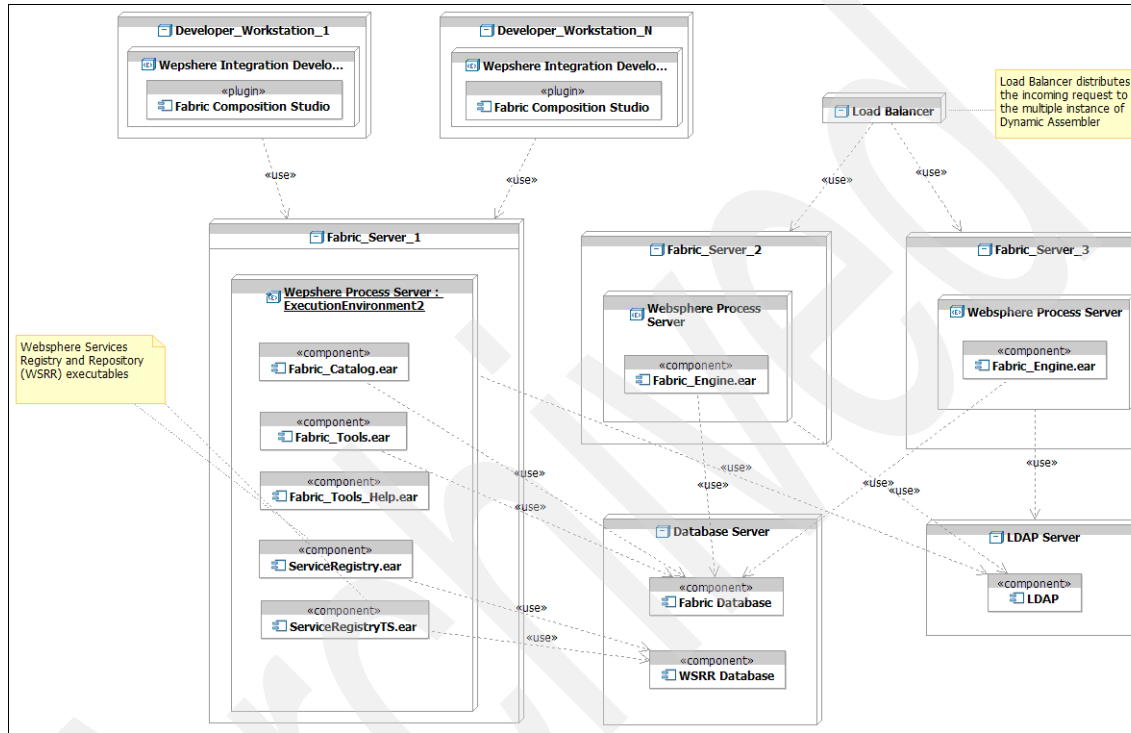


Figure 6-37 Cluster topology

As shown in Figure 6-37, we created three WebSphere Process Server instances, Fabric\_Server\_1, Fabric\_Server\_2, and Fabric\_Server\_3. Because the Foundation Pack gets deployed on top of WebSphere Process Server, the clustering environment for WebSphere Business Service Fabric is similar to the best practices that WebSphere Process Server cluster deployments follow.

For information about deployment topologies and clusters with WebSphere Process Server, refer to *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413.

Fabric\_Server\_1, Fabric\_Server\_2, and Fabric\_Server\_3 are part of a cluster:

- Fabric\_Server\_1 hosts Fabric Tools, Fabric Catalog, Fabric Help, Service Registry, and Repository applications.

WebSphere Service Registry and Repository applications are used for sourcing technical service meta data. The fabric-pm-mdb module, which is part of Fabric Tools, is the message driven bean that receives Dynamic Assembler events from configured Topics and inserts invocation data in Performance Manager tables. Configure the message driven bean across the cluster as per the best practices that we follow for the WebSphere Process server cluster configuration.

**Note:** For information about how to configure message driven beans with WebSphere MQ, refer to:

<http://websphere.sys-con.com/read/136459.htm>

- The Developer workbench contains WebSphere Integration Developer with Composition Studio connects to Fabric Tools for replication and synchronization of the Business Service Repository and the governance process.
- Fabric\_Server\_2 and Fabric\_Server\_3 are part of a cluster that hosts the Fabric Engine enterprise application, which comprises the Dynamic Assembler SCA libraries. The Dynamic Assembler supports Distributed Context Management, where the context is propagated across clusters. Deploy your business processes (SCA modules using Dynamic Assembler) on this clustered environment.
- Fabric\_Server\_1, Fabric\_Server\_2, and Fabric\_Server\_3 share the Fabric Database and LDAP User registry.

Currently there is no interactive installer to install WebSphere Business Services Fabric on a clustered environment. You must manually install WebSphere Business Services Fabric a clustered environment. Refer to the *Configuring WebSphere Business Services Fabric 6.1* section in the *FabricInstallGuide\_en.pdf* (located in the documentation\InstallGuide folder of the Foundation Pack installable) for setting up WebSphere Business Services Fabric 6.1 in a clustered environment. Alternatively refer to:

[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.ws.fabric.install.doc/fpi/task/t\\_configuring\\_wps\\_clustered\\_%20environment.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.ws.fabric.install.doc/fpi/task/t_configuring_wps_clustered_%20environment.html)



## Overview of developing a Fabric solution

In this chapter, we give you an overview of what it takes to develop a composite business application using WebSphere Business Services Fabric (Fabric).

A Fabric solution follows the principles of an SOA solution. We touch upon the IBM SOA Foundation life cycle and correlate the steps as related to developing a Fabric solution.

This chapter is classified into the following topics:

- ▶ 7.1, “Introduction to IBM SOA Foundation life cycle” on page 138
- ▶ 7.2, “Life cycle for developing composite business applications” on page 140

## 7.1 Introduction to IBM SOA Foundation life cycle

From the perspective of business, SOA is about modeling the business (in other words, capture the business design) and gaining insight from that effort to refine the business design.

IBM views SOA as a holistic relationship between the business and the IT organization. An effective enterprise SOA encompasses techniques, best practices, methodologies, programming model, and a set of supporting tools for implementing the business design in information systems. Further, SOA encompasses the middleware infrastructure to support such implementations. To top it all, SOA should possess capabilities to manage the end-to-end implementation to ensure availability to business and to ensure efficient usage of resources in the execution of that implementation.

The life cycle for an SOA solution begins by modeling the business, which includes business design parameters that include key performance indicators of business goals and objectives, translating the model into an information system design, deploying that information system, managing the deployment, and using the results that come out of that environment to identify ways to refine the business design. This life cycle should ensure that effective feedback is extracted from each of the iterative steps to facilitate the needs of the business.

The concepts are captured in the IBM SOA Foundation life cycle in Figure 7-1 on page 139.



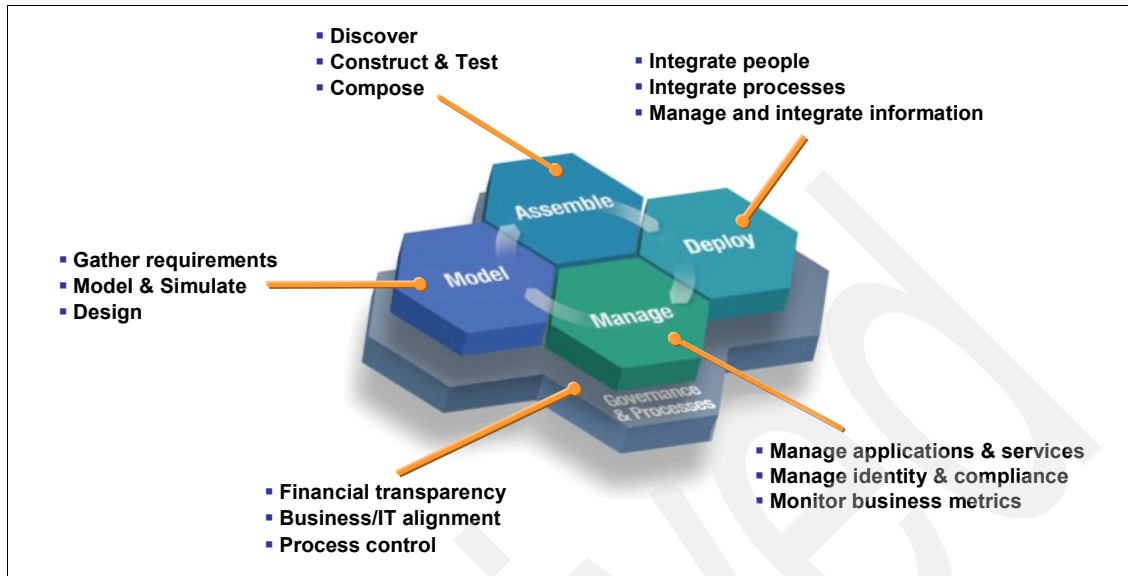


Figure 7-1 IBM SOA Foundation life cycle

The life cycle is layered on a backdrop of a set of governance processes that ensure that compliance and operational policies are enforced and that change occurs in a controlled fashion and with the appropriate authority, as the business design specifies.

**Note:** For elaborate discussions about the IBM SOA Foundation Life cycle, refer to the following resources:

Redbook:

*Patterns: SOA Foundation Service Creation Scenario*, SG24-7240

Web Links:

<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>

<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-enter4/index.html>

<http://www-128.ibm.com/developerworks/webservices/library/ws-enter5/index.html>

<http://www-128.ibm.com/developerworks/webservices/library/ws-sla4/>

<http://www-306.ibm.com/software/tivoli/products/composite-application-mgr-soa/>

## 7.2 Life cycle for developing composite business applications

A composite application is built using a set of related and integrated services, which support a business process that is built on SOA. It is the premise of these applications to provide effective feedback, in an iterative manner, to facilitate the needs of the business.

A life cycle for developing composite business applications complements the SOA Foundation life cycle to incrementally model the business context, transform into information systems design, assemble different constituent services to a composite service, deploy the composition, manage the deployed solution to extract information for business needs, and govern the entire infrastructure for operational efficiency and compliance.

In this section, we explain the life cycle iterative steps that are needed to build composite business applications using WebSphere Business Services Fabric.

## 7.2.1 Model

*Modeling* is the first step in planning a solution with Fabric. Modeling a composite business application starts by capturing the business design with the understanding of the business requirements, objectives and goals. The objective of the modeling phase is to achieve a set of specifications for information systems to effectively represent the business language, which consists of business processes, goals, policies, users roles, objectives, and so on.

### Business modeling

*Business modeling* is an exercise to understand the business domain in which composite business applications need to be built. The key step in business modeling is to capture the business domain, business environment, and associated business processes. This is a sophisticated method, which typically includes the use of advanced tools to simulate what-if scenarios with various inputs that can affect the way business runs. There are pre-built industry models that give industry and domain-specific knowledge and best practices for business models. (Though, not all industry domains have a formal industry model.)

The following list is the set of high-level iterate steps, not necessarily in sequence, that are involved in business modeling:

- ▶ Identify the business domain, identify the business use cases, and create the process models.

This is an important step to identify the business domain in which a composite business application is being built. Optionally, you can use industry process models as input to designing process models.

- ▶ Identify points of variability in the process models and arrive at a possible linear process model by taking out the points of variability.

The benefit of Fabric is in being able to simplify a complex business process into a linear process by extracting out points of variability. The intention of arriving at a linear process is to be able to apply assertions and business policies at runtime, which gives you the flexibility that business needs.

- ▶ Identify and document business roles and channels.

A composite business application has a set of business services that are provisioned through a set of channels to specific users. It is important to arrive at different business roles and their specific channels to apply business policies.

- ▶ Document business requirements.

Business requirements are primarily driven by the needs of the business. A team of Business Analysts and Business Architects are involved in scoping

the business requirements, taking business strategy and direction into consideration. Business requirements are typically unstructured in nature.

- ▶ Capture objectives, goals, and Key Performance Indicators (KPIs).

Business requirements are driven by a set of goals and objects. Record key performance indicators to measure them.

- ▶ Extract and specify business rules, assertions, and business policies.

It is important to make architectural decisions about how business rules, assertions, and policies are defined and implemented.

- ▶ Document Non Functional Requirements (NFRs).

Document NFRs, such as availability, performance, scalability, security, and so on. Define these requirements considering the constraints of the technology, skills, and budget.

A team of Business Architect, Business Analysts, and Solution Architects perform this phase of modeling.

The end deliverables of the business modeling activity are:

- ▶ Business Requirements Document that details goals, objectives, and key performance indicators
- ▶ Business Use Cases
- ▶ Business Process Models
- ▶ Business rules, business assertions, and business policies
- ▶ Non functional requirements from the business perspective

## **Solution modeling**

The *solution modeling* phase is closely associated with envisioning how business design can be specified in a way that can be used to develop information system implementations. Solution modeling is also greatly influenced by the existence of the industry models for data and services. Solution modeling also needs to take advantage of the enterprise assets and provide specific business functionality.

The following list contains the high-level iterative steps, not necessarily in that sequence, that are involved in a solution modeling phase:

- ▶ Create a use case model.

Business requirements are further polished and translated into a set of functional requirements that are encapsulated in a set of use cases. There are different ways of capturing use cases. An effective and structured way of capturing use cases is to use Unified Modeling Language (UML). These use

cases represent interaction between different actors (users of the system) and the system, illustrated as a set of simple steps. A use case diagram further captures the relationship between various use cases and sequence flow.

► Create a service model.

A service model consists of the list of all enterprise services that the composite business application will use. A service model must specify these services along with their messages, such as input, output, and faults. It further consists of the relationships between various services and some of the realization decisions taken for realizing the services on different endpoints.

This is an important and complex process for any organization to realize an SOA architecture. Service Identification, Specifications, and Realization decisions of the IBM service-oriented modeling and architecture (SOMA) method is an effective way to arrive at a service model.

**Note:** Service modeling is a wide topic in itself and is outside the scope of this book. Refer to the following Web site for further information about SOMA methodology:

<http://www.ibm.com/developerworks/library/ws-soa-design1/>

► Create information/data models.

A data model represents the key business entities that are used in the services implementation and their relationships. A data architect typically refers to an existing enterprise data model and industry data models, if available. There are many tools that are available to create logical representations of data and transform them to physical data models.

► Create component models.

A composite business application has many components that are logically grouped to provide a specific functionality. These components might use one or more packaged applications (existing assets) within the organization.

► Create operational models.

An operational model defines and documents the distribution of an IT system's components onto geographically distributed nodes together with connections that are necessary to support the required component interactions to achieve the functional and non functional requirements within the constraints of technology, skills, and budget.

- ▶ Create Fabric ontology extension models.

This is a UML model that extends the base ontology model of WebSphere Business Services Fabric. You can use this ontology extension to create new roles, channels, or new industry or solution-specific content assertions.

- ▶ Design and document the overall solution and integration architecture.

This is a set of documents that detail the overall composite business application implementation details and the integration techniques with other systems that are within the organization.

- ▶ Document architectural decisions to support the proposed solution.

Important decisions about any aspect of the architecture, which includes the structure of the system, the provision and allocation of function, the contextual fitness of the system, and adherence to standards, are documented as part of the solution architecture.

A team that consists of Solution Architects and Data Architects in consultation with Enterprise Architects performs solution modeling.

The end deliverables of the solution modeling activity are:

- ▶ Use case model
- ▶ Service model
- ▶ Data/information model
- ▶ Component model
- ▶ Operational model
- ▶ Fabric ontology extension model
- ▶ Overall solution architecture document
- ▶ Architectural decisions document

Figure 7-2 on page 145 illustrates the modeling phase with the inputs, outputs, and roles that are involved.

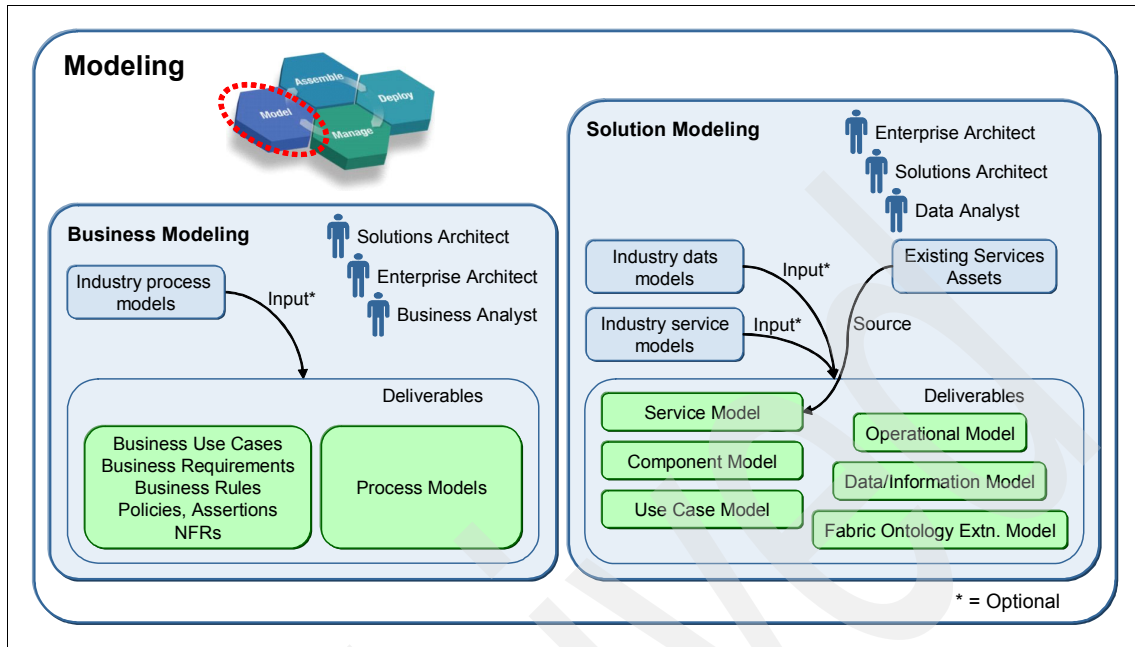


Figure 7-2 Modeling phase - Inputs, deliverables, and involved roles

The inputs, deliverables, and roles that are in Figure 7-2, are illustrative in nature and are not comprehensive, for example, a solution modeling phase can also include a user experience model that explains the requirements for the user interfaces for the composite application.

Software modeling is a broad and elaborate topic and is outside of the scope of this book. The example considered for illustration using ITSOBank is intended to explain the design and development methods and concepts around WebSphere Business Services Fabric.

### Modeling for the ITSOBank Loan Process

For the ITSOBank scenario, a sample process for loan processing is considered along with a set of business use cases and business requirements. We explain the business modeling and key business challenges, non functional requirements, performance indicators, and a high-level process model for the loan composite application in Chapter 5, “Business scenario for this book” on page 85.

Chapter 8, “Modeling Fabric solutions” on page 153, explains how to extract a simplified linear process along with points of variability. It further concentrates on

creating an extension to the Fabric Ontology Model to support the assertions and business policies.

The end of the modeling phase results in an enhanced SOA solution stack, as shown in Figure 7-3.

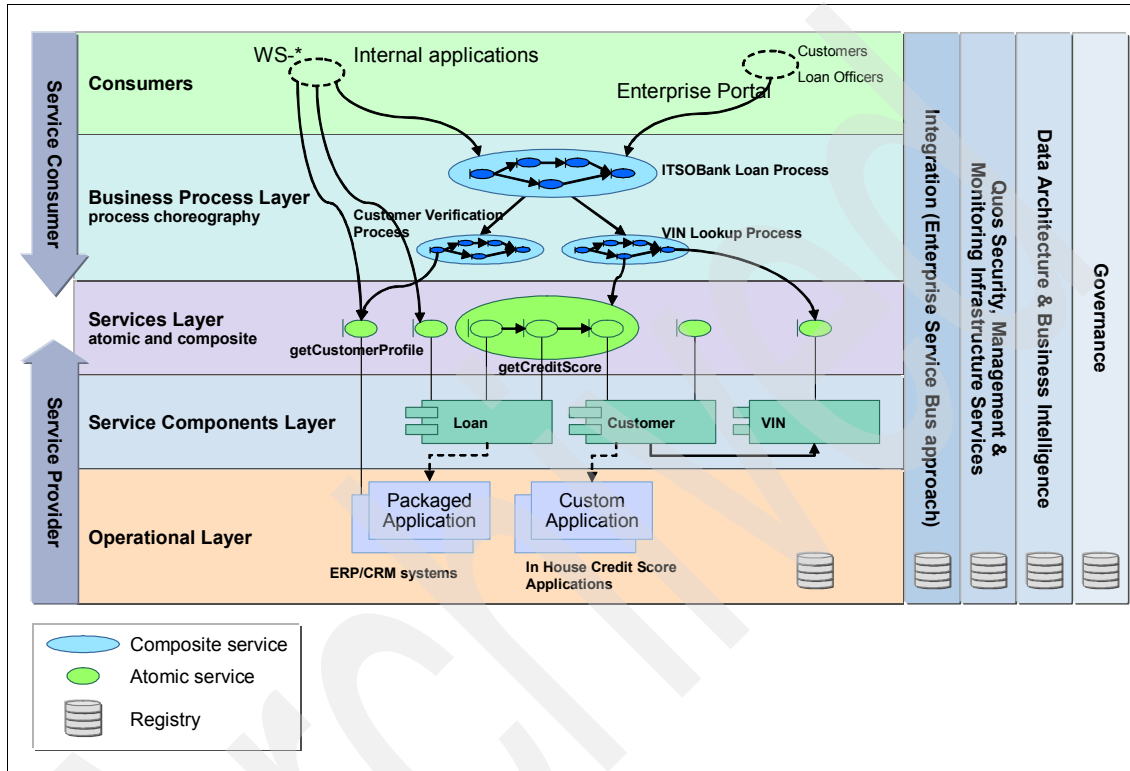


Figure 7-3 SOA solution stack example for the ITSO Bank Loan process

## 7.2.2 Assemble

Before the beginning of the assemble phase, you should have a fairly good amount of information about how the business design is to be implemented in terms of IT assets. This is done by *assembling* the information system artifacts that are developed using the business design, which we explained in the modeling phase.

A team of Enterprise Architects and Solution Architects can convert the business design into a set of business process definitions and interfaces by deriving the required services from the business process model and activity definitions in the use case model.



It is an important step to search for existing runtime assets, packaged application components, and heritage systems, which can provide the required functionality of the business process. You might need to augment some of these assets to meet the requirements of service design to assemble them into a composite application. Consider these assets while you come up with a service model during the modeling phase. The search for assets here is more from an assembly perspective.

From the perspective of composite business applications, we assume that an activity to identify these assets and enable them technically to be assembled in a business service is already taken care of. Information about the constituent runtime services are made available through an enterprise services registry, such as WebSphere Service Registry and Repository (WSRR) and services meta data.

The assemble phase for composite applications is a development activity and is done in a development environment that consists of a Unit Test Environment. Developers in this environment assemble and unit test their assembly for completeness, correctness, and consistency.

The following list are the steps that are involved during the assemble phase:

1. Implement constituent service components in cases where an existing application component does not provide the required functionality.
2. Create a wrapper module in WebSphere Integration Developer to implement the business process implementation, which consists of Dynamic Assembler components and wire necessary components, services, and extensions.
3. As an administrator of Fabric, use the Web administration console to create the necessary organization structure in the development environment.
4. Create Fabric namespaces, project, and test environments using the Web administration console.
5. Import the ontology extensions by creating a Fabric Content Pack, which you can do by using the Fabric Ontology extension model, running the transformations, and packaging the same to a Fabric Content Pack (FCP).
6. Synchronize this project in Composition Studio, and create application suites and applications.
7. Source services meta data for existing services from an enterprise services registry and repository, such as WSRR.
8. Import the wrapper module into the composite services of Composition Studio (Business service perspective in WebSphere Integration Developer), which creates the necessary composite services and their endpoints.

9. Using Composition Studio, create other business services, context specifications, service interfaces, endpoints, assertions, and policies.
10. Simulate policies to make sure that policies result in proper endpoint selections as per the linear business process defined at the time of modeling.
11. Govern the assembling activities, and publish the artifacts into Business Services Registry.

The Assembly phase primarily takes care of how these composite applications operate in the production environment, which might include taking care of the business and government regulations, packaging, localization constraints, resource dependency, integrity control, access protection, and so on.

Perform any user interface (UI) requirements in parallel so that the UI artifacts are ready for deployment.

### **Assembling the ITSOBank Loan Processing composite application**

For the ITSOBank example that we use in this book, we showcase how different constituent services are wired together for the loan process. We further explain how to use Composition Studio and WSRR to create the composite business application that is bundled with business policies and assertions. In Chapter 9, “Assembling Fabric solutions” on page 193, we show how policy simulation can prove to be a powerful tool to understand the selection of endpoints based on assertion criteria and policies.

## **7.2.3 Deploy**

The *deploy* phase deals with creating a hosting environment for the actual deployment of the artifacts that belong to the composite business application, which includes resolving the application’s resource dependencies, operational conditions, capacity requirements, and integrity and access constraints.

The following list contains a high-level set of activities during the deploy phase:

1. Create the hosting environment, as per the operational model that is generated during the modeling phase for hosting.
2. Set up the necessary integration infrastructure, and resolve any dependencies and integration issues.
3. Using Composition Studio, deploy the business services and meta data to the hosting Fabric environment, which is followed by a publishing of services meta data to the production environment by a Fabric Administrator.

4. Deploy the composite business application artifacts, such as Enterprise Archive (EAR) to the hosting environment. Deploy the required user interfaces.
5. Use the Subscription Manager of the Fabric administration console to subscribe various users that are associated with various roles to different business services.
6. Govern the deployment activities.

The deploy phase considers techniques that need to be employed to ensure availability, reliability, efficiency, integrity, and service availability.

Figure 7-4 shows an overview of how different artifacts flow between the involved products.

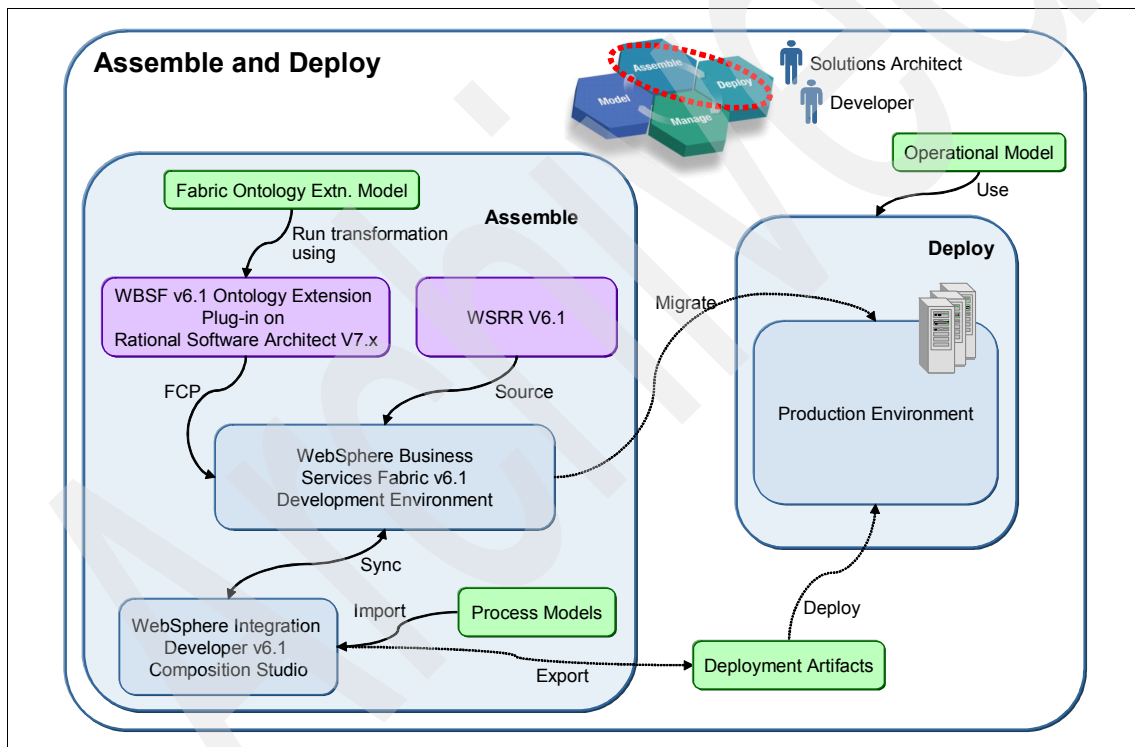


Figure 7-4 Assemble and deploy artifacts flow

## Deployment for the ITSOBank Loan Processing composite application

For the Loan Processing composite application, which we first discuss in this book in Chapter 10, “Deploying Fabric solutions” on page 289, we describe the way

the different modules are deployed in the hosting environment for ITSOBank. These modules are developed and tested using the Unit Test Environment, which is installed on top of WebSphere Integration Developer.

## 7.2.4 Manage

The *manage* phase deals with considerations for maintaining the operational environment and the policies that are expressed in the assembly of the composite business applications, which are deployed to the hosting production environment.

The manage phase also includes managing the business model, tuning the operational environment to meet the business objectives that are expressed in the business design, and measuring success or failure to meet those objectives.

The manage phase includes the following activities, not necessarily in sequence:

- ▶ Monitor the performance of the requests that are coming to various services of the composite business application.

For this purpose, you can use either WebSphere Business Services Fabric performance manager or integration with IBM Tivoli Composite Application Manager for SOA.

- ▶ Manage changes to the business policies, assertions based on the business need, and requirements.

This is a critical success factor because the hosting environment that represents a business process should effectively give feedback in order to change the business model that is based on changing market conditions.

- ▶ Manage subscriptions of users to various business services, which also includes managing security domains, such as authentication, authorization, single sign-on, user provisioning, federated identity management, and so on.
- ▶ Maintain problem logs to detect failures in various services and system components, localize failures, and restore the operational state of the system.

### **Management of the ITSOBank Loan Processing composite application**

For the ITSOBank Loan Processing composite application, we showcase the parameters that guide the management of the composite application. We explain how to manage the subscribers and performance of different business services. Refer to Chapter 16, “Integrating with IBM Tivoli Composite Application Manager for SOA” on page 459.

## 7.2.5 Govern

Each phase of a composite business application solution needs governance. Governance principles that are adapted for any SOA infrastructure holds good for composite business applications, which are built on SOA principles.

Some of the core objectives of governance in the context of composite business applications are:

- ▶ Ensure maturity of the service orientation within the enterprise.
- ▶ Assign decision rights, policies, measures around services, processes, and the entire composite business application life cycle.
- ▶ Measure effectiveness of the services, and ensure continuous improvement.
- ▶ Define roles and responsibilities within the organization from an IT application infrastructure perspective.
- ▶ Ensure adoption of industry and open standards within the enterprise.
- ▶ Manage services infrastructure enhancements from the point-of-view of registration, discovery, modeling, development, consumption, publishing, ownership, funding, auditing, diagnostics, security, performance, monitoring, versioning, provisioning, shared usage, and so on.

**Note:** For a detailed discussion about SOA governance, refer to the following:

- ▶ Redbook - *Implementing Technology to Support SOA Governance and Management*, SG24-7538
- ▶ Redpaper - *Case Study: SOA Governance Scenario*, REDP-4384

### Governance in the ITSOBank

In this book, we explain governance through a scenario, where considerable changes, with respect to business hierarchy of the bank, are expected because of new acquisitions. We explain how this can be dealt within the ITSOBank and how changes in the business landscape can further be handled in the ITSOBank Loan Processing composite application.

## 7.3 Conclusion

In this chapter, we introduced the IBM SOA Foundation life cycle and touched upon a solution development life cycle for developing composite business applications using WebSphere Business Services Fabric. While it is possible to view the process of developing and managing a composite application using a life cycle, the progression of activities are not always linear and do not follow the

exact steps of the life cycle that we described in this chapter. Operational environment is often dynamic, and key changes to the business model might affect the operating environment without following the life cycle steps.

Information technology constraints that you establish in the assembly phase might limit the business design that is created during the model phase. It is important to configure multiple environments during deployment for test purposes before you migrate the composite business application environment into the production or hosting environment.

# Modeling Fabric solutions

Modeling is the first step within the Fabric solution life cycle, which consists of Model, Assemble, Deploy, and Manage. In this chapter, we describe the practical steps in modeling a WebSphere Business Services Fabric (Fabric) solution.

This chapter contains the following sections:

- ▶ 8.1, "Introduction" on page 154
- ▶ 8.2, "Identifying process variability points" on page 156
- ▶ 8.3, "Identifying context information" on page 156
- ▶ 8.4, "Policy identification" on page 160
- ▶ 8.5, "Simplified process model" on page 161
- ▶ 8.6, "Fabric ontology core model" on page 162
- ▶ 8.7, "Modeling extensions" on page 165

## 8.1 Introduction

The Fabric Business Service Model provides a rich domain for modeling composite business applications. Using Composition Studio and the Fabric Administrative Console, you can create Business Service Model instances and store them in the Business Service Repository as metadata solutions. After the metadata is available in the Business Service Repository, other developers can use the metadata to:

- ▶ Understand solution's components
- ▶ Reuse solution's components
- ▶ Refine solution's behavior
- ▶ Manage solution change requirements
- ▶ Drive new changes in solution

In this chapter, we cover the business and solution modeling of the solution. We explain the extraction of a simplified linear process along with points of variability. We also discuss creating an extension to the Fabric Ontology Model to support the assertions and business policies.

### 8.1.1 Definition of modeling terms

The following definitions are relevant to the modeling process:

- ▶ **Industry Source Model**  
This model is typically an industry standard, such as ACORD, HL7, or an IBM industry model, such as IFW/IAA.
- ▶ **Industry Glossary Model**  
This model is part of the Industry Business glossary and is a UML representation of the industry terms, which includes their properties, relationships to other terms, and so on.
- ▶ **Industry Core Model**  
This model is part of the Industry Content Pack and is an UML representation of Fabric concepts, such as assertions, channels, and roles.
- ▶ **Stereotype**  
A way of highlighting certain classes to be used to perform transformations.
- ▶ **Transformation**  
A method by which a UML model that is defined in one notation is converted into another UML model or a text.



- ▶ **Property**  
Refers to an attribute of a class or a stereotype.
- ▶ **Profile**  
Refers to a collection of stereotypes that you can apply to an UML project.

## 8.1.2 Steps in modeling a Fabric solution

In this section, we provide the steps that are involved in modeling a Fabric solution:

1. Identify points of variability in the process models and arrive at a possible linear process model by taking out the points of variability. All of the points of variability are dealt with by leveraging the features of Fabric, which simplifies the business process.
2. Identify context information. Identify and document business roles and channels. Identify content-based assertions.
3. Identify policies. In a Fabric solution, a policy is a set of assertions that represent requirements, constraints, or capabilities for a business service. It includes policy targets, context, and content-based assertions.
4. Create Fabric ontology extension models, which is a UML model that extends the base ontology model of WebSphere Business Services Fabric. You can use this ontology extension to create new roles, channels, or new industry or solution-specific content assertions. Ontology modeling is done using the Fabric Modeling Tool in Rational Software Architect. This ontology model is then exported as a Fabric Content Archive (FCA) file. The ontology extensions are packaged as an FCA file and uploaded to the Business Service Repository using the Fabric Governance Manager Console. An FCA file is similar to other packaging constructs, such as JAR and WAR.

The following subsequent steps are covered in Chapter 9, “Assembling Fabric solutions” on page 193:

1. Create Fabric project and namespaces in Fabric. A Fabric project is owned by a development team and contains one or more owned namespaces and any number of imported namespaces. The ontology model FCA file is imported within this Fabric project. The Fabric project, namespaces, subscribers, and repository are then configured.
2. Configure environments. An environment represents a system where endpoints are deployed, for example, a service might be deployed in a development, test, training, or production environment. Finally, environments are configured and exported to be reused by the developers.

## 8.2 Identifying process variability points

*Points of variability* refers to the variances in a business process that cause the business process to be complex and nonlinear; therefore, it is in your best interest to extract and manage the variances outside of the business process. These variances are captured as a part of the Fabric meta model. When you manage the variances outside of the business process, you can drive changes into the business process using configuration data (meta models) rather than by changing design. The business process then becomes more flexible and adaptive for changes.

To model a Fabric solution:

1. Define the abstract business process.
2. Identify the variability points in the business process.
3. After you know the points of variability, you can leverage the features of the Fabric solution to implement the solution.

For the ITSOBank scenario, the following process variability points are identified:

- ▶ Loan Amount, for example, more than \$20,000
- ▶ Customer Type, for example, Premium, Regular, New
- ▶ Bank Identifier, for example, ITSOBank
- ▶ Hours of Availability, for example, Business Hours, Non-Business Hours
- ▶ Rating, for example, High, Medium, Low
- ▶ Credit Check Cost, for example, \$0 (internal), \$5 (external), \$10 (external)
- ▶ Service Deprecation, for example, multiple versions of Service

## 8.3 Identifying context information

A *context specification* specifies what context is needed at run time for a Dynamic Assembly to make the endpoint selection decision. There are several advantages to using a context specification. Because the context at run time determines how policies are applied, it is important to ensure that all needed context is provided with every invocation. A context specification that indicates that a particular dimension is required ensures that this dimension filled in the context at run time.

The dynamic selection of service endpoints for bindings at runtime, on a request by request basis, is called Dynamic Assembly. When using WebSphere Business Services Fabric, service consumers do not directly bind to service provider endpoints. Instead the service is called through the Dynamic Assembler:

1. The Dynamic Assembler determines which endpoints to use based on the incoming requests.
2. The Dynamic Assembler examines the incoming request and applies the logic defined in terms of Business Policies and Assertions for endpoint selection.
3. The Dynamic Assembler invokes the appropriate service endpoint.

Using this method, the service provider endpoints are virtualized from the service consumer and not called directly.

A context specification clearly indicates which items in the context are relevant for a particular decision point, and it provides a prebuilt set of entry fields for new simulations of a Dynamic Assembly decision.

### 8.3.1 Roles and channels

*Roles* are typed associations between a user and an organization. When creating this relationship, administrators can pick from an extensible taxonomy of role types, for example, a user is assigned the role of Premium Customer with ITSOBank. A policy can then be used to ensure faster response times for premium customers.

A *channel* is the mechanism through which messages are sent to business services. You can provision a business service to one or more channels.

Table 8-1 shows the roles and channels that we identified for the ITSOBank scenario.

Table 8-1 Identified roles and channels

Roles	Channels
<ul style="list-style-type: none"><li>▶ Loan Officer</li><li>▶ Customer</li><li>▶ Loan Agent</li></ul>	<ul style="list-style-type: none"><li>▶ Internet</li><li>▶ B2B</li><li>▶ Phone</li></ul>

### 8.3.2 Content-based assertions

In a Fabric solution, *assertions* are characteristics that describe the capabilities of an endpoint. You can define these capabilities using five dimensions: Performance, Reliability, Interoperability, Security, and Manageability, for example, testing can reveal that the maximum transaction time for a service is 500 milliseconds. A maximum transaction time assertion can be added to the endpoint to declare that it has this performance capability.

During dynamic assembly, the selection policy filters the set of candidate endpoints for an interface. If the selection policy has a required assertion in its contract, then an endpoint must have a matching assertion to remain a candidate. The matching is determined by the comparator of the assertion. Assertion comparators are determined when modeling the assertion.

## **Making endpoint assertions required**

During endpoint selection, the set of candidate endpoints is filtered by eliminating any endpoints that do not match the assertions in the selection policy. Then, endpoints can be further eliminated when the system evaluates the requirements of the endpoints.

If an endpoint assertion is marked as required, the endpoint is not picked unless that assertion is matched in the selection policy. When you use the required flag on an endpoint, take care that your assertions have comparators that are symmetric, for example, it is not a good idea to use the required assertion with Maximum Transaction time because this assertion uses a Less Than comparator to ensure that the endpoint's time is less than that specified in the policy. An endpoint requirement enforces the reverse to be true as well.

## **Special assertions**

There are some special assertions that you can only add to endpoints:

- ▶ **Hours of Operation**

This assertion specifies when an endpoint is available. Using this assertion, you can define several repeating intervals to define the "open hours" for an endpoint, for example, an endpoint might be available from 7 a.m. to 10 p.m. Monday through Friday and from 8 a.m. to 2 p.m. on Saturdays. Rather than putting an assertion in the policy, the system automatically compares the time of invocation with the hours-of-operation of an endpoint.

- ▶ **Deprecation Assertion**

This assertion provides a sunset period for an endpoint, for example, a service becomes deprecated on September 9, 2009 and goes offline on December 12, 2009. When the deprecation date is reached, the Dynamic Assembler no longer selects that endpoint.

- ▶ **Propagate Policy Assertion**

This assertion specifies that an endpoint is capable and interested in receiving all or part of the selection policy that was used to select it. If the selected endpoint has this assertion, the invocation injects a header that conveys the values of the assertion properties to the endpoint. The endpoint can then use these settings to customize the behavior of the service.

## Identified assertions

You can extend the Fabric core ontology to add assertions that are custom to their enterprise. At run time, Fabric uses these characteristics to find the best-suited endpoint or service realization for a consumer based on their business requirements. You can attach assertions to endpoints to specify a particular characteristic of an endpoint, for example, maximum transaction time available.

Table 8-2 lists the identified assertions for the ITSOBank business scenario.

*Table 8-2 Identified assertions for the ITSOBank scenario*

Assertion name	Description	Points of variability	Assertion type	Use case
Loan Amount Assertion	datatype-int	Loan Amount (>20K, <20K)	Fabric → Assertions → Content Based Assertion	Policy
Customer Type Assertion	datatype-enumeration String	Customer Type (Premium, Regular, New)	Fabric → Assertions → Content Based Assertion	Both
BankIdentifier Assertion	datatype-string	Bank Identifier (ITSO)	Fabric → Assertions → Content Based Assertion	Endpoint
Hours Of Availability Assertion	NA	Hours of Availability	Special Assertion	Endpoint
Rating Score	datatype-enumeration String	Rating (High, Medium, Low)	Fabric → Assertions → Content Based Assertion	Both
Credit Check Cost Assertion	datatype-string	Credit Check Cost (10, 100)	Fabric → Assertions → Content Based Assertion	Endpoint
Service Deprecation Assertion	NA	Service	Special Assertion	Endpoint

Out of these, 'Hours Of Availability' and 'Service Deprecation' are available in Fabric as special assertions that you can use directly. You model the remaining assertions manually in Rational Software Architect using the Fabric Modeling Tool plug-in, which we explain further in 8.7, "Modeling extensions" on page 165.

## 8.4 Policy identification

Think of a *policy* as a simple "If-Then" rule of the following form:

IF <policy conditions> = true THEN <contract assertions> are applied.

At runtime, the conditions of each policy are evaluated against the current context to determine which policies are effective. These effective policies are merged into a single composite policy that is used for endpoint selection.

### 8.4.1 Policy definition

In a Fabric solution, a policy is a set of assertions that represent requirements, constraints, or capabilities for a business service. A policy consists of three sections:

- ▶ The context of service request
- ▶ The request itself
- ▶ The requirements that must be met

#### Policy targets

You must associate all policies as the policy target with some object in the Business Service Model. In addition to associating policies with a primary object, the target plays a role in determining how to merge two policies that have the same assertion in their contracts. The natural ordering of the policy target is used to disambiguate which assertion to apply.

#### Conditions based on model dimensions

A model dimension represents the set of possible instances for a given type, such as, a business service or interface. For each of the supported model dimensions, there is a special context key for which the URI identity of a particular instance is set.

#### Conditions based on content-based assertions

You set property values for content-based assertion properties in the context using the assertion property URI as the key. You can then construct expressions based on these properties.

In Composition Studio, the policy editor allows you to select a single policy target. It also allows you to add up to two additional conditions, where one condition is a conjunction or disjunction of model dimensions and the other is a conjunction or disjunction of expressions that are based on content-based assertion properties.

## **8.5 Simplified process model**

In this section, we discuss:

- ▶ 8.5.1, “Comparison with the core BPM approach” on page 161
- ▶ 8.5.2, “Streamlined process in Fabric” on page 161

### **8.5.1 Comparison with the core BPM approach**

The original ITSOPBank business scenario model shows the business process modeled using the functionality in WebSphere Business Modeler. If this process is realized using the core Business Process Management (BPM) approach, the decision points are modeled in process choreography itself.

This approach is static because making any changes in the process flow involves design and development changes; therefore, it becomes difficult to manage the business process with this model when the number of services and process variability points increase. This approach offers reduced flexibility, and to incorporate changes in business processes requires that the developer make changes in the business process and redeploy it again, for example, adding a new category of Customer Type, requires another If-Then-Else construct in the business process to invoke a particular endpoint, which satisfies the requirement for the new category of Customer. So, this situation requires another iteration of design, development, and deployment of the business process. Also without the proper governance process in place makes it difficult to identify the impact of the change to the overall business process.

### **8.5.2 Streamlined process in Fabric**

Now let us look at how Fabric solution model removes the limitations and enhances the core BPM capabilities to create flexible SOA applications. Figure 8-1 on page 162 shows how the ITSOPBank model can be streamlined using the Fabric solution.

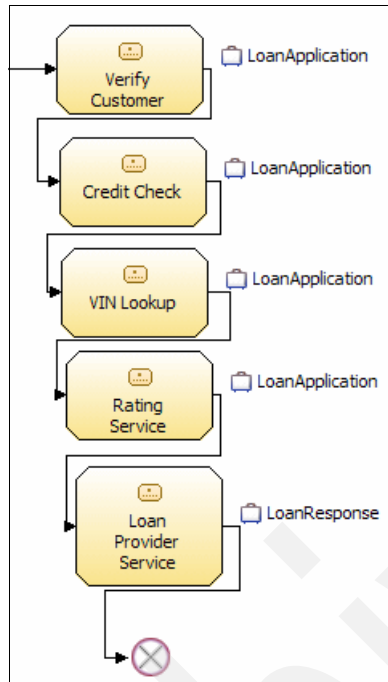


Figure 8-1 Streamlined process model

As you see in Figure 8-1, the process variability points are abstracted out from the process, which makes the process more generalized. The process variabilities that we identified earlier are captured as business policies while assembling the Fabric solution. The context that is required for each of the services is provided by extending the Fabric model. Based on context, content, and contract of the service request, the right implementation (endpoint) is selected and executed at runtime.

## 8.6 Fabric ontology core model

Model-driven software development drastically alters the software development process in that a model-driven architecture constructs a software system by designing models that are translated into executable software systems by code generators that operate in the background.

The Fabric Business Service Model provides a rich domain for modeling composite business applications. Using Composition Studio and the Fabric Administrative Console, developers can create instances of the types that the Business Service Model defined and store them to the Business Service



Repository. Rational Software Architect with the Fabric Modeling Tool plug-in (detailed later in the chapter) is used to model ontology extensions.

After this metadata is available in the Business Service Repository, other developers can use the metadata to understand and reuse solution components, refine solution behavior, and so on. The Business Services Dynamic Assembler can also use the metadata to make routing decisions at runtime.

### 8.6.1 Governance domain

The governance domain provides important concepts for teams working together and sharing content across multiple physical environments:

- ▶ Fabric projects and namespaces

A Fabric project is owned by a development team and contains one or more owned namespaces and any number of imported namespaces.

- ▶ Environments

An environment represents a system where you can deploy endpoints, for example, you can deploy a service in a development, test, training, or production environment.

### 8.6.2 Business service domain

The business service domain includes business services, applications, and application suites:

- ▶ Business services

A business service is a business function whose execution can be adapted at runtime based on business policies and user context.

- ▶ Applications

An application or composite business application is a logical collection of related business services.

- ▶ Application suites

An application suite is a logical grouping of related applications.

### 8.6.3 Service consumer domain

The model for service consumers allows you to write policies that customize behavior based on organizations, roles, specific users, or service level agreements:

- ▶ Organizations  
Models a business entity and the people that work within it.
- ▶ Users  
Can be found in any federated repository and assigned roles with organizations that were defined using the Subscriber Manager.
- ▶ Roles  
Typed associations between a user and an organization.
- ▶ Service levels  
Represents established levels of service.

### 8.6.4 Service technical domain

The service technical domain is composed of interfaces, composite services, dynamic assembly components, and endpoints:

- ▶ Interfaces  
Equivalent to Web Services Description Language (WSDL) port types.
- ▶ Composite services and dynamic assembly components  
You create composite services in Composition Studio by importing Service Component Architecture (SCA) module projects from the same workspace. Because a Dynamic Assembly component knows its own identity at runtime, the system also automatically supplies the composite service and dynamic assembly component dimensions in the context.
- ▶ Endpoints  
Represents a location where a service can be invoked. An endpoint indicates which interfaces it supports and which environments it is deployed in. The address of an endpoint provides the details on the location where the service can be invoked.

## 8.7 Modeling extensions

You can model extensions using the Fabric Modeling Tool (FMT), which is a plug-in for Rational Software Architect. The Fabric Tool Pack contains an UML model representation of Fabric core concepts that are available as stubs. You can extend classes from this model inside the industry core model to create new assertion, channel, or role types.

In this section, we discuss how to:

- ▶ 8.7.1, “Installing plug-ins and importing profiles” on page 165
- ▶ 8.7.2, “Creating the business glossary model” on page 169
- ▶ 8.7.3, “Creating the ontology model” on page 175
- ▶ 8.7.4, “Glossary transformation” on page 181
- ▶ 8.7.5, “Ontology modeling” on page 184
- ▶ 8.7.6, “Ontology transformation” on page 189
- ▶ 8.7.7, “Exporting the Fabric Content Archive” on page 192

### 8.7.1 Installing plug-ins and importing profiles

The following list contains the pre-requisites to use the Fabric Modeling Tool in Rational Software Architect. You need Rational Software Architect V7 or above for this plug-in:

- ▶ Extensibility Features (all the features, excluding Reusable Asset Specification)
- ▶ WSDL and XSD modeling and transformations
- ▶ WebSphere Business Services Fabric Core Stubs (available as part of the Fabric Tool Pack V6.1 under the modeling folder)

The following steps summarize the installation of the Fabric Modeling Tool plug-in into Rational Software Architect:

1. Open Rational Software Architect.
2. Click **File** → **New** → **Other**. Expand **RAS** → **RAS Repository Connection**, and click **DeveloperWorks Repository**, as shown in Figure 8-2 on page 166. Click **Next**.

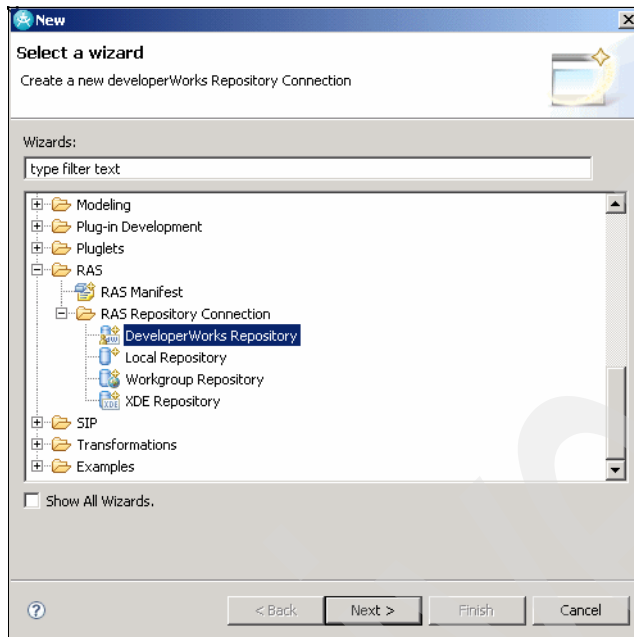


Figure 8-2 New connection to Repository

3. Confirm the Developer Works Repository location, as shown in Figure 8-3. Click **Finish**. A connection to the repository is created.

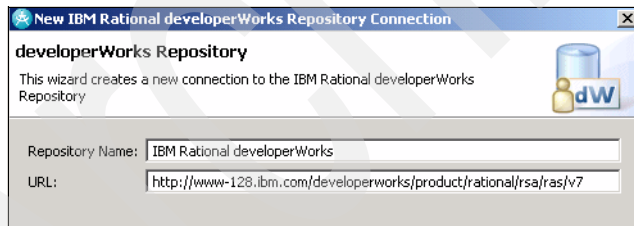


Figure 8-3 Connection to developerWorks Repository location

4. Click **File** → **Import**:
  - a. In the dialog box, expand **RAS**, and click **RAS Asset**.
  - b. Click **Next**.
  - c. Click **Browse** next to the Repository.

- d. In the resulting dialog box, expand **IBM Rational developerWorks**, and click **IBM WebSphere Business Services Fabric Modeling Tool**. Click **OK**.
- e. As shown in Figure 8-4, click **Next**.

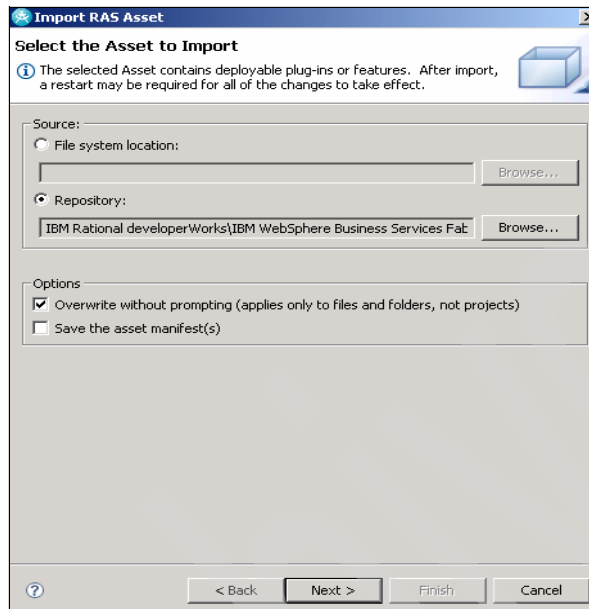


Figure 8-4 Import the Fabric Modeling Tool Asset

5. In the next panel, accept the License terms by selecting the check box. Click the **Finish** button to import the Asset from the repository.
6. Import the Fabric core stubs project:
  - a. Click **File** → **Import**, expand **Other**, and click **Project Interchange**. Click **Next**.
  - b. Click **Browse** for **Select Archive File**, and specify the location of the fabric-core-stubs-project-interchange.zip file. Make sure that you select **Fabric-Model-Stubs**, and click **Finish**, as shown in Figure 8-5 on page 168.

**Note:** The fabric-core-stubs-project-interchange.zip file is located in fabric-core sub-folder as part of the modeling folder, which is part of the Fabric Tool Pack installation.

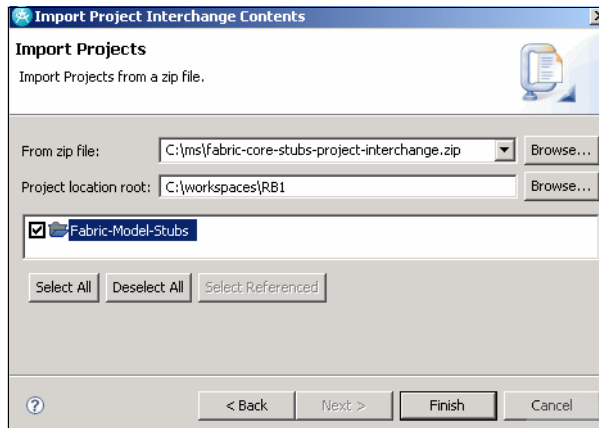


Figure 8-5 Import Fabric Model Stubs project

7. Import the Fabric profile:

- a. Click **File** → **Import**, and click **Other** → **Project Interchange**.
- b. Click **Browse** for **From Zip File**, and specify the location of the wbsf-profile-project-interchange.zip file. This zip file is located in the profiles sub-folder as part of the modeling folder, which is part of the Fabric Tool Pack installation.
- c. Select **wbsf\_profile** and, click **Finish**, as shown in Figure 8-6.

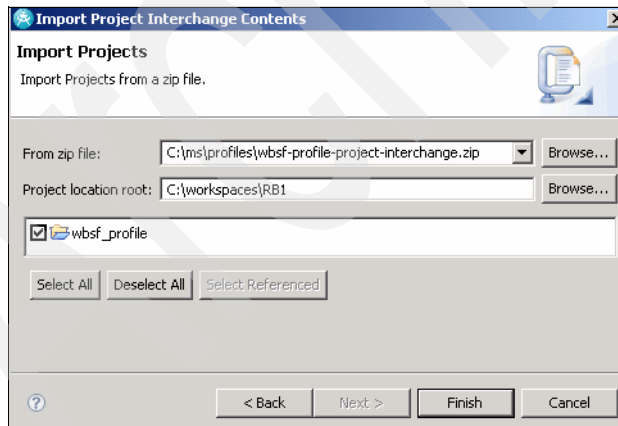


Figure 8-6 Import Fabric profile

8. Import the Glossary profile:

- a. Click **File** → **Import**, and click **Other** → **Project Interchange**.

- b. Click **Browse** for **From Zip File**, and specify the location of the wbsf-glossary-profile-project-interchange.zip file. This zip file is located in the profiles sub-folder as part of the modeling folder, which is part of the Fabric Tool Pack installation.
- c. Select **glossary\_profile**, and click **Finish**, as shown in Figure 8-7.

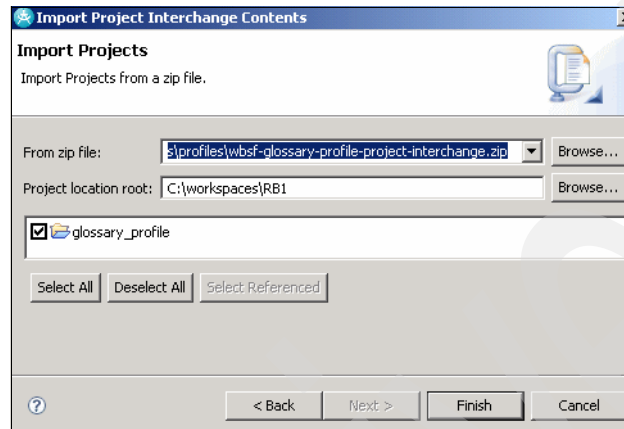


Figure 8-7 Import Glossary profile

This completes the installation of the plug-in. Figure 8-8 shows the Modeling view after the installation is finished.

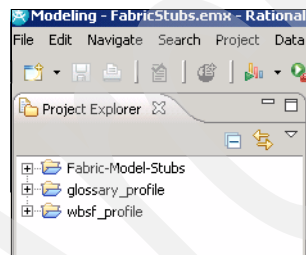


Figure 8-8 RSA Modeling View with installed plug-in

## 8.7.2 Creating the business glossary model

After the Fabric Model Tool plug-in is available, you can create a business glossary model, which is comprised of standard industry terms that are represented as ontology classes with properties in UML notation. The industry core model comprises of assertions, roles, and channels.

To create the business glossary model:

1. Create a new UML project and UML model:
  - a. Click **File** → **New** → **Other**, and click **UML Project**. Click **Next**.
  - b. Specify the Project Name as `VehicleLoanExtensions`, as shown in Figure 8-9. Click **Next**.

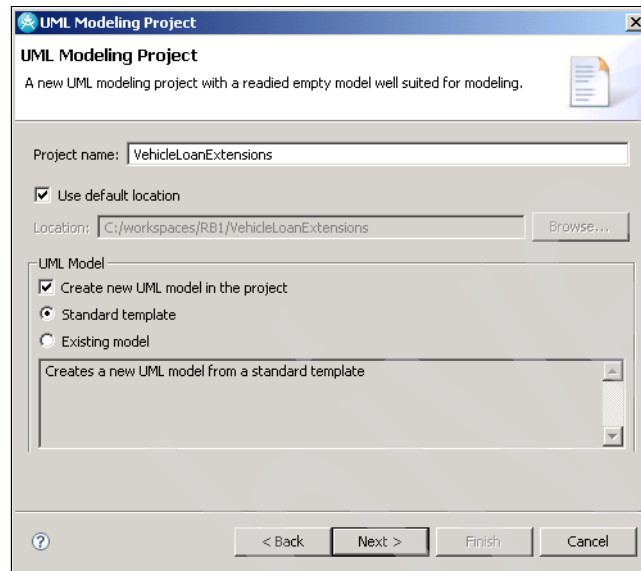


Figure 8-9 Create UML project

- c. Type the File Name as `VehicleLoanExtensionsModel`. In the Templates section, click **Blank Model**, and clear the **Create a default diagram in the new model** check box, as shown in Figure 8-10 on page 171. Click **Finish**. If you are prompted to change the perspective to the Modeling perspective, click **Yes**.



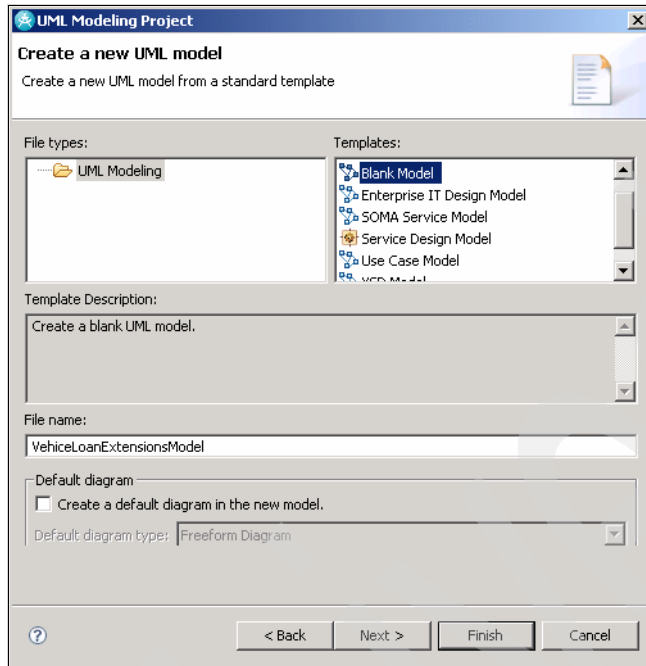


Figure 8-10 Create new UML Model

2. Apply the business glossary profile:
  - a. Double-click the **VehicleLoanExtensionModel** to open the model.
  - b. On the VehicleLoanExtensionModel view, click the **Details** tab. In the **Applied Profiles** section, click **Add**.
  - c. Click **Profile in Workspace**, and click **Browse**.
  - d. Add the Business Glossary Profile by expanding **glossary\_profile** and clicking **WBSFBusinessGlossaryProfile.epx**, as shown in Figure 8-11.

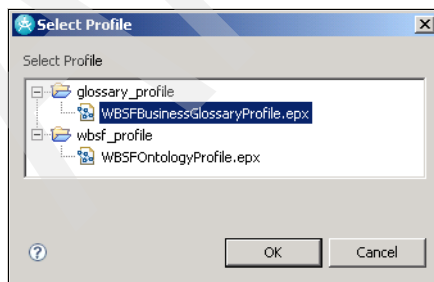


Figure 8-11 Select Business Glossary Profile

e. Click **OK**. The selected profile is displayed, as shown in Figure 8-12.

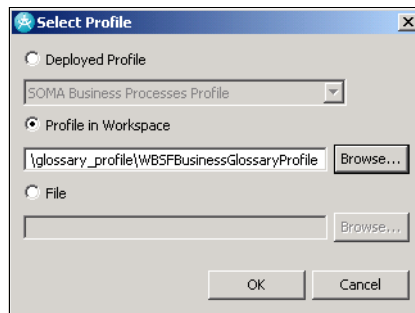


Figure 8-12 Selected profile

f. Click **OK**. If you are prompted with a dialog box with a warning message that tells you that the selected profile has not been released, click **OK**. The business glossary profile is applied, as shown in Figure 8-13.

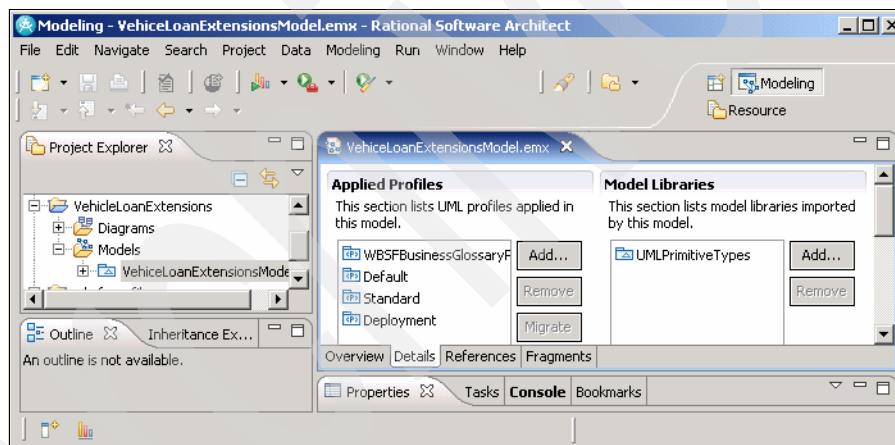


Figure 8-13 Applied Business Glossary profile

3. Create a new package named glossary (Figure 8-14 on page 173):
  - a. Right-click the model, and click **Add UML** → **package**.
  - b. Specify glossary for the package name, and press **Enter**.

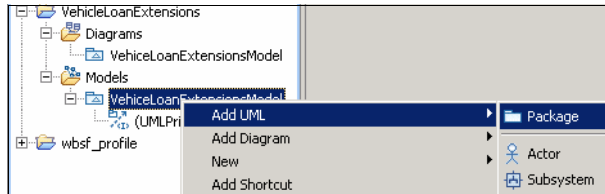


Figure 8-14 Create package

4. Open the Main diagram by expanding **Model** → **glossary** and double-clicking **Main**. Locate the Palette menu, which is on the right side, click **Class**, and drag it on to the diagram. Name this class **CustomerTypeEnumeration**. as shown in Figure 8-15.

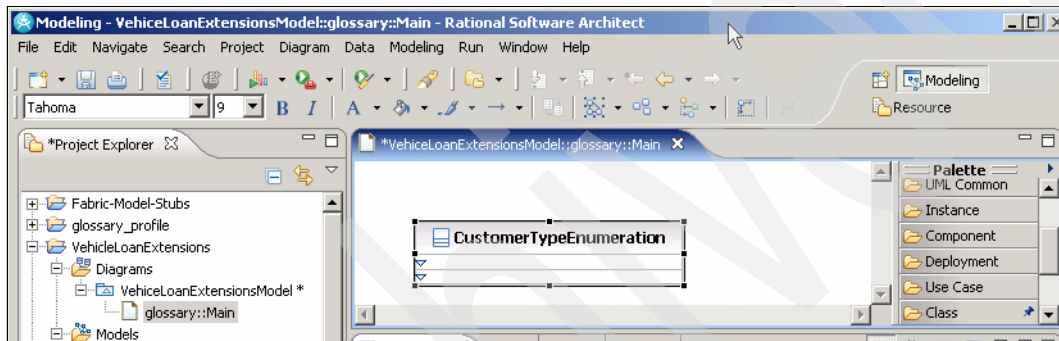


Figure 8-15 Create Class CustomerTypeEnumeration

5. While the **CustomerTypeEnumeration** class is selected, go to the Properties view, and click **Stereotypes** → **Apply Stereotypes**. When the Apply Stereotypes panel appears, select **Glossary Term**, as shown in Figure 8-16 on page 174. Click **OK**.

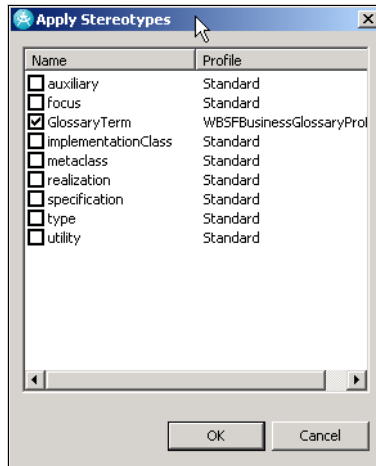


Figure 8-16 Apply Stereotypes

- Repeat steps 1 through 5 to create BankIdentifierEnumeration and RatingScoreEnumeration. See Figure 8-17.

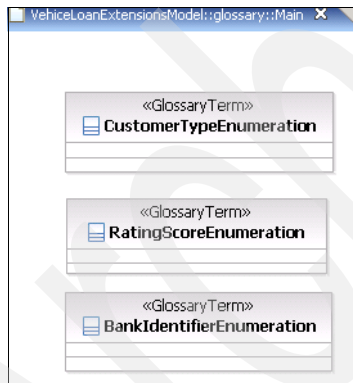


Figure 8-17 Enumerations

- Press **Ctrl+S** to save your work.

**Note:** Save your work regularly using the key combination Ctrl+S, or from the **File** menu, select **Save**.

### 8.7.3 Creating the ontology model

To create the ontology model:

1. Create a new UML modeling project called VehicleLoanOntology:
  - a. Click **File** → **New** → **Other** and click **UML Project**. Click **Next**.
  - b. In the Project Name field, type VehicleLoanOntology, as shown in Figure 8-18, and click **Next**.

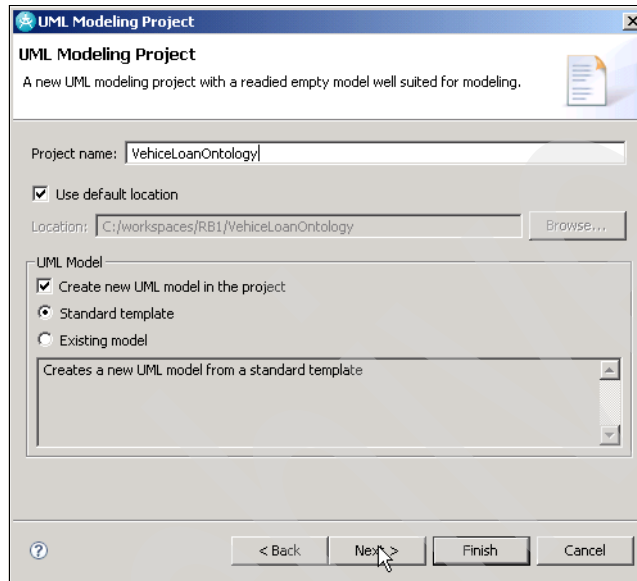


Figure 8-18 Create ontology project

2. In the File Name field, type VehicleLoanOntologyModel, and in the Templates section, click **Blank Model**. Clear the **Create a default diagram in the new model** option. Click **Finish**, as shown in Figure 8-19 on page 176.

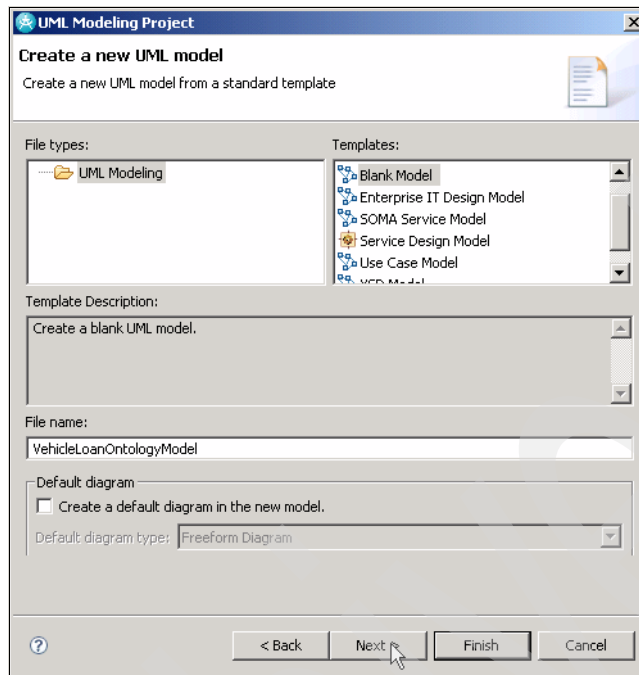


Figure 8-19 Create ontology model

3. Add WBSFOntologyProfile to the model:
  - a. Open the model by double clicking **VehicleLoanOntologyModel**.
  - b. Click the **Details** tab. In the Applied Profiles section, click **Add**, as shown in Figure 8-20 on page 177.
  - c. Click **Profile in Workspace**, and click **Browse**.

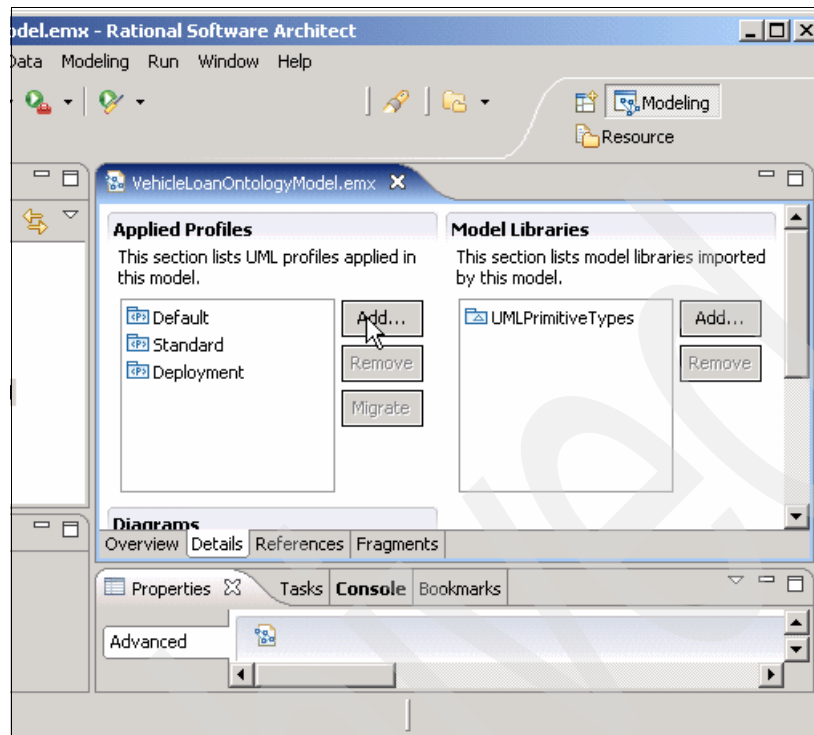


Figure 8-20 Add ontology profile

- d. Add the Business Glossary profile by expanding **wbsf\_profile** and double-clicking **WBSFOntologyProfile.epx**, as shown in Figure 8-21. Click **OK**. If a warning appears that says that the selected profile has not been released, click **OK**.

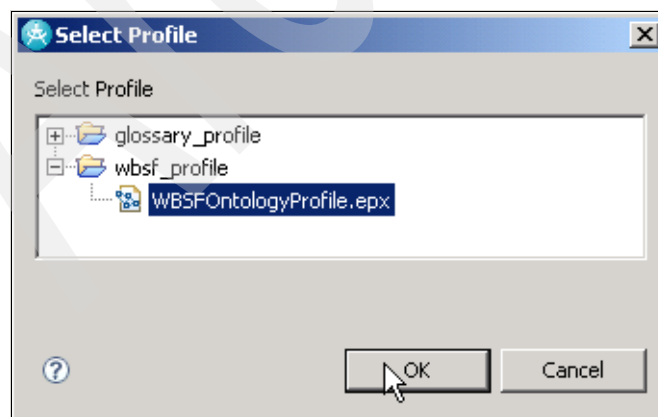


Figure 8-21 Select Ontology profile

4. In the Import Model Library section, click **Add**. Click **Deployed Library**, and select **XSDDataTypes** from the pull-down menu, as shown in Figure 8-22.

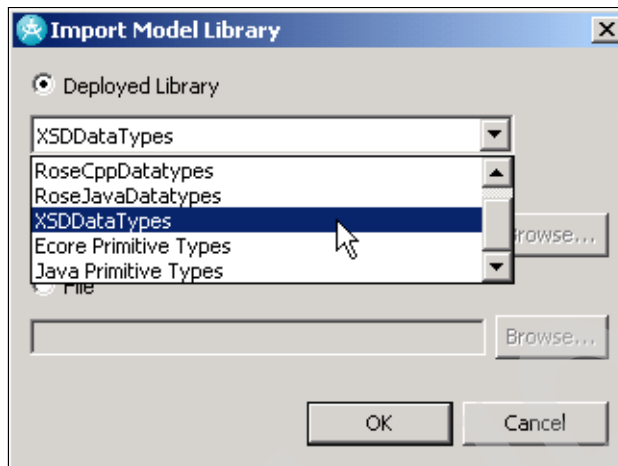


Figure 8-22 Select XSDDataTypes

5. Click **OK**, as shown in Figure 8-23.

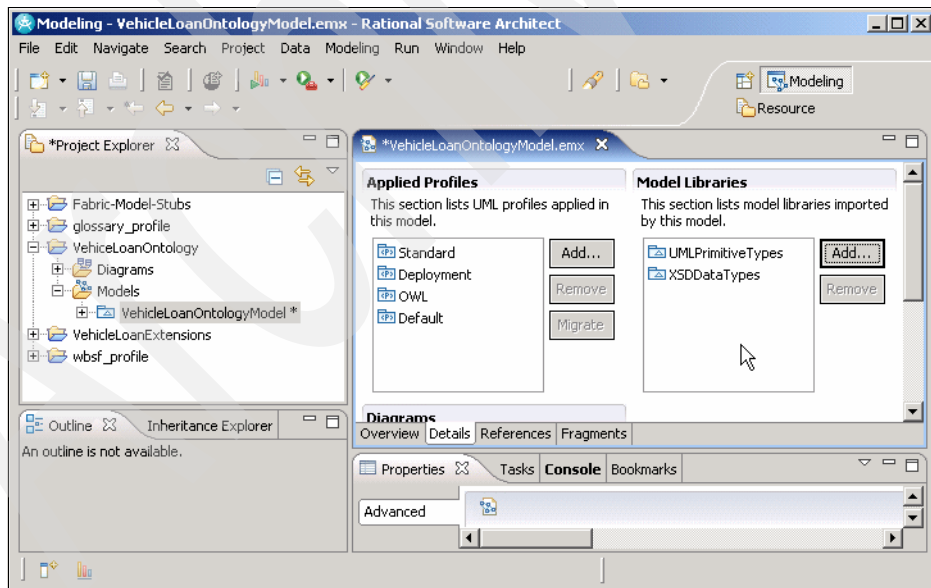


Figure 8-23 Imported Profile and Model Library



6. Create a new package named extensions:
  - a. Right-click the **VehicleLoanOntologyModel**, and click **Add UML → package**.
  - b. Specify extensions for the package name. Press **Enter**. (Figure 8-24).

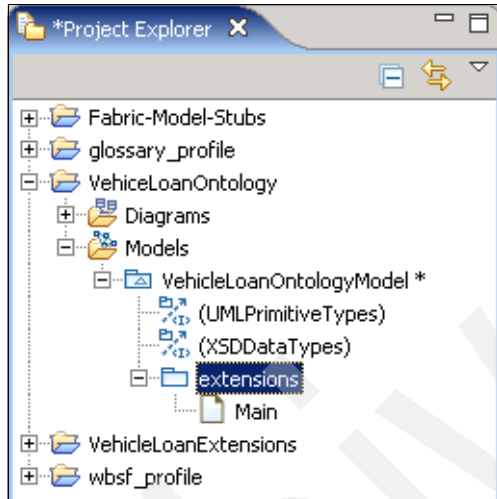


Figure 8-24 Create extensions package

7. Apply the fabricNamespace stereotype to this package:
  - a. Go to Properties view, click **Stereotypes → Apply Stereotypes**, and select **fabricNamespace**, as shown in Figure 8-25 on page 180. Click **OK**

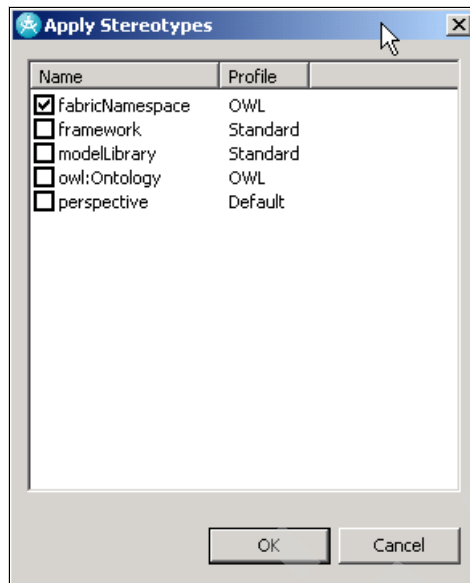


Figure 8-25 Apply Stereotypes

- b. Scroll down to the Stereotype Properties, in the Value field for directory, type vehicleloan-assertions, and in the Value field for uri, type <http://www.ibm.com/vehicleloan/extensions>, as shown in Figure 8-26.

Applied Stereotypes:			
Stereotype	Profile	Required	
FabricNamespace	OWL	False	

Stereotype Properties:	
Property	Value
<input type="checkbox"/> FabricNamespace	
directory	vehicleloan-assertions
javaPackage	
type	
uri	<a href="http://www.ibm.com/vehicleloan/extensions">http://www.ibm.com/vehicleloan/extensions</a>

Figure 8-26 Specify Stereotype Properties

## 8.7.4 Glossary transformation

Glossary transformation occurs to transform the glossary terms (for example, the enumeration created in the previous section) so that they can be used as OWL classes in the ontology model. Perform the following steps:

1. Run the glossary transformation:
  - a. From the menu, select **Modeling** → **Transform** → **New Configuration**.
  - b. Expand **Fabric Transformations**, and click **Glossary Transformation**.
  - c. In the Name field, type VehicleLoanGlossaryTransform, and from the Configuration file destination pull-down menu, select VehicleLoanOntology, as shown in Figure 8-27. Click **Next**.

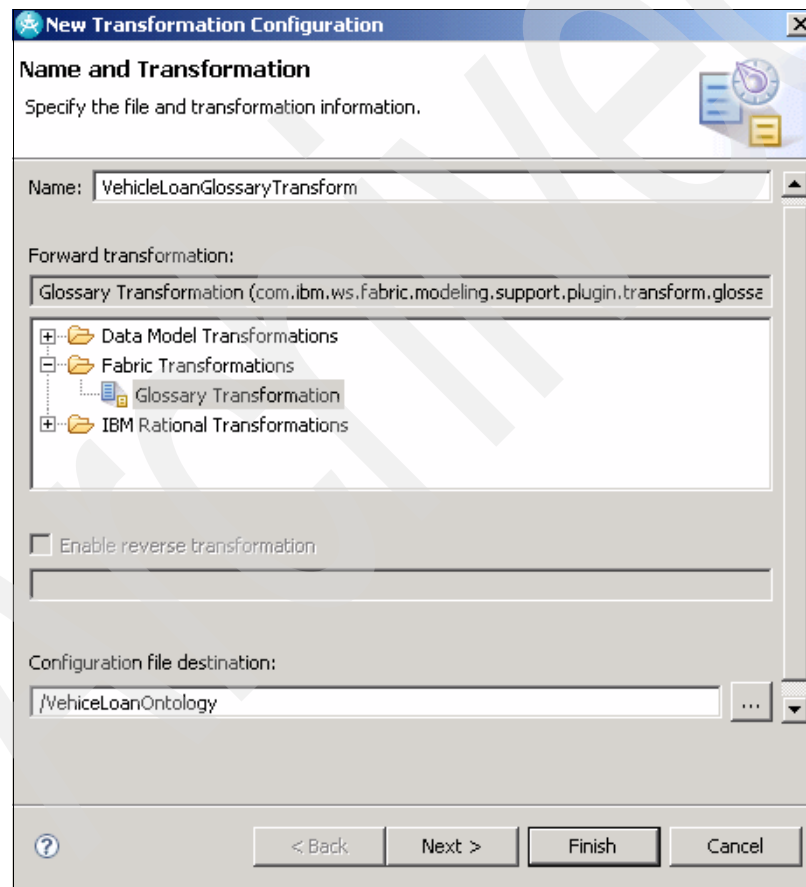


Figure 8-27 Glossary Transformation

- For the source, select **VehicleLoanExtensionsModel**, and for the target, select **VehicleLoanOntologyModel**, as shown in Figure 8-28. Click **Next**.

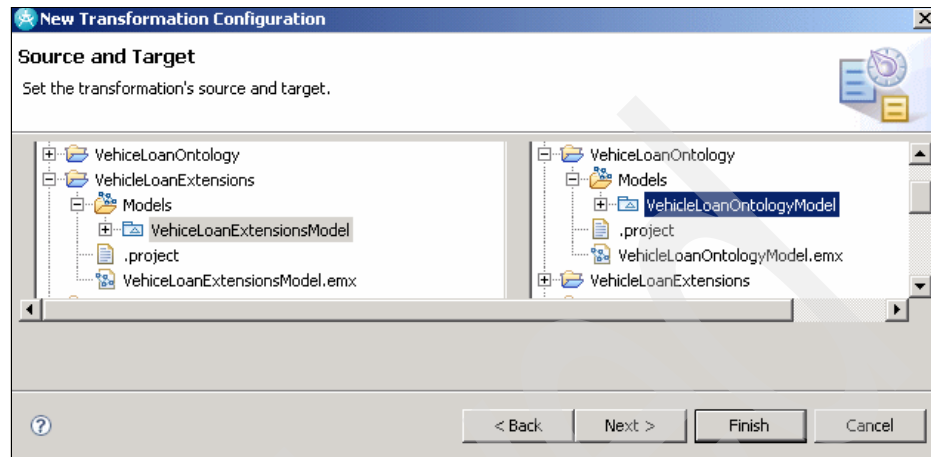


Figure 8-28 Select source and target

- For the value of Target Package, select **extensions**, as shown in Figure 8-29. Click **Finish**.

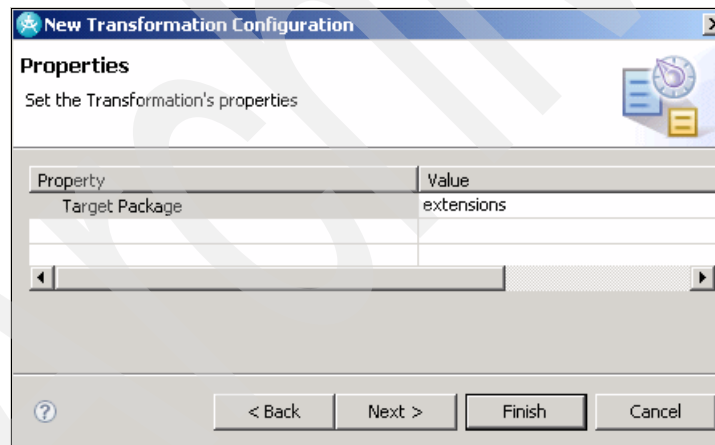


Figure 8-29 Select package

- Click the **Run Glossary Transformation** button, as shown in Figure 8-30 on page 183.

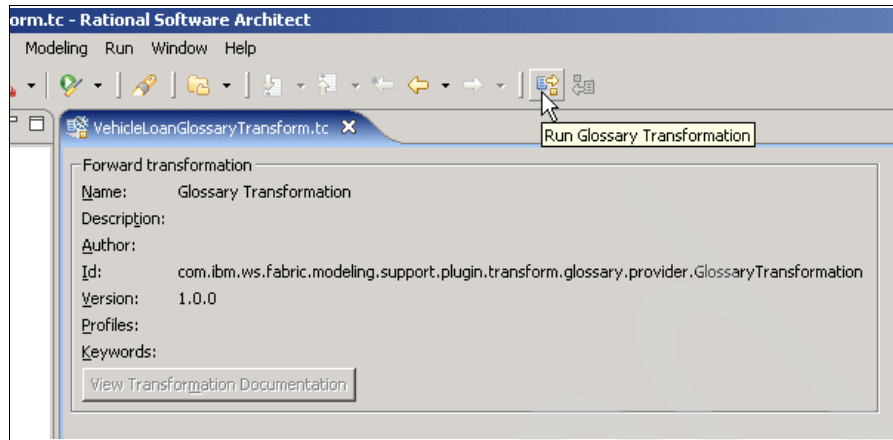


Figure 8-30 Run Transformation

After the transformation is successful, the OWL class definitions are created in the **VehicleLoanOntologyModel** → **extensions** package, which we show in Figure 8-31.

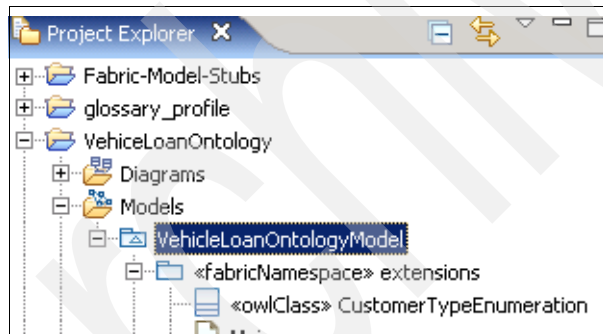


Figure 8-31 Generated OWL classes

**Note:** OWL is a Web Ontology Language. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes, cardinality (for example, "exactly one"), equality, richer typing of properties, characteristics of properties (for example, symmetry), and enumerated classes.

For more information see:

<http://www.w3.org/2004/OWL/>

## 8.7.5 Ontology modeling

Perform the following:

1. Click **VehicleLoanOntology** → **Model** → **VehicleLoanOntologyModel**. Expand the **extensions** package. Double-click the Main diagram. Click **extensions** → **<<owlClass>>CustomerTypeEnumeration**, and drag the **CustomerTypeEnumeration** OWL class from the **extensions** package and drop it onto the Main diagram, as shown in Figure 8-32.

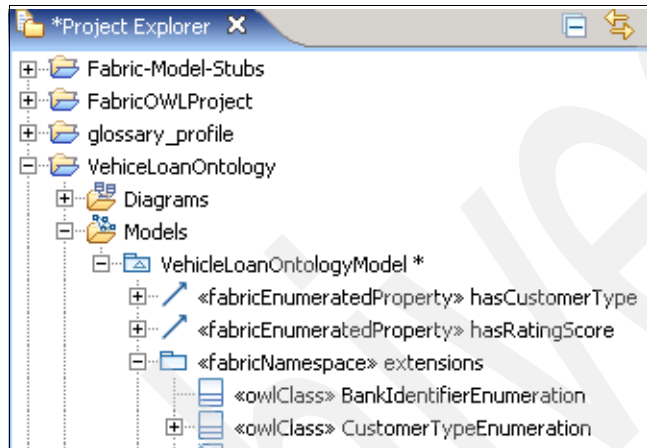


Figure 8-32 Extensions package view

2. Expand **Fabric-Model-Stubs** → **Models** → **Fabric-Core** → **Core Ontology**. Click the **Enumerated Value** class, and drag it to and drop it on the Main diagram.
3. From the Palette menu, click **Generalization**, and click **CustomerTypeEnumeration**. Hold the mouse, and click **EnumeratedValue**.

4. From the Palette menu, click **Class Instance**, as shown in Figure 8-33.

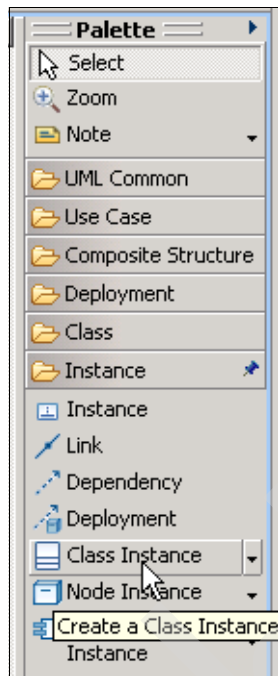


Figure 8-33 Create new class instance

5. Click the canvas, and click **Select Existing Element**, as shown in Figure 8-34.

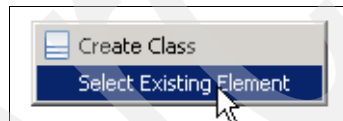


Figure 8-34 Select existing element

6. In the dialog, click **CustomerTypeEnumeration**. Go to Properties and change the name to `CustomerTypeEnumeration:Premium`. Type `Premium` for the Value, which creates an instance with value `Premium`. Similarly, create Instances for the `Regular` and `New` customer types, as shown in Figure 8-35.

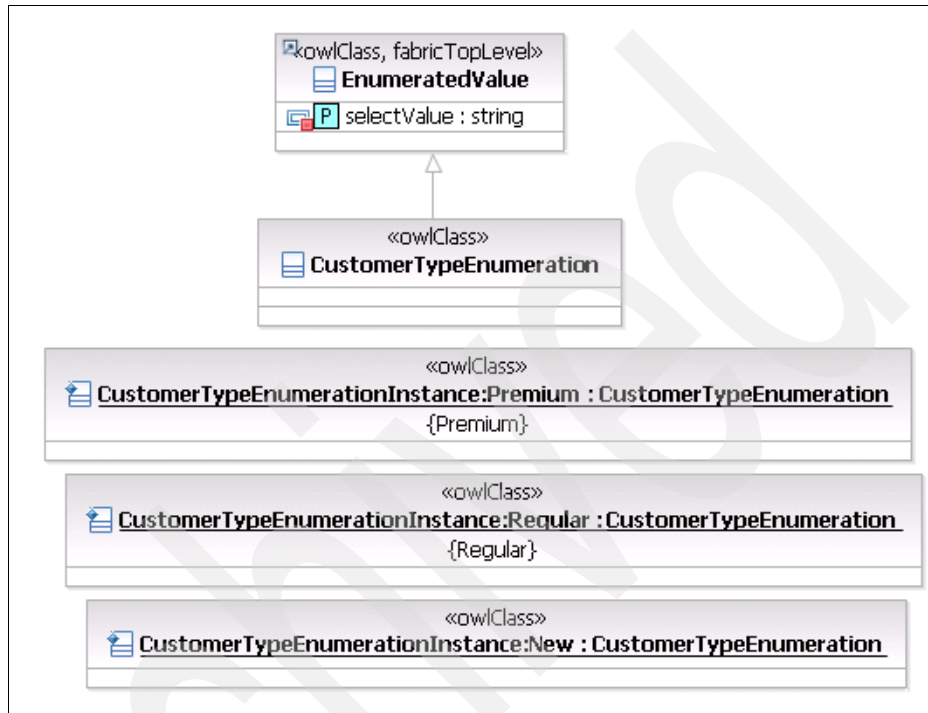


Figure 8-35 CustomerType enumeration

7. Create a package named `ontology`. Apply the **fabricNamespace** stereotype to this package. In the Stereotype Properties table, in the Directory field, type `vehicleloan-assertions`, and in the uri field, type `http://www.ibm.com/vehicleloan/assertions`.
8. Double-click the Main diagram for the **ontology** package. From the Palette menu, click **Class**, and create a class named `CustomerTypeAssertion`. Click the **Stereotypes** tab, and apply the stereotypes for this class by clicking the **fabricAssertion**, **owlClass**, and **rdfs:label** stereotypes. In the Label field, type `Customer Type Assertion`, as shown in Figure 8-36 on page 187.



Applied Stereotypes:			
Stereotype	Profile	Required	
fabricAssertion	OWL	False	
owlClass	OWL	False	
rdfs:label	OWL	False	
<input type="button" value="Apply Stereotypes..."/> <input type="button" value="Unapply Stereotypes"/>			

Figure 8-36 Applied Stereotypes for Assertion

- Go to the Properties section, and click **Advanced**. Click **AssertionDuplicateAllowed**, and click the Value **True**, as shown in Figure 8-37.

Properties	Tasks	Console	Bookmarks
General	<Class> «fabricAssertion, owlClass, rdfs:Class»		
Attributes	Property		
Operations	Value		
Stereotypes	<input type="checkbox"/> fabricAssertion		
Documentation	abstractType		
Constraints	assertionComparator		
Advanced	<b>assertionDuplicateAllowed</b>		
	assertionType		
	contentAssertionScope		

Figure 8-37 Properties view

- Expand the **Fabric-Model-Stubs** → **Models** → **Fabric-Core** → **Policy Assertion Ontology** project, and drag the **ContentBasedAssertion** OWL class from the Policy Assertion Ontology, and drop it onto the Main diagram. From the Palette menu, click **generalization**, and click **CustomerTypeAssertion**. While you hold the mouse, click **ContentBasedAssertion**.
- In the Properties section, click **attribute**. Add an attribute to the CustomerTypeAssertion class called hasCustomerType of type **xsd:string**.
- Click **Stereotypes**. Apply the following stereotypes to this attribute: **fabricAssertedProperty**, **owl:DatatypeProperty**, **owl:FunctionalProperty**, **rdfs:label**.

13. Scroll down to the Stereotype Properties section, and enter the following values:
  - assertionPropertyComparator: EqualsComparator
  - controlWidget: referenceSelect
  - label: CustomerType
14. Expand package **extensions**. Click and drag the **CustomerTypeEnumeration** OWL class from the extensions package onto the diagram. Click and create a **directed** association from the **CustomerTypeAssertion** towards **CustomerTypeEnumeration**. In the label field, type hasCustomerType, and apply the **fabricEnumeratedProperty** stereotype to it, as shown in Figure 8-38.

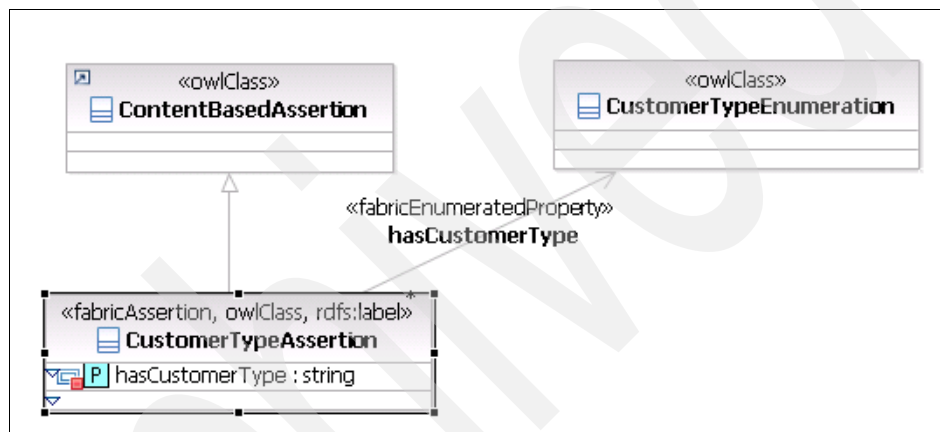


Figure 8-38 Create directed association

15. Repeat the above steps to create a new assertion called **LoanAmountAssertion** with the attribute **hasLoanAmount**, and create a generalization association from **LoanAmountAssertion** to **ContentBasedAssertion**, as shown in Figure 8-39 on page 189.

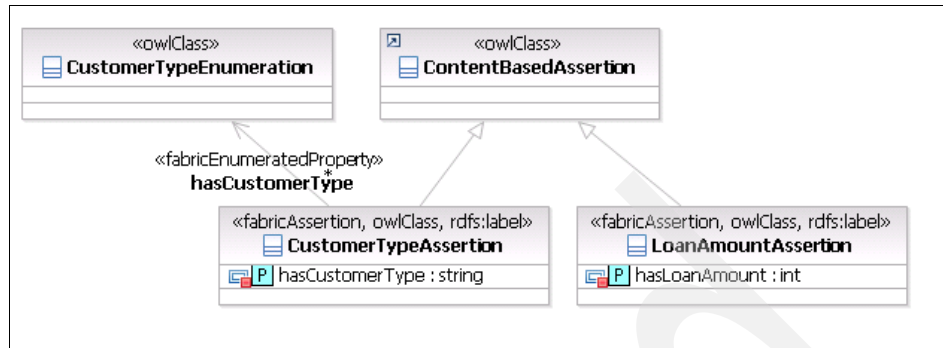


Figure 8-39 Create generalization association

16. Repeat the steps and create assertions BankIdentifierAssertion, RatingScoreAssertion, and CreditCheckCostAssertion. Figure 8-40 is the final diagram that shows all of the assertions.

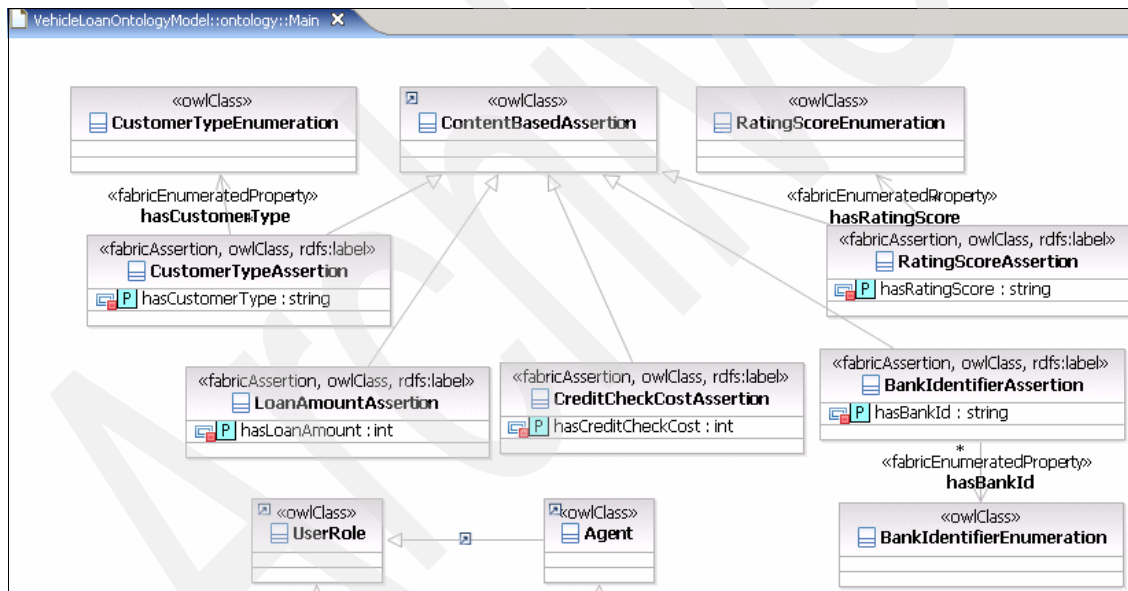


Figure 8-40 Modeled assertions

## 8.7.6 Ontology transformation

Ontology transformation is a method by which a UML model that is defined in one notation is converted into another UML model or a text.

To transform an ontology:

1. Click **Modeling** → **Transform** → **New Configuration**, as shown in Figure 8-41. In the dialog box, expand **Fabric Transformations**, and click **OWL Transformation**. In the Name field, type **VehicleLoanOntologyTransform**, and for the Configuration File Destination field, select **VehicleLoanOntology**, as shown in Figure 8-41. Click **Next**.

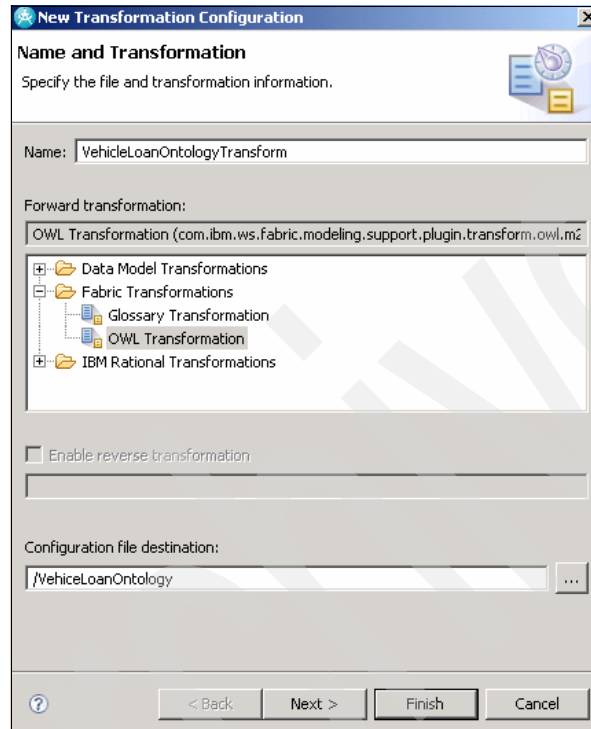


Figure 8-41 New Configuration

2. Click **Vehicle Loan Ontology Model** for the source and target, as shown in Figure 8-42 on page 191. Click **Finish**.

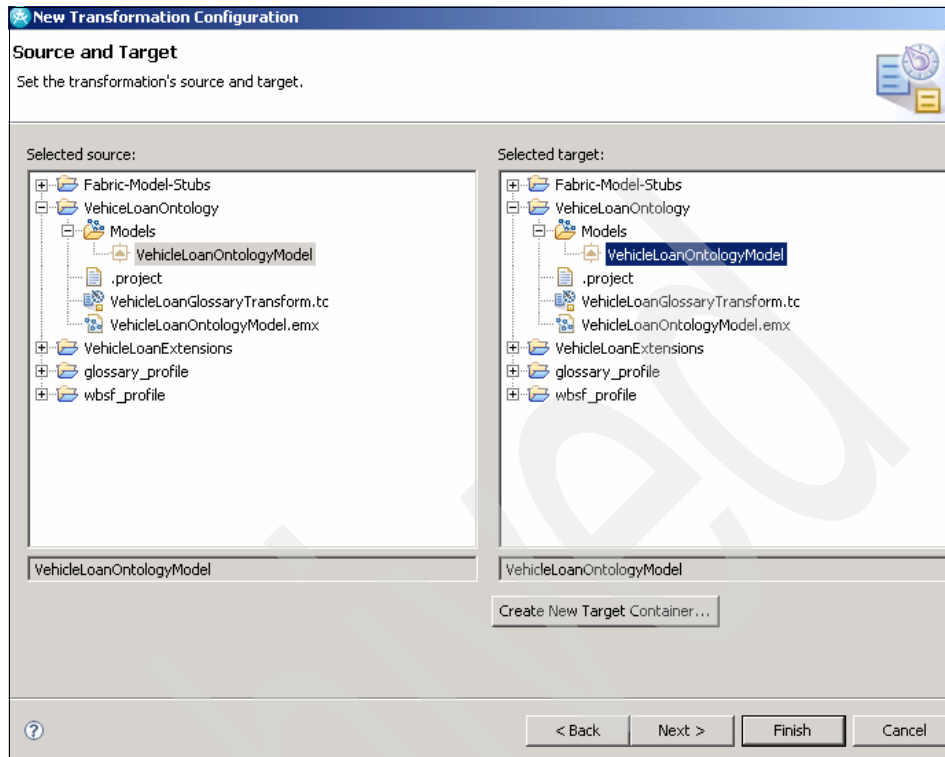


Figure 8-42 Select Source and Target

- Run the ontology transformation by going to the top of the tab and clicking the **Run OWL Transformation** symbol, as shown in Figure 8-43.

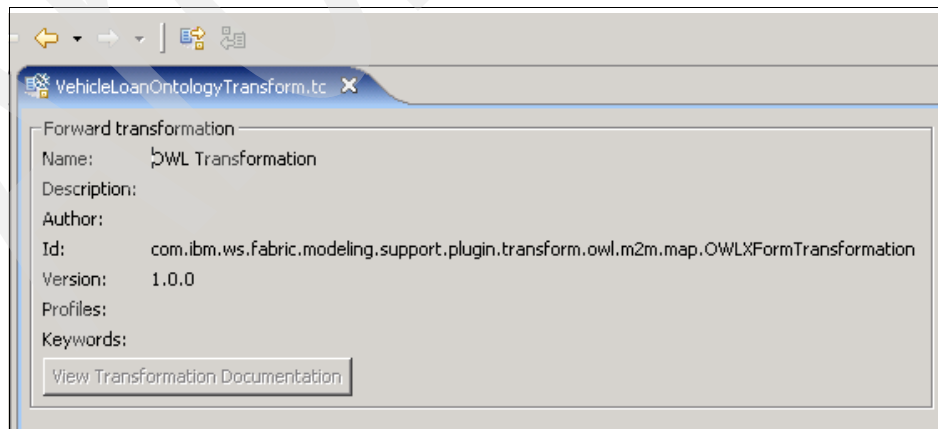


Figure 8-43 Run Ontology Transformation

## 8.7.7 Exporting the Fabric Content Archive

A Fabric Content Archive (FCA) is a compressed file or a JAR file that contains Web Ontology (OWL) files and a content-pack.xml file. The FCA provides the manifest of the Fabric Content Archive contents with definitions of projects and namespaces for the model extensions.

To export the Fabric Content Archive:

1. Expand **FabricOWLProject** → **vehicleloan-assertions**. Figure 8-44 shows the generated extensions.owl and ontology.owl files.

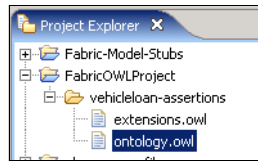


Figure 8-44 owl files for extensions and ontology

2. Export it as a .zip file:
  - a. Right-click **FabricOWLProject**, and click **Export**.
  - b. Click **General** → **Archive File**. Click **Next**.
  - c. Type the File Name as fabricitsobank.zip. Click **Finish**.
3. The generated .zip file fabricitsobank.zip contains ontology.owl and extensions.owl files.

Chapter 9, “Assembling Fabric solutions” on page 193 details the assembly steps for this Fabric solution. In Chapter 9, we describe the steps for importing the generated ontology extensions that we created in this section into Fabric.

# Assembling Fabric solutions

In Chapter 8, “Modeling Fabric solutions” on page 153, you modeled the Fabric solution. In this chapter, we look at how to Assemble Fabric solutions.

Assembling the Fabric solution consists of two parts:

- ▶ **Assembly in WebSphere Integration Developer**  
Designing the generalized process in WebSphere Integration Developer and using the Dynamic Assembler SCA component while you assemble your services.
- ▶ **Assembly in Composition Studio**  
Importing the SCA implementation in Composition Studio and creating the business context for your application.

## 9.1 Assembly in WebSphere Integration Developer

After you model the “AS-IS” business process, as described in 5.2.2, “AS-IS business scenario” on page 89, we need to create a business process executable from the model, which we can do by creating a Web Services Business Process Execution Language (WS-BPEL) process in WebSphere Integration Developer. If you are using the WebSphere Business Modeler for business modeling, you can generate WS-BPEL as an export that you can easily import in WebSphere Integration Developer.

Figure 9-1 on page 195 shows the loan process WS-BPEL that is generated in WebSphere Integration Developer by importing the WS-BPEL generated by WebSphere Business Modeler.





**Note:** You must cleanse the Business Process model (WS-BPEL), shown in Figure 9-1 on page 195, after you export the WS-BPEL from WebSphere Business Modeler. In this book, we assume that you already cleansed the WS-BPEL, normalization of the service interface for credit check, VIN lookup, rating, and loan provider service, using different available techniques.

Because this approach is static, it becomes difficult to manage your business process with this model when the number of services and process variability points increase. This offers reduced flexibility, and to incorporate changes in business processes requires the developer to make changes in the business process and redeploy it again, for example, adding a new category of Customer Type, requires another If-Else construct in the business process to invoke a particular endpoint that satisfies the requirement for the new category of Customer. So, this requires that we change the business process and redeploy it. Also without the proper governance process in place, it is difficult to identify the impact of the change to the overall business process.

## 9.2 Assembling a Fabric solution in WebSphere Integration Developer

Assembling a Fabric solution in WebSphere Integration Developer encompasses the following process:

1. Create a generalized WS-BPEL process.
2. Make Fabric context information available to the Dynamic Assembler.
3. Use the Dynamic Assembler SCA component.
4. Use Dynamic Assembler extensions.
5. Inject context into the WS-BPEL process.

### 9.2.1 Creating generalized WS-BPEL process

In this section, we provide steps to import the LoanProcess Project Interchange:

1. From the Additional Material section of this book, download the LoanProcessPI.zip. For information about how to obtain the additional material, refer to Appendix A, “Additional material” on page 503.
2. Start WebSphere Integration Developer.
3. Go to **File** → **Import**. In the Import dialog box, select node **Other** → **Project Interchange**. Click **Next**.

4. Browse to the location where the LoanProcessPI.zip is located, and select the zip file.
5. Click the **Select All** button, and click **Finish**, which builds the required ITSO, ITSOImpl, and WebSphereEnvUtil modules.
6. After the build is complete, open the Assembly Editor of the ITSO module by clicking the **Assembly Diagram**.
7. Review the Assembly Diagram, which is similar to Figure 9-1 on page 195.

Let us look at how Fabric solution helps to remove the limitations that we explained in 9.1, “Assembly in WebSphere Integration Developer” on page 194, and enhances the core BPM capabilities to create flexible SOA applications. We can streamline the model in Figure 9-1 on page 195 using the Fabric solution, as shown in Figure 9-2.

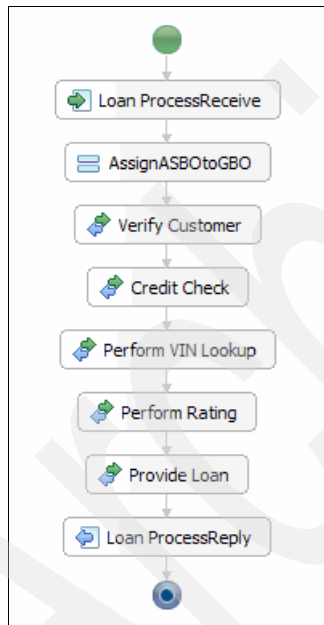


Figure 9-2 Simplified WS-BPEL using Fabric

The process gets simplified because all of the variabilities, such as If-Else constructs, are extracted out of the process and saved as business metadata in Fabric, which you can easily configure making the process more linear and abstract, as shown in Figure 9-2. Effecting changes in the business process becomes easier because changes are driven by changing configuration files (Fabric metadata) rather than by changing the business process at design time.

## 9.2.2 Making Fabric context information available to the Dynamic Assembler

Based on the incoming request, the Dynamic Assembler component of Fabric invokes the appropriate endpoint that needs to serve the request; however, to make the intelligent decision on which endpoint to be invoked, the Dynamic Assembler needs some extra information about the request. This extra information is called the *context*.

By definition, context is extra data about a request or response that gives Fabric's Dynamic Assembler information about the request or response so that the framework can make intelligent decisions about how to handle either one, for example, context might include things, such as:

- ▶ The data about the originator of the request (existing versus new customer, relative importance of that customer, and so on).
- ▶ How the framework received the request (through an HTTP Web service request versus JMS versus a Web page, and so on).
- ▶ The information in terms of which user is invoking the service (service entitlement of the user) for example, the subscription information of the user to access a business service.

Section 10.4, “Enrollments and subscriptions management” on page 297, explains how to generate the subscription information (subscription ID) using the Fabric Subscription Manager.

The Dynamic Assembler performs a policy match on the context to determine which policies to use in making the endpoint routing decision. Thus, the context is central to the operation of the Dynamic Assembler.

There are two approaches to make the Fabric context information available to the Dynamic Assembler:

- ▶ Modify WSDL to add Fabric context explicitly in the SOAP header.
- ▶ Inject the context using Fabric SDK APIs.

### **Modify WSDL to add Fabric context explicitly in the SOAP header**

This approach needs the change in the process WSDL to carry the context in the SOAP header to make it available to Dynamic Assembler. We are not going to follow this approach of passing the context information in this book.

## Injecting the context using Fabric SDK APIs

This approach involves creating a Fabric context and populating it with the required content-based information and with the subscription information. The subscription information (subscription ID) is necessary because it helps to attach all of the service invocation to a single context, which provides a single view of the performance matrix.

Example 9-1 shows how to use the Fabric SDK to get the subscription information of the user that is populated in the Fabric context in a SOAP header.

*Example 9-1 Using Subscriber Manager APIs to get the subscription ID*

---

```
/**
 * This method returns the subscription id for a business service by
 * providing the username of the user invoking the service
 * and the business service name that is being invoked for e.g in
 * our case its LoanProcessBS.
 */
public String getUserSubscriptionIDByService(String userName,
      String serviceName) {

    DataObject searchRequest = null ;
    String retValue = "NONE";
    try {
        DataObject searchResponse = null ;

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

        searchRequest = boFactory.create(

            "http://schemas.webifysolutions.com/wsf/2006/2/sdk/subscriber",
            "FindUserFromUserNameRequest");

        searchResponse = boFactory.create(

            "http://schemas.webifysolutions.com/wsf/2006/2/sdk/subscriber",
            "FindUserFromUserNameResponse");

        searchRequest.setString("userName", userName);

        //Invoking the method "findUserFromUserName" of Subscriber
        Manager Service
```

```

        searchResponse = (DataObject)
locateService_SubscriberManagerPartner().invoke("findUserFromUserName",
searchRequest);

        if(searchResponse != null){

            // Get the Subscriptions

            java.util.List subscriptionsList =
searchResponse.getDataObject("user").getList("subscriptions");

            for(int i=0; i < subscriptionsList.size(); i++){
                // Iterate the subscription list
                DataObject subscriptions =
(DataObject)subscriptionsList.get(i);
                //Get reference to the business service data object
                DataObject businessService =
subscriptions.getDataObject("enrollment").getDataObject("businessService");

                String businessServiceName =
(String)businessService.getString("name").trim();

                logger.info("business Service Name is " +
businessServiceName + " serviceName = " + serviceName + " searchIndex =
"+businessServiceName.indexOf(serviceName.trim()) );

                if(!(businessServiceName.indexOf(serviceName)== -1)){
                    // This is the service for which we need the
subscription ID
                    logger.info("Found the business service .. extracting
the ID for " + businessServiceName +" and user " + userName);
                    retValue = subscriptions.getString("id").trim();
                    continue ;
                }
                else {
                    logger.info("Compare results: " +
serviceName.compareTo(businessServiceName));
                }
            }

        }

    } else {

```

```

        System.out.println("ERROR: NO RESPONSE FROM THE SUBSCRIBER
MANAGER WEB SERVICE CALL ..") ;
    }

    }catch (Exception e) {
        e.printStackTrace();
    }

    logger.info("Returning subscription ID:" + retValue);

    return retValue;
}

```

---

We explain how to inject the context in a SOAP header in 9.2.5, “Injecting context into the WS-BPEL process” on page 214.

### 9.2.3 Using the Dynamic Assembler SCA component

In this section, we explain how to use the Dynamic Assembler SCA Component.

To use the Dynamic Assembler SCA component:

1. In IBM WebSphere Integration Developer 6.1, open the **Business Integration** perspective.
2. Open the Assembly Editor of the ITSO module by clicking the **Assembly Diagram**.
3. Select the Dynamic Assembler component from the palette, as shown in Figure 9-3 on page 202.

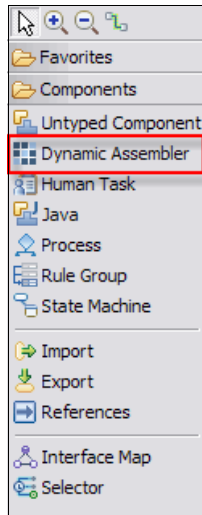


Figure 9-3 Selecting the Dynamic Assembler component

4. Drag the Dynamic Assembler component on to the assembly editor and label it **VerifyCustomerDA**, as shown in Figure 9-4.

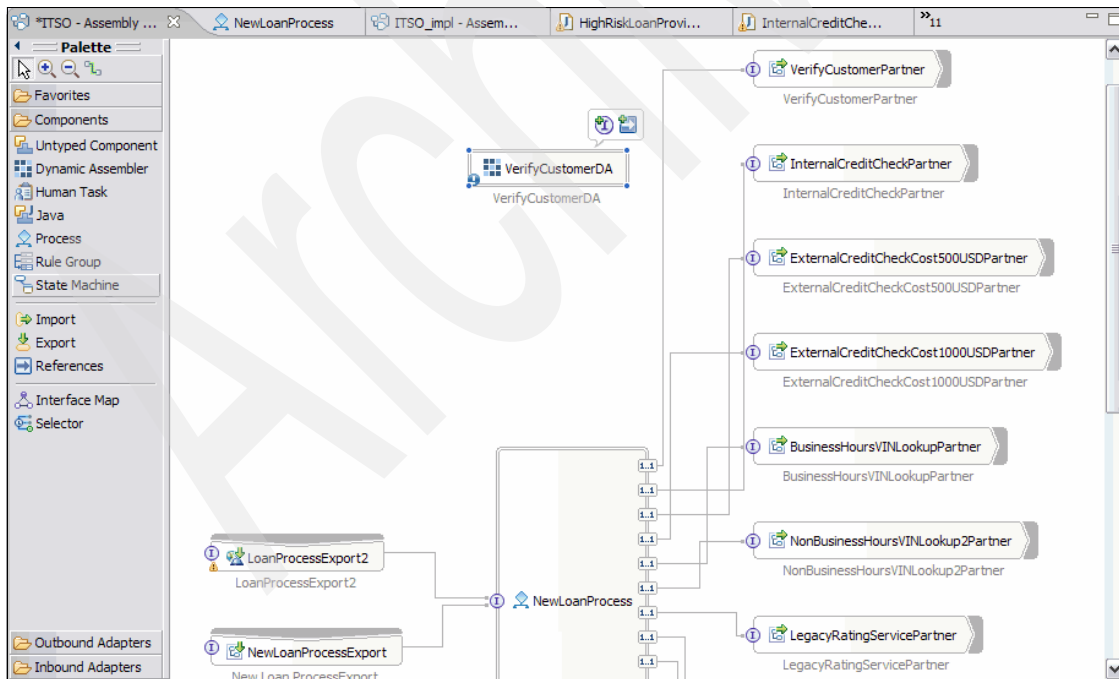


Figure 9-4 Adding Dynamic Assembler component



5. Right-click the **VerifyCustomerDA** component, and select **Add → Interface**. Select the **Verify Customer** interface, as shown in Figure 9-5.

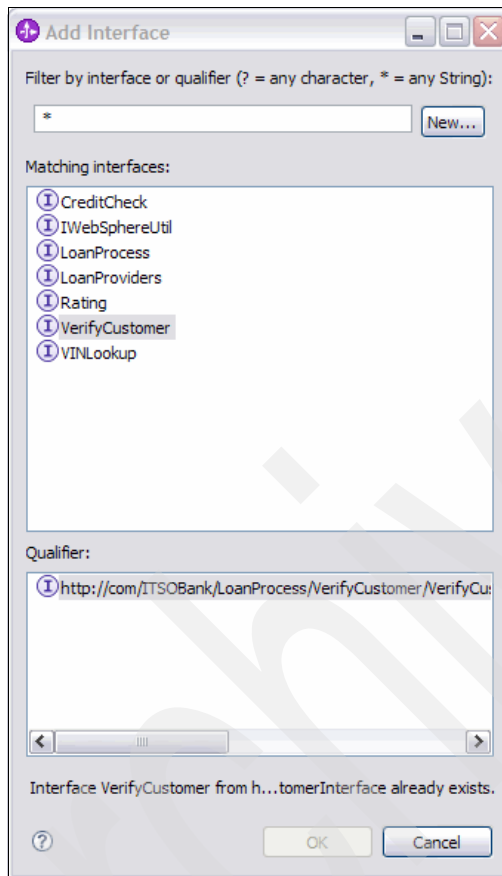


Figure 9-5 Adding interface to Dynamic Assembler component

6. Wire the VerifyCustomerDA to the VerifyCustomer reference partner of the LoanProcess WS-BPEL, as shown in Figure 9-6 on page 204.

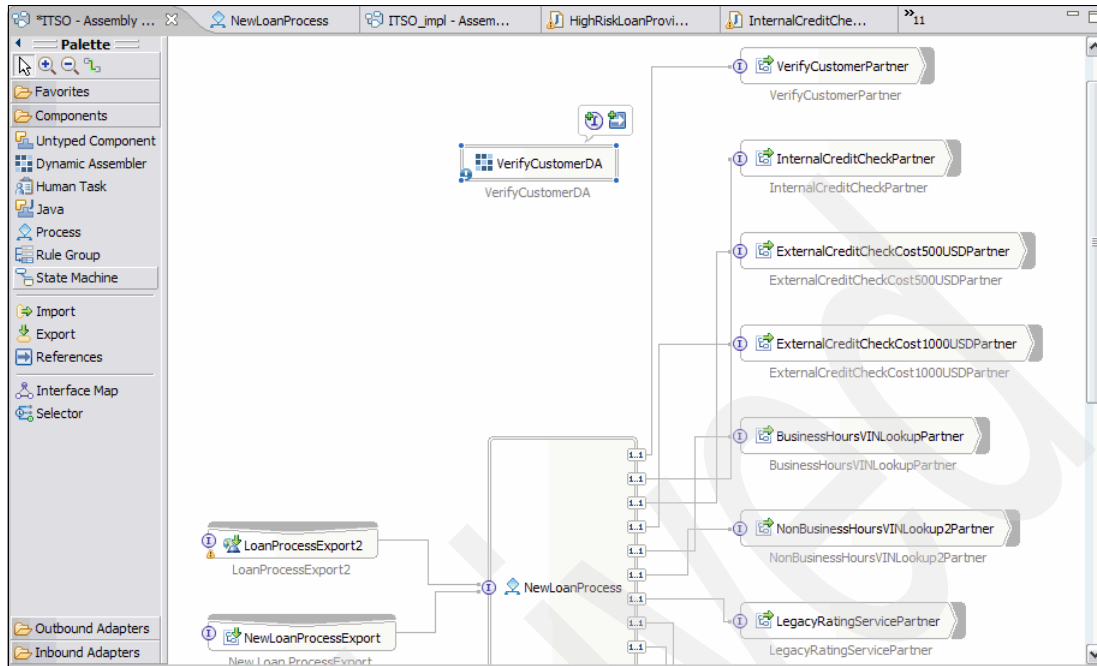


Figure 9-6 Linking the Dynamic Assembler component to WS-BPEL process

7. Double-click the **VerifyCustomerDA** to configure the Dynamic Assembler characteristics, as shown in Figure 9-7.

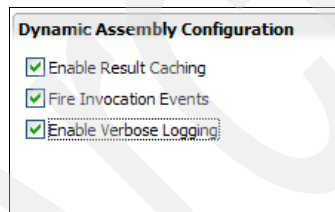


Figure 9-7 Dynamic Assembler configuration settings

There are three types of Dynamic Assembler configuration settings:

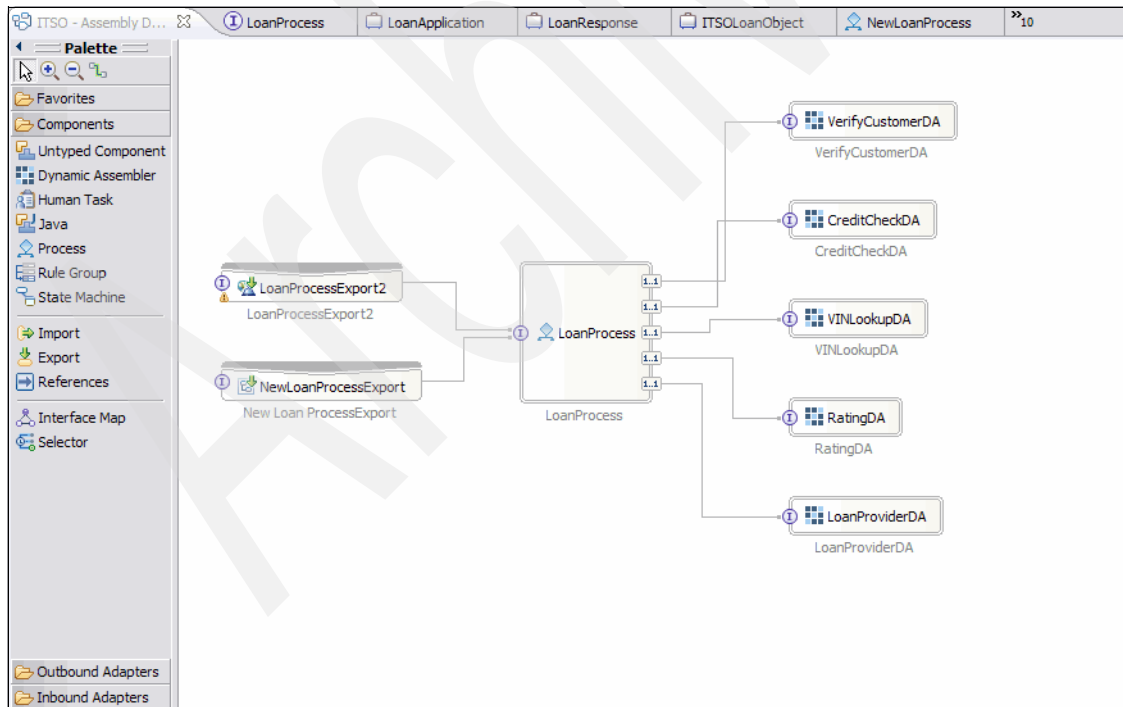
- ▶ **Enable Result Caching:** Caches the results so that Dynamic Assembler can use it in the policy computation. Clearing the Enable Result Caching option forces the Dynamic Assembler to recalculate the selected endpoint each time.
- ▶ **Fire Invocation Events:** When selected, the Dynamic Assembler fires the invocation events that the Dynamic Assembler's Event Listener plug-in can handle.

- Enable Verbose logging: When turned on for an individual Dynamic Assembler instance, will write in the server log file the activities that the Dynamic Assembler performed in the process of handling the request.
- 8. Repeat steps 2 to 7 for the Dynamic Assembler components in Table 9-1, and each Dynamic Assembler will implement the corresponding interfaces, as shown in the Table 9-1.

*Table 9-1 Dynamic Assembler components with its interfaces*

Dynamic Assembler component	Interface
CreditCheckDA	CreditCheck
VINLookupDA	VINLookup
RatingDA	Rating
LoanProviderDA	LoanProviders

- 9. On completing the creation of all the Dynamic Assembler components, the ITSO loan process assembly look similar to Figure 9-8.



*Figure 9-8 All Dynamic Assembler components linked to the ITSO Loan Process*

## 9.2.4 Using Dynamic Assembler extensions

The Dynamic Assembler defines the concept of plug-ins that run at certain defined times in the life cycle of a request and response pair. Each plug-in type is called with defined parameters under defined conditions. Some plug-ins are only informational (they are called when certain actions have taken place). Plug-ins that can actually affect a request and a response are called *operational plug-ins*. There are five types of plug-ins that are available to extend the behavior of the Dynamic Assembler:

- Context extractor

In many cases, the data that the Dynamic Assembler needs to make decisions is in the actual request or response message body. However, the Dynamic Assembler cannot operate directly upon the message body; instead, it can only use data that is in the context. Thus, there is a type of plug-in called a context extractor. These plug-ins are invoked early in the life cycle of a Dynamic Assembler request. Their job is to look through the request message body and insert (or update) data items in the context based on what is found there.

As an example, in our loan process we have the ontology extended to define the concept, Customer Type, with the possible values New, Regular, and Premium. A context extractor could be designed to look at the inbound request message body to determine who the user is and then look for that user in a customer database and appropriately insert the context with the customer type.

- Policy Guard

A Policy Guard plug-in can completely stop the Dynamic Assembler from continuing to locate and invoke an endpoint for a request. In other words, Policy Guards can cause a request to fail without actually invoking an endpoint. The reason that they are called Policy Guards is because they are invoked with, among other things, the original request and the composite policy (the result of merging all applicable policies into a single policy that is used to find a target endpoint to handle the request). They can look through the request and the composite policy and then decide whether or not to allow the request to continue.

As an example, there might be two policies that are applicable to a particular request. One policy says that the endpoint should respond in less than 100 ms, while the other policy says that the endpoint must support JMS. In your particular environment, you might know that the minimum response time for any JMS endpoint is 500 ms (for whatever reason). A Policy Guard could then cause any request to be rejected if both these policy assertions are present. Such a Policy Guard prevents any request from going through.

► **Candidate filter (Endpoint filter)**

A candidate filter plug-in is invoked after the Dynamic Assembler performs all of its policy matching and has given a policy matching score to all of the endpoints, but before an endpoint is selected. It is invoked with the entire list of all candidates that were considered.

For each such endpoint, the following items are also provided:

- The cost (a configuration item).
- The policy matching score (calculated).
- An integer tier, such as a numeric bucket into which this endpoint falls (set by the plug-in).
- Whether or not the endpoint was rejected (set by the plug-in).

The plug-in is then free to take the following actions:

- Reject an endpoint (with a string that describes the reason).
- Assign a tier for the endpoint.

If an endpoint is rejected, it is no longer considered as a potential handler for the request. Otherwise, if the plug-in assigns tiers, then all other scoring is ignored. Every endpoint in the same tier is considered equivalent for the purposes of finding an endpoint to handle the request, for example, Table 9-2 shows the resulting tiers after calling an endpoint filter plug-in.

*Table 9-2 Endpoint filter tiers*

Endpoint	Tier
A	2
B	3
C	2
D	4

Endpoints A and C are selected to handle the request, with the requests being routed in a round-robin fashion. If A nor C is available (assume they are both inoperative due to limitations on the hours of operation), then endpoint B is tried next. Endpoint D is chosen only if all of the other endpoints are unavailable.

► **Response Listener**

A response listener plug-in is an operational type of plug-in that can actually modify the response from a successful endpoint call.

This plug-in can perform two types of tasks:

- It can make modifications to the actual response message. The response message is a Service Data Object (SDO). SDO operations can add, modify, or remove data from the message.
- It can make updates to the parent context, which makes data available to other Service Component Architecture modules that are invoked after this one, in the course of processing an SCA assembly.

Suppose that you have an SCA assembly that includes a Business Process Execution Language (WS-BPEL) component that invokes three services in a sequence. The first of these services responds with information about the type of customer for the request. You could use a Response Listener to update the parent context with information about the customer type (similar to what we previously described). Then the customer type is available for endpoint selection for the other two services in the WS-BPEL.

► **Event Listener**

Event listener plug-ins are informational plug-ins that have two methods:

- One runs after response listeners when an endpoint is found and called.
- The other runs whenever an endpoint cannot be found.

Event listeners are not supposed to make any decisions or take any actions based upon what the endpoint returned. They mainly notify other frameworks of invocations (for example, you might want to notify a management framework that an endpoint was invoked).

For the ITSOBank Loan Processing Scenario, we use the context extractor extensions to populate the context that the Dynamic Assembler requires.

Context extractors are implemented as a Java component that implements the ContextExtractor interface, which is available in the DA plug-in APIs. You must modify the build path of the project to add these DA plug-in JARs:

1. Right-click the **ITSO** project. Select **Properties** → **Java build path**. Click the **Libraries** tab. Select the **WebSphere Process Server v6.1** server runtime library. Click the **Remove** button.
2. Click **Add Library** → **Server Runtime**. Click **Next**.
3. Select **WebSphere Business Services Fabric Server V6.1** runtime from the list of libraries. Click **Finish**.
4. In the **Libraries** tab, expand the **WebSphere Business Services Fabric Server v6.1** profile. Verify the three JARs, as shown in Figure 9-9 on page 209. Click **OK** to confirm.

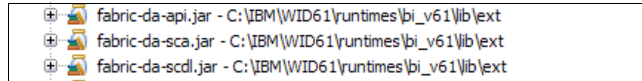


Figure 9-9 Fabric plug-in jars

The following steps get you to the context extractor, the Dynamic Assembler components that we created in 9.2.3, “Using the Dynamic Assembler SCA component” on page 201.

1. From the palette, select the **Java** component, and drag it on the assembly editor. Label it **CreditCheckContextExtractor**, as shown in Figure 9-10.

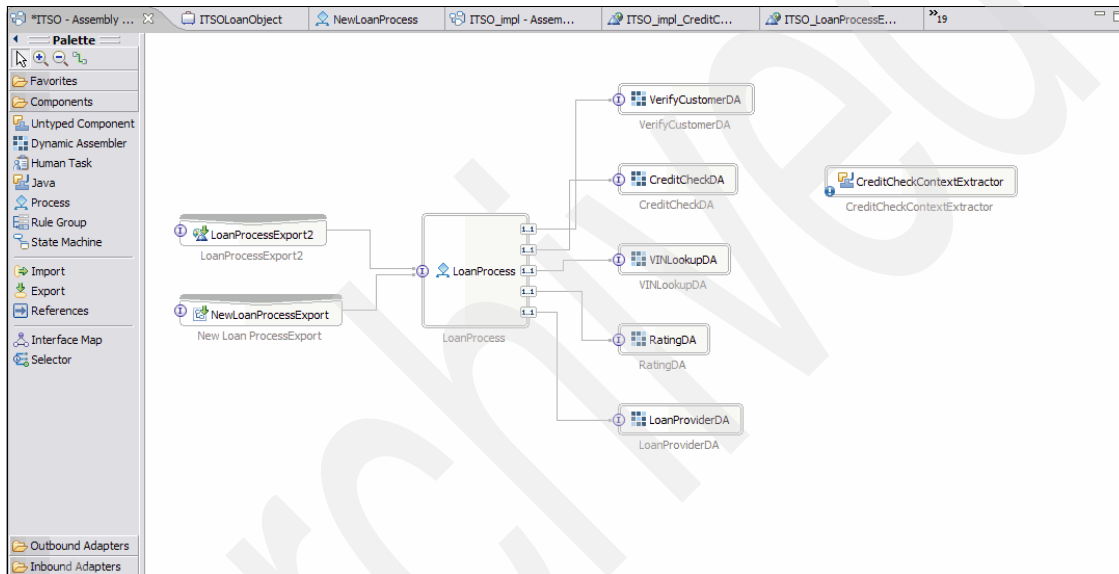


Figure 9-10 Adding the context extractor component

2. Right-click the component, and select **Add** → **Interface**. Select the **ContextExtractor** interface, as shown in Figure 9-11 on page 210.

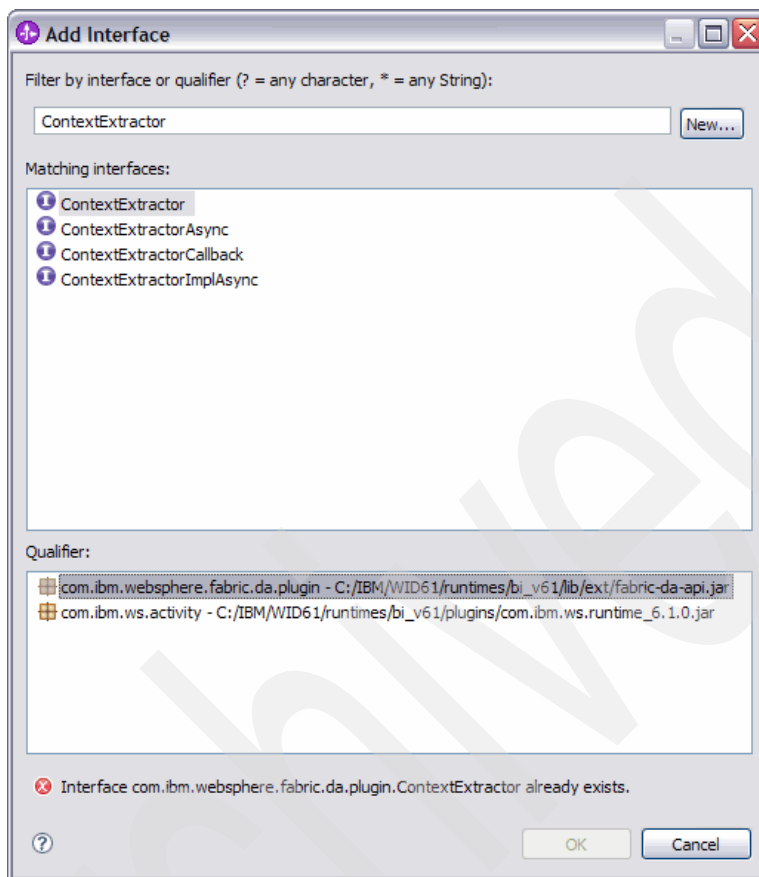


Figure 9-11 Adding ContextExtractor interface

3. Wire the **CreditCheckContextExtractor** Java component with the **CreditCheckDA** component, as shown in Figure 9-12 on page 211.



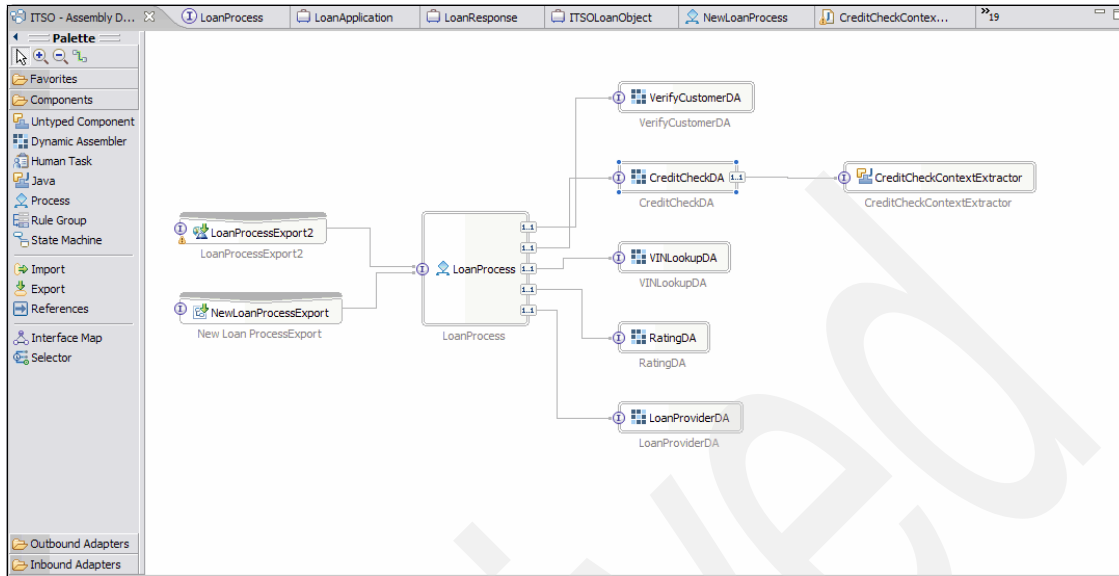


Figure 9-12 Wiring the ContextExtractor component with the Dynamic Assembler component

4. Double-click the **CreditCheckContextExtractor** component for implementation. Set the Customer Type and Loan Amount Assertion from the LoanApplication business object, which is part of the request object. The code is shown in Example 9-2, Example 9-3 on page 212, and Example 9-4 on page 212.

*Example 9-2 Required imports for the CreditCheckContextExtractor component*

---

```
import java.util.logging.Logger;

import com.ibm.websphere.fabric.da.PendingRequest;
import com.ibm.websphere.fabric.da.context.Context;
import com.ibm.websphere.fabric.da.plugin.ContextExtractor;
import
com.ibm.websphere.fabric.da.plugin.UnexpectedContentException;
import com.ibm.websphere.fabric.da.types.TypedValue;
import com.ibm.websphere.sca.ServiceManager;
import com.ibm.ws.fabric.da.sca.util.SdoUtil;

import commonj.sdo.DataObject;
import commonj.sdo.Sequence;
```

---

*Example 9-3 Required constant variables for the CreditCheckContextExtractor component*

---

```
/**
 * Logger instance for logging the messages.
 */
private final Logger logger = Logger

.getLogger(CreditCheckContextExtractorImpl1.class.getName());

/**
 * URI for customer type assertion.
 */
private static final String CUSTOMER_TYPE_ASSERTION =
"http://itsobank.com/schema#hasCustomerType";

/**
 * URI for loan amount assertion.
 */
private static final String LOAN_AMOUNT_ASSERTION =
"http://itsobank.com/schema#hasLoanAmount";
```

---

Example 9-4 is the implementation of the `extractContext` method of the `CreditCheckContextExtractor` component.

*Example 9-4 Using Context Extractor to populate the context with required information*

---

```
public Context extractContext(PendingRequest pendingRequest)
    throws UnexpectedContentException {

    Context context = pendingRequest.getContext();

    logger.info("ContextExtractorImpl messagebody "
        + SdoUtil.printTree(pendingRequest.getMessageBody()));

    Sequence sequence =
pendingRequest.getMessageBody().getSequence(0);
    DataObject requestDataObject = (DataObject)
sequence.getValue(0);

    logger.info("requestDataObject "
        + SdoUtil.printTree(requestDataObject));

    sequence = requestDataObject.getSequence(0);
```

```

        DataObject itsoLoanRequest = (DataObject)
sequence.getValue(0);

        logger.info("ITS0LoanObject " +
SdoUtil.printTree(itsoLoanRequest));

        //extracting the loan amount from the request and inserting in
the context.

        float loanAmount = 0.0f;
        String strLoanAmount = null;

        try {
            DataObject loanApplication =
itsoLoanRequest.getDataObject("LoanApplication");
            loanAmount =
loanApplication.getFloat("LoanAmountRequested");
            strLoanAmount = String.valueOf(loanAmount);
        } catch (NullPointerException e) {
            strLoanAmount = "";
        }

        TypedValue loanAmountAssertionValue = new
TypedValue(strLoanAmount);
        context.setSelectionProperty(LOAN_AMOUNT_ASSERTION,
loanAmountAssertionValue);

        logger.info("Loan amount assertion set to context.");

        //extracting the customer type from the request and inserting
in the context.

        String customerType = null;

        try {
            customerType = itsoLoanRequest.getString("CustomerType");
        } catch (NullPointerException e) {
            customerType = "";
        }

        TypedValue customerTypeAssertionValue = new
TypedValue(customerType);
        context.setSelectionProperty(CUSTOMER_TYPE_ASSERTION,
customerTypeAssertionValue);

```

```
        logger.info("Customer type assertion set to context.");

        return context;
    }
}
```

---

5. Repeat steps 1 to 4 to add the following context extractor components in the LoanProcess assembly diagram:
  - VerifyCustomer Context Extractor
  - Rating ContextExtractor
  - LoanProvider Context Extractor

**Note:** The implementation that is specific to each mentioned context extractor is provided as part of the src.zip file that is available in the Additional Materials section of this book. Refer to Appendix A, “Additional material” on page 503.

As a reader, you can review and use the available context extractor implementations for the remaining Verify Customer, Rating, and LoanProvider components.

### 9.2.5 Injecting context into the WS-BPEL process

Fabric provides the facility to track service performance metrics using IBM Business Services Performance Manager. But if we pass the request directly to the WS-BPEL, separate context identifiers are generated for each of the service invocations that are wired through Dynamic Assembler. So that all of the service invocations are a part of a single transaction, we need to provide the initial context to the Dynamic Assembler by defining a Java component called Context Injector.

To create a Context Injector component:

1. From the palette, select a **Java** component and drag it on the assembly editor. Label it LoanProcessContextInjector, as shown in Figure 9-13 on page 215.

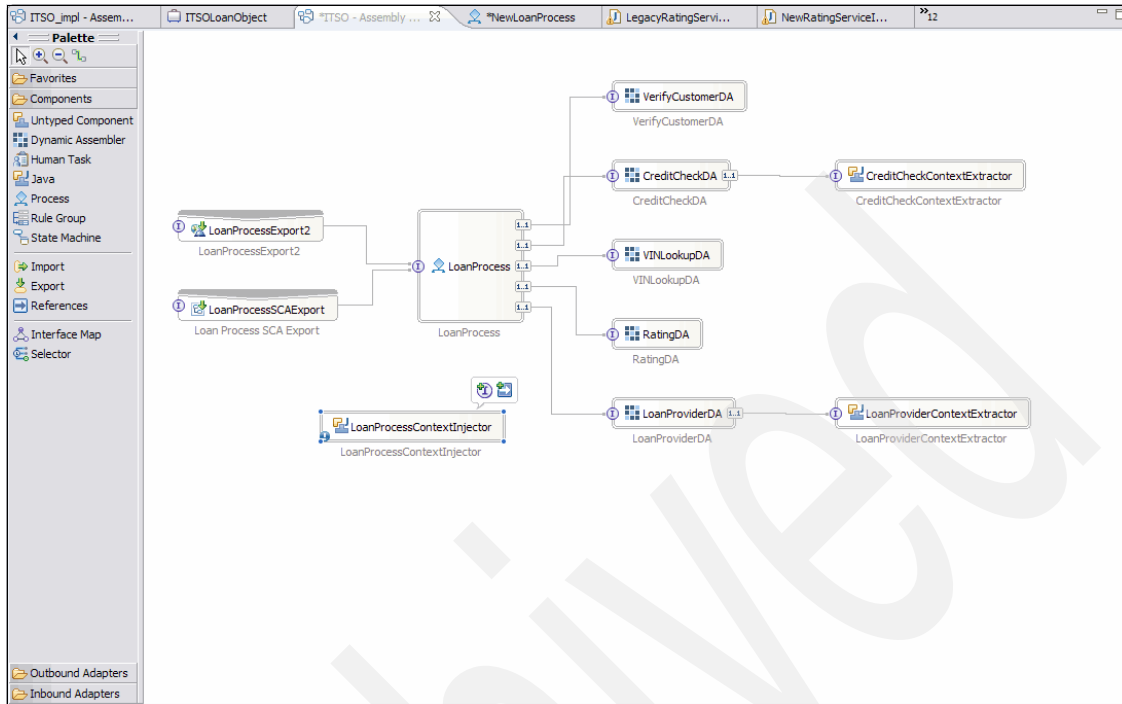


Figure 9-13 Adding the Context Injector component

2. Right-click the component, and select **Add** → **Interface**. Select the **LoanProcess** interface, as shown in Figure 9-14 on page 216.

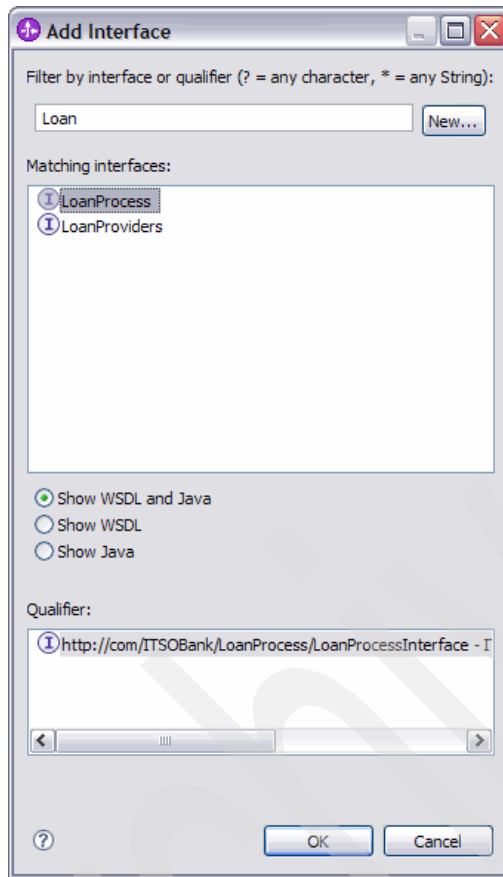


Figure 9-14 Adding interface to Context Injector component

3. Wire the LoanProcessContextInjector Java component with the LoanProcess WS-BPEL component, as shown in Figure 9-15.

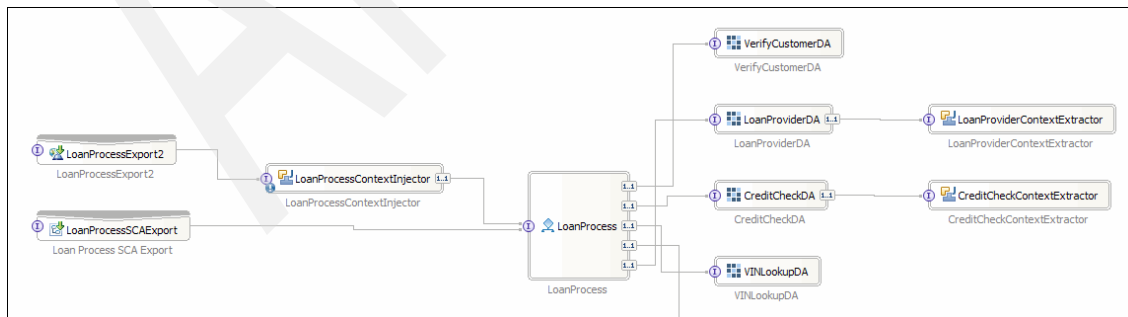


Figure 9-15 Wire Context Injector component to Loan Process

4. Double-click the **LoanProcessContextInjector** Java component to open the implementation.

Example 9-5 shows how to inject the context in the business process. The Dynamic Assembler uses this context in the process to invoke the appropriate endpoints.

*Example 9-5 Injecting the context information*

---

```
private class ContextInjectInvoker extends InContextServiceInvoker {

    DataObject requestGlobal = null;

    ContextInjectInvoker(DataObject request, DataObject
contextHeader) {

        super(request, contextHeader);

        //Set the environment variable
        String envParamStr = (String)
locateService_IWebSphereUtilPartner().getWebSphereEnvVar("FABRIC_ENV
IRONMENT_PARAM");

        logger.info("FABRIC_ENVIRONMENT_PARAM : " + envParamStr);

        setSelectionProperty("http://www.webifysolutions.com/2005/10/catalog
/governance#Environment", envParamStr);
        //Set environment variable ends here.

        // Get the user who is invoking this service from the
service header

        DataObject userDetails =
(DataObject)locateService_IWebSphereUtilPartner().getUserDetails("HE
ADER");

        logger.info("GOT USER DETAILS....");

        String executingAs = userDetails.getString("userName");

        logger.info("executing as user = " + executingAs);
```

```

        // One should know which Business Service does this
        component implement
        BUSINESS_SERVICE_NAME = "LoanProcessBS" ;

        if(!(executingAs.trim().equals("NONE")) &&
        !(executingAs.trim().equals("UNAUTHENTICATED"))){
            // User is found ..

            String subscriptionID =
            locateService_IWebSphereUtilPartner().getUserSubscriptionIDByService
            (executingAs, BUSINESS_SERVICE_NAME);

            logger.info("obtained subscription ID as = " +
            subscriptionID);

            setSelectionProperty(

            "http://www.webifysolutions.com/context/subscription",
            subscriptionID);
        }

        //adding the bank id in the context
        String bankId = request.getString("BankId");
        setSelectionProperty(BANKID_ASSERTION, bankId);
        logger.info("In ContextInjector -> Setting bankId in
        context.");
    }
}

```

---

**Note:** The code snippet in Example 9-5 on page 217 is sample implementation. The actual LoanProcessContextInjector implementation is provided as part of the src.zip, which is supplied as additional material with this book. Refer to Appendix A, “Additional material” on page 503.

The LoanProcessContextInjector implementation uses the WebSphereEnvUtil module, which contains the implementation of the Fabric Subscriber Manager APIs. The WebSphereEnvUtil module contains the WebSphereUtilComp Java component that connects to Fabric’s SubscriberManager service to retrieve the subscription information based on the username and business service.



Fabric's SubscriberManager service interface (WSDL), which is available as part of the fabric-catalog-api module, binds to the SubscriberManager service that is deployed and running in the Fabric runtime.

**Note:** In case you are implementing the LoanProcessContextInjector component, make sure that WebSphereEnvUtil and WebSphereEnvLIB are imported into your workspace.

This subscription information (subscriber ID) is passed in the context. The subscription information provided allows the Dynamic Assembler to extract the related information, such as user, channel, and role, that is used for service invocation. This is required for capturing the performance metrics using Business Services Performance Manager.

5. Create the Web service Export and SCA Export binding from the **LoanProcess** WS-BPEL component and wire them to the **LoanProcessContextInjector** component. Figure 9-16 shows the completed LoanProcess assembly diagram with the Dynamic Assembler components, Context Extractor components, and Context Injector components.

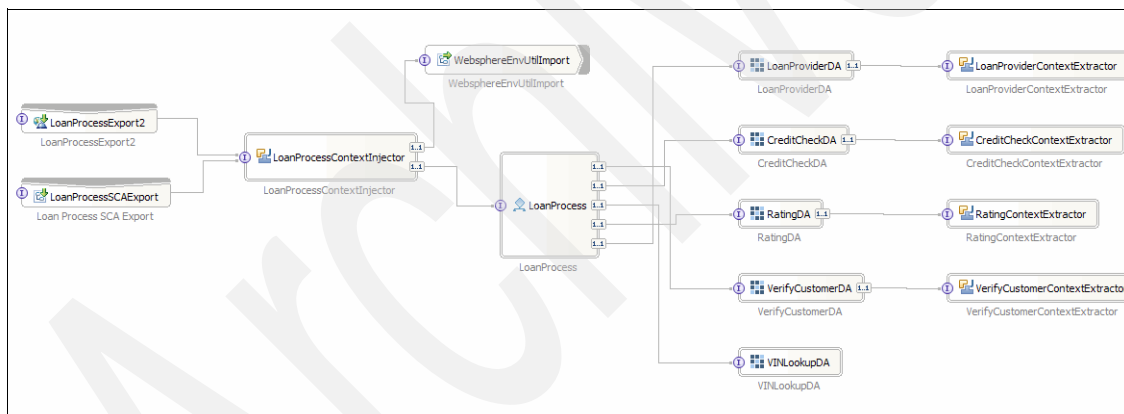


Figure 9-16 Loan Process wired with Context Extractor and Context Injector components

## 9.3 Assembling in Composition Studio

You created the technical service implementation of your business process in WebSphere Integration Developer. The next thing you need to do is provide business context to your services in Composition Studio. Assembling in Composition Studio consist of creating the business service metadata for your composite business application (CBA).

Here are the high-level steps for assembling the CBA in Composition Studio. Later in this section, we explain the tasks you need to follow to complete each step:

1. Create the subscriber model for your CBA project using the Fabric Administration Console and Subscriber Manager. The subscriber model consist of an organization and users.
2. Synchronize WebSphere Service Registry and Repository (WSRR) with Fabric. In this step you create a WSRR project in the Fabric Administration Console and Governance Manager to source technical service metadata from WSRR.

**Note:** Before you start developing the CBA, you first identify the services assets that can be reused, create new services or provide wrapper services to existing assets. You then import your technical service artifacts (WSDL, XSDs) into WSRR so that your technical assets can be reused across your enterprise. This step is generally carried out as part of your service identification phase, which is part of your SOA implementation.

3. If you extended the ontology to provide context information for your process, you will import the FCA project in Fabric using the Fabric Administration Console and Governance Manager. This loads all metadata definitions for your project in Business Service Repository. After this metadata is available in the Business Service Repository, other developers can use the metadata to understand and reuse solution components and refine solution behavior while assembling the CBA. This metadata can also be used by the Business Services Dynamic Assembler to make routing decisions to address the points of variability in your process and to select the right endpoint implementation at runtime.
4. Create the business service project for your CBA project, and reuse the technical service metadata from WSRR, which we created in Step 1, and context metadata, which we created in Step 2) by importing the respective project namespaces.
5. Replicate the business service project in Composition Studio and source composite service metadata for your business process (SCA module containing your business process) that we created earlier in WebSphere Integration Developer.
6. Create business context for your technical services, create business policies for your points for variability, and simulate your business polices to ensure that the right endpoints are being picked. The next step is to publish the metadata in to Business Service Repository, which you do as part of the Deploy phase.

## 9.3.1 Creating the Subscriber Model

Creating the Subscriber Model consists of creating the Organization model. After you create the Organization, you source users from the external repository and and associate them with organizations through roles.

### Creating an organization

Organizations model a business entity and the people that work within it. Organizations are managed using the Subscriber Manager in the Fabric Administrative Console and are visible in the Subscriber Explorer in Composition Studio. Supplying an organization in the context allows policies to contextualize behavior for certain organizations.

To create an organization with Fabric:

1. In an internet browser, launch the Fabric console at <http://localhost:9082/fabric/login.jsp> (refer to installation configuration for the correct port name). Specify the user ID as `fabricadmin` and password as `password`.
2. Create organization `ITSOBankOrg`:
  - a. From the menu, click **MyServices** → **Subscriber Manager** → **Manage Subscribers**.
  - b. In the Create Organization panel, in the Organization Name field, type `ITSOBankOrg`, and in the Address field, type `ITSOBank`.
  - c. Click the **Create Organization** button, as shown in Figure 9-17.

The screenshot displays the 'WebSphere Business Services Fabric' administrative console. On the left is a navigation menu with 'My Services' expanded, showing options like 'Business Services Repository', 'Governance Manager', 'Performance Manager', 'Subscriber Manager', 'Manage Enrollments', 'Manage Subscribers' (which is selected), 'Manage Subscriptions', 'Service Portfolio', and 'Help'. The main content area is titled 'Create an Organization' and includes a note: 'Fields marked with an asterisk (\*) are required fields.' The form is divided into three sections: 'Organization Details' with a required 'Organization Name' field containing 'ITSOBankOrg' and a 'Select a Parent...' dropdown set to 'N/A'; 'Address Information' with a required 'Address' field containing 'ITSOBank'; and 'Contact Information' with a 'Phone' field. At the bottom right are 'Create Organization' and 'Cancel' buttons. The top right corner shows a user greeting 'Welcome, Fabric' and a login timestamp.

Figure 9-17 Create organization

# Associate user

Users in Fabric can be classified as two types:

- ▶ Fabric users  
Fabric users are users who need access to Fabric modules, particularly the Fabric Administration console and Composition Studio for assembling CBA.
- ▶ Business service users  
Business service users are typically non Fabric users who do not want to access the Fabric administration console or develop CBAs, but want to subscribe to the business service, for example Loan Officer or Customer who needs to subscribe and invoke the Loan Processing service.

In the Assemble phase, you typically associate Fabric users, while in the Manage phase you associate business service users.

For the ITSOPBank scenario, we created a user, Jerry Sander, with a role of Developer in LDAP, which we would associate to ITSO Bank Organization. Jerry Sander is one of the developers in ITSO Bank who will use Composition Studio to assemble a CBA.

To associate the user:

1. Logon to the Fabric Administration Console using the user name fabricadmin, and click **My Services** → **Subscriber Manger** → **Manage Subscribers**, and select the **ITSOPBankOrg** organization, as shown in Figure 9-18.

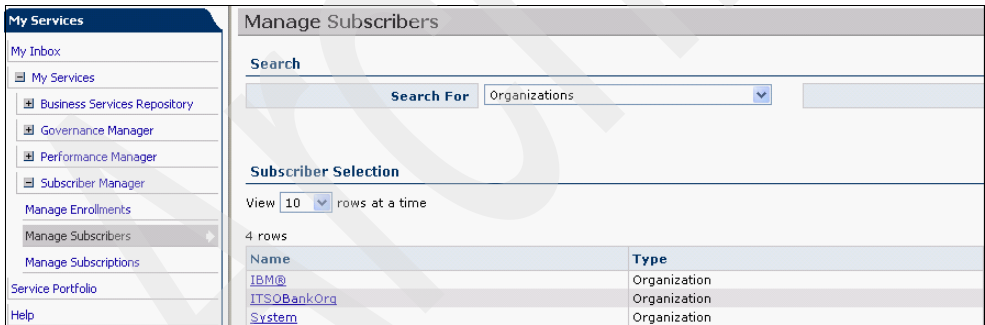


Figure 9-18 Select Organization

2. Click the **Users** tab, and select **Grant User Roles**.
3. On the Grant User Roles window, select **Jerry Sander** from Available Users, and add it to the selected Users list box by clicking the > button, as shown in Figure 9-19 on page 223. Also from the Available Roles, select **Developer**, add it to the Selected Roles list box, and click the **Grant User Roles** button. The Available users are fetched from the user repository which is configured

through VMM in WebSphere Process Server. In this way, you linked a user from the external repository, assigned him the required role, and added him to the organization.

Organization: ITSOBankOrg

Organization Details | Sub Organizations | **Users** | Enrollments

**Grant Users Roles**

Fields marked with an asterisk (\*) are required fields.

**User Selection**

Search Type: Last Name | Search For:

**Available Users**

- Admin, Fabric (fabricadmin)
- Administrator, System (system\_user)
- Administrator, System (wadmin)
- wsadmin, wsadmin (wsadmin)
- Admin, WSRRFabric (WSRRFabricAdmin)

**\* Selected Users**

- Sander, Jerry (jerrys)

**Role Selection**

**Available Roles**

- Administrator
- Agent
- Architect
- Business Analyst
- Business Architect
- Customer
- Default Role
- Employee
- Knowledge Engineer
- Loan Agent

**\* Selected Roles**

- Developer

**Grant Us**

Figure 9-19 Associate users to organization

## 9.4 Synchronizing WSRR with Fabric

Before you synchronize WSRR with Fabric, you first need to source all technical service artifacts (WSDLs and XSD) in WSRR. This is fairly independent and can be carried out when you have the service artifacts ready. After you add WSDLs in and publish them as part of your governance process in WSRR, you then create a WSRR project in Fabric to source technical data from WSRR.

## 9.4.1 Importing the WSDL in WSRR

To import the WSDL for the ITSOBank in WSRR, logon to WSRR Service Registry using URL `http://localhost:port/ServiceRegistry` (replace port according to your configuration) and type in `WSRRFabricAdmin` (or `fabricadmin`) as the username.

**Note:** WSRRFabricAdmin is a user in LDAP. See Chapter 17, “Integrating with Lightweight Directory Access Protocol” on page 481 for more details. We created a separate user to manage the WSRR configuration.

To add the WSDLs for ITSOBank:

1. Select **Administration** from the perspective, and click **Go**. After the Administration perspective is loaded, click **Service Documents** → **Load Documents**.
2. From the additional material that we supplied with this book, go to the Asset\WSDL folder, and select **Local file system** and **ITSO\_impl\_CreditCheckExport1.wsdl**. Select Document Type as **WSDL**, and click **OK**, as shown in Figure 9-20.

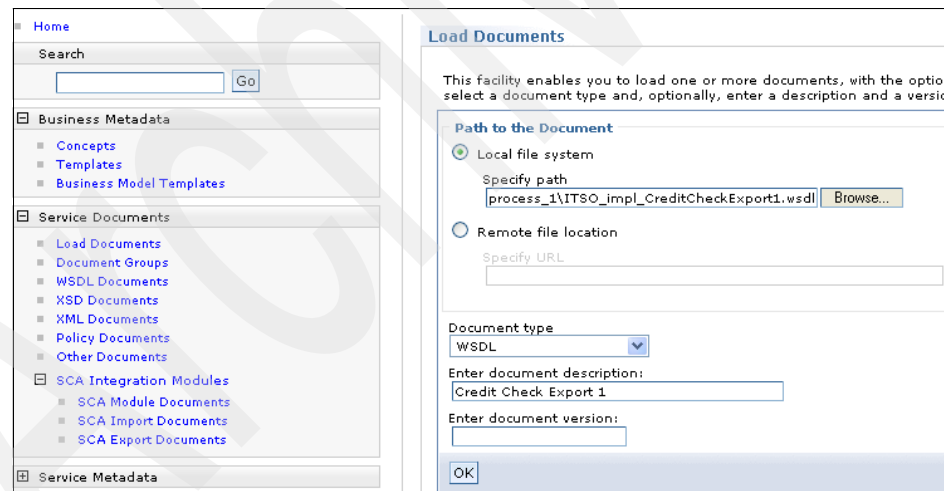


Figure 9-20 Import WSDL

3. As `ITSO_impl_CreditCheckExport1.wsdl` references the `CreditCheck WSDL` file, the next thing you need to add is the `CreditCheck WSDL` by clicking the **Add Another Document** button next to `CreditCheck WSDL`, as shown in Figure 9-21 on page 225.

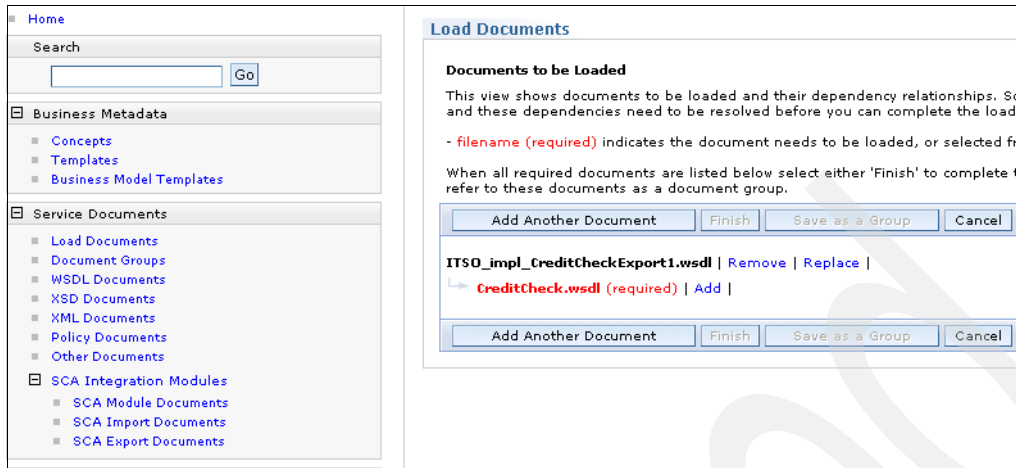


Figure 9-21 Add reference WSDL

4. From the Asset\WSDL folder, add the CreditCheck WSDL, and click **OK**.
5. As the CreditCheck WSDL references an XSD, the next thing you need to add is a corresponding XSD file. Click **Add** next to the xsd, for example ITSOLoanObject.xsd. On the following page, import the corresponding XSD from the Asset\XSD folder, and click **OK**, as shown in Figure 9-22.

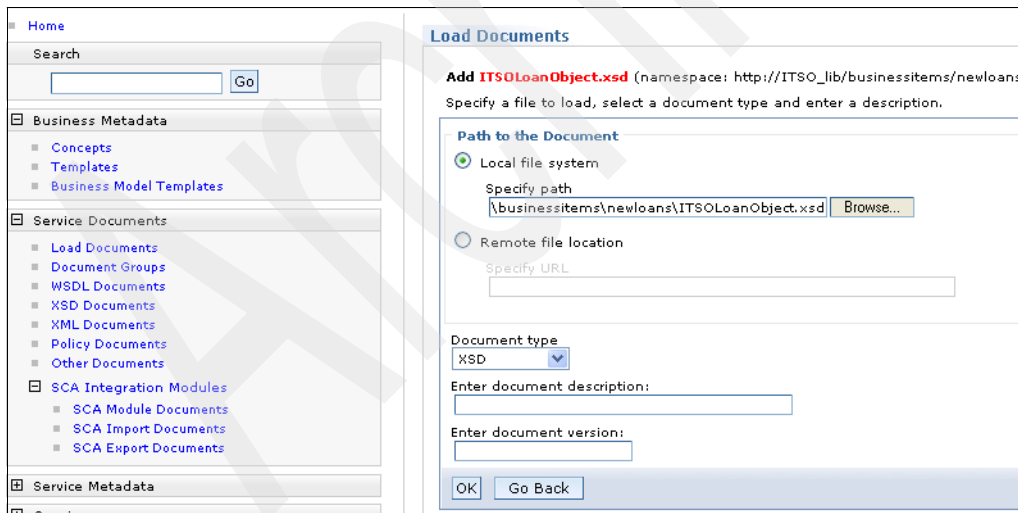


Figure 9-22 Add XSD

6. Add all of the references for CreditCheck, and click **Finish** to save the WSDLs and XSD in WSRR.

- Repeat steps 1-6 to add all of the remaining WSDLs, which contain Exports from the Asset folder. The Export WSDLs contains binding information. If the WSDL or XSD already exists, it is reused, so you do not need to replace it. Figure 9-23 list all of the WSDL that is added to WSRR.

Name	Description	Namespace
ITSO_LoanProcessExport2.wsdl		http://com/ITSOBank/LoanProcess/LoanProcessInterface/Binding
VerifyCustomerInterface.wsdl		http://com/ITSOBank/LoanProcess/VerifyCustomer/VerifyCustomerInterface
ITSO_impl_CreditCheckExport2.wsdl		http://com/ITSOBank/LoanProcess/CreditCheck/CreditCheckInterface/ExternalCreditCheckLowCost/Binding
ITSO_impl_LoanProvidersExport4.wsdl		http://com/ITSOBank/LoanProcess/LoanProviders/LoanProvidersInterface/LowRiskLoanProvider/Binding
ITSO_impl_LoanProvidersExport2.wsdl		http://com/ITSOBank/LoanProcess/LoanProviders/LoanProvidersInterface/PremiumLoanProvider/Binding
LoanProviders.wsdl		http://com/ITSOBank/LoanProcess/LoanProviders/LoanProvidersInterface
CreditCheck.wsdl		http://com/ITSOBank/LoanProcess/CreditCheck/CreditCheckInterface
Rating.wsdl		http://com/ITSOBank/LoanProcess/RatingService/RatingServiceInterface
newloans.xsd		http://ITSO/BusinessItems/NewLoans
ITSO_impl_LoanProvidersExport3.wsdl		http://com/ITSOBank/LoanProcess/LoanProviders/LoanProvidersInterface/MediumRiskLoanProvider/Binding
ITSO_impl_VINLookupExport1.wsdl		http://com/ITSOBank/LoanProcess/VINLookup/VINLookupInterface/BusinessHours/Binding
ITSOLoanObject.xsd		http://ITSO_lib/businessitems/newloans
ITSO_impl_CreditCheckExport1.wsdl		http://com/ITSOBank/LoanProcess/CreditCheck/CreditCheckInterface/ExternalCreditCheckHighCost/Binding
ITSO_impl_CreditCheckExport3.wsdl		http://com/ITSOBank/LoanProcess/CreditCheck/CreditCheckInterface/InternalCreditCheck/Binding
LoanProcess.wsdl		http://com/ITSOBank/LoanProcess/LoanProcessInterface
ITSO_impl_RatingExport2.wsdl		http://com/ITSOBank/LoanProcess/RatingService/RatingServiceInterface/Binding
ITSO_impl_VINLookupExport2.wsdl		http://com/ITSOBank/LoanProcess/VINLookup/VINLookupInterface/NonBusinessHours/Binding
ITSO_impl_VerifyCustomerExport1.wsdl		http://com/ITSOBank/LoanProcess/VerifyCustomer/VerifyCustomerInterface/Binding
ITSO_impl_RatingExport1.wsdl		http://com/ITSOBank/LoanProcess/RatingService/RatingServiceInterface/Binding
VINLookup.wsdl		http://com/ITSOBank/LoanProcess/VINLookup/VINLookupInterface
ITSO_impl_LoanProvidersExport1.wsdl		http://com/ITSOBank/LoanProcess/LoanProviders/LoanProvidersInterface/HighRiskLoanProvider/Binding

Figure 9-23 WSDLs for the ITSOBank

**Note:** You can further classify these service artifacts and provide your own governance around the publishing process for the WSDL. For more details, refer to Chapter 15, “Integrating with WebSphere Service Registry and Repository” on page 403.

## 9.4.2 Creating a WSRR project in Fabric

To create a WSRR project in Fabric:

**Note:** For detail integration of WSRR with Fabric, refer to Chapter 15, “Integrating with WebSphere Service Registry and Repository” on page 403.

- Logon to the Fabric Administration console using the username fabricadmin.
- Go to **Governance Manager** → **Configure Projects** → **Create Project**, and enter the information, as shown in Figure 9-24 on page 227.



Create a Project

Fields marked with an asterisk (\*) are required fields.

Project Details

\* Project NameITSOWSRRPrj

Description

Project Specification

\* Project TypeExternal WSRR

External Details

\* Source Nameitsowsrr

\* Sponsored NS URIhttp://itsobank.com/wsrr#

Project Team

\* Team OrganizationITSOBankOrg

Create Project

Cancel

Figure 9-24 Create WSRR project

**Note:** The Source Name is like a prefix for your WSRR namespace.

- Click **Federation Settings**, and provide the required settings for connecting to WSRR, as shown in Figure 9-25. Click **Test Connection** to verify the connection, and click **Save** to store the settings.

ProjectITSOWSRRpRJ

Back to Projects

Project Details

Project Team

Namespaces

Federation Settings

Fields marked with an asterisk (\*) are required fields.

Info

- The connection test was successful.

Connection Details

\* Hostnamelocalhost

\* Port9445

Security Settings

\* Requires SecurityTrue

\* UsernameWSRRFabricAdmin

\* Password

Expiration Settings

\* Data ExpirationTime to Live

\* Time to Live5 min

External Settings

Source Nameitsowsrr

Sponsored NamespaceITSOWSRRpRJ Sponsored NS 1

Save

Test Connection

Figure 9-25 Federation settings

**Note:** If you set up WSRR on a separate WebSphere Process Server profile, you need to create an SSL certificate authentication to enable the two profiles to talk to each other. The settings that we carried out in the ITSOBank scenario, place Fabric and WSRR on the same server profile.

Based on the Time to Live settings, Fabric gets the WSDL information from WSRR and creates the required technical metadata (Interface and Endpoints) in the Business Service Repository.

- To verify if the WSDLs and endpoint metadata is created, click the **Resources** tab, click **Business Service Repository** → **Search Repository**, and in the Search For section, select **Interfaces**. You will see the interface definition, as shown in Figure 9-26, which implies that metadata is sourced successfully from WSRR.

The screenshot shows the 'Resources' tab selected in the top navigation bar. Below it, the 'Search' section has a 'Search For' dropdown menu with 'Interfaces' selected. A 'Name' search box is also present. Below the search section, the 'Resource Selection' section shows 'View 10 rows at a time'. A table displays 7 rows of search results, all of which are 'WSDL Interface' types published by 'WsrrProvider'.

Name	Type	Publisher
<a href="#">CreditCheck</a>	WSDL Interface	WsrrProvider
<a href="#">Event Consumer</a>	WSDL Interface	System
<a href="#">LoanProcess</a>	WSDL Interface	WsrrProvider
<a href="#">LoanProviders</a>	WSDL Interface	WsrrProvider
<a href="#">Rating</a>	WSDL Interface	WsrrProvider
<a href="#">VerifyCustomer</a>	WSDL Interface	WsrrProvider
<a href="#">VINLookup</a>	WSDL Interface	WsrrProvider

Figure 9-26 Viewing Interface metadata

## 9.5 Loading the extended business model into Fabric

If you extended the Fabric business service model to provide business context, you need to import that model into the Business Service Repository. The model

that you created in the Model phase needs to be packaged in an Ontology project, which gets imported into the Business Service Repository.

**Note:** In Fabric, the business service model is organized into projects, and all business service model objects are assigned to various namespaces. You can use namespaces as folders within a project to organize the instances. Namespaces are used to avoid name collisions.

To import the business service model into Fabric you create the project, and then create the namespaces.

### 9.5.1 Creating an Ontology project

1. Open the Fabric administrative console.
2. Click **Governance Manager** → **Configure Projects**. In the Create a Project window, specify the following values, as shown in Figure 9-27:
  - Project Name: ITSOBankOntPrj
  - Project Type: Ontology
  - Team Organization: ITSOBankOrg
  - Description: *Project containing ITSO ontology definitions*

WebSphere Business Services Fabric

Welcome, Fabric • [Logout](#)  
Login Time: Feb 11, 2008 1:32:21 PM

**My Services**

- My Inbox
- My Services
  - Business Services Repository
  - Governance Manager
    - Configure Environments
    - Configure Namespaces
    - Configure Projects
    - Configure Repository
    - Import/Export
    - Manage Changes
    - Performance Manager
    - Subscriber Manager
- Service Portfolio

**Create a Project**

Fields marked with an asterisk (\*) are required fields.

**Project Details**

\* Project Name: ITSOBankOntPrj

Description: Project containing ITSO ontology definitions

**Project Specification**

\* Project Type: Ontology

**Project Team**

\* Team Organization: ITSOBankOrg

Create Project Cancel

Figure 9-27 Create Fabric project

3. Click **Create Project**.

**Note:** You can create four types of projects in Fabric: WSRR, Ontology, Ontology Extension, and Business Service. We already created a WSRR project, which is used when you need to integrate with an external WSRR. Ontology extensions are created when you extend the Fabric business service model to provide your own extensions. The Business Service project is created when you want to create CBA. A Business Service project contains business service instance data that is created using Composition Studio while developing CBAs.

## 9.5.2 Creating namespaces

1. Create a schema namespace:
  - a. Click **Governance Manager** → **Configure Namespaces**. In the Create a Namespace window, specify the following values:
    - Display Name: ITS0BankSchemaNS
    - Namespace Type: Schema
    - Namespace URI: http://itsobank.com/schema#
    - Namespace Prefix: itsosh
    - Owner Project: ITS0BankOntPrj
  - b. Click **Create Namespace**, as shown in Figure 9-28.

The screenshot shows the 'Create a Namespace' dialog in the WebSphere Business Services Fabric interface. The left sidebar contains a 'My Services' menu with options like 'My Inbox', 'My Services', 'Business Services Repository', 'Governance Manager', 'Configure Environments', 'Configure Namespaces' (selected), 'Configure Projects', 'Configure Repository', 'Import/Export', 'Manage Changes', 'Performance Manager', 'Subscriber Manager', 'Service Portfolio', and 'Help'. The main area is titled 'Create a Namespace' and includes a note: 'Fields marked with an asterisk (\*) are required fields.' The form is divided into three sections: 'Namespace Details' with a required 'Display Name' field containing 'ITS0BankSchemaNS' and an optional 'Description' text area; 'Namespace Specification' with required fields for 'Namespace Type' (set to 'Schema'), 'Namespace URI' (set to 'http://itsobank.com/schema#'), and 'Namespace Prefix' (set to 'itsosh'); and 'Namespace Ownership' with a required 'Owner Project' field set to 'ITS0BankOntPrj'. At the bottom right, there are three buttons: 'Create Namespace', 'Create and Start Another', and 'Cancel'.

Figure 9-28 Create namespace

2. Create an extension namespace:
  - a. Click **Governance Manager** → **Configure Namespaces**, and specify the following values:

- Display Name: ITS0BankSchemaextNS
- Namespace Type: Schema
- Namespace URI: http://itsobank.com/schema/extensions#
- Namespace Prefix: itsoshext
- Owner Project: ITS0BankOntPrj

b. Click **Create Namespace**, as shown in Figure 9-29.

**My Services**

- My Inbox
- My Services
  - Business Services Repository
  - Governance Manager
  - Configure Environments
  - Configure Namespaces
  - Configure Projects
  - Configure Repository
  - Import/Export
  - Manage Changes
- Performance Manager
- Subscriber Manager
- Service Portfolio
- Help

**Create a Namespace**

Fields marked with an asterisk (\*) are required fields.

**Namespace Details**

\* **Display Name** ITS0BankSchemaExtNS **Description**

**Namespace Specification**

\* **Namespace Type** Schema

\* **Namespace URI** http://itsobank.com/schema/extensions#

\* **Namespace Prefix** itsoshext

**Namespace Ownership**

\* **Owner Project** ITS0BankOntPrj

**Create Namespace**

Figure 9-29 Create Namespace

### 9.5.3 Importing the Ontology extension project

The business service model that you created needs to be packaged as a Fabric Content Archive (FCA) and then imported into the Business Service Repository. The Fabric Content Archive is a compressed file that can contain two types of content:

- ▶ The first type is Web Ontology Language (OWL) files for one or more namespaces.
- ▶ The second type of content is Translation Bundles that are a collection of properties files for an application that contains static information, such as strings for user interface labels and messages.

Each FCA package also contains an XML manifest that describes the resources in the package, namely content-pack.xml.

To construct an FCA:

1. Create a template of an FCA package by exporting the project using the Export by Project function of the Import/Export business service in Governance Manager. This creates a compressed file that contains a set of OWL files for the exported project namespaces and the XML manifest that describes the contents of the FCA package. To carry out this step for ITSOBank:
  - a. Click **Governance Manager** → **Import/Export**.
  - b. In the Export by Project tab, specify:
    - Content Type: Full content as Content Type
    - Project: ITSOBankOntPrj as Project
    - Namespace: select ITSOBankSchemaNS and ITSOBankSchemaextNS as shown in Figure 9-30.

**WebSphere Business Services Fabric** Welcome, Fabric • [Logout](#)  
Login Time: Feb 11, 2008 1:32:21 PM

**My Services**

- My Inbox
- My Services
  - Business Services Repository
  - Governance Manager
    - Configure Environments
    - Configure Namespaces
    - Configure Projects
    - Configure Repository
    - Import/Export**
    - Manage Changes
  - Performance Manager
  - Subscriber Manager
  - Service Portfolio
- Help

**Import/Export**

**Export by Project**

Fields marked with an asterisk (\*) are required fields.

**Content Type**

\* Content Type Full Content

**Project Selection**

\* Project ITSOBankOntPrj

**Namespace Selection**

1 rows

	Name	Type	Description
<input checked="" type="checkbox"/>	ITSOBankSchemaNS	Schema	

[Export To File](#)

Figure 9-30 Export Project

A compressed file is generated, **ITSOBankOntPrjxxx-owl.zip**, as shown in Figure 9-31.

Name	Size	Type	Date Mo
ITSOBankOntPrj20080211-owl.zip	3 KB	Compressed (zippe...	2/11/2008

Figure 9-31 Exported project from Fabric

2. Browse the content by expanding the zip file `ITSOBankOntPrjxxx-owl.zip`. As shown in Figure 9-32, it contains the following files:
  - `content-pack.xml`
  - `http_itsobank.com_schema_extensions.owl`
  - `http_itsobank.com_schema.owl`

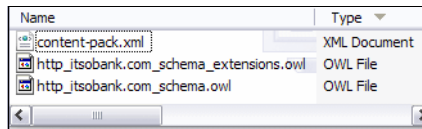


Figure 9-32 Contents of zip file

3. Retrieve the two OWL files **extensions.owl** and **ontology.owl** that were generated as part of the Model phase, which we discussed in 8.7.7, “Exporting the Fabric Content Archive” on page 192:
  - a. Rename the **extensions.owl** file to `http_itsobank.com_schema_extensions.owl`.
  - b. Rename the **ontology.owl** file to `http_itsobank.com_schema.owl`.
4. Replace the original files in the zip file from Step 2 with these new files from the previous step:
  - `http_itsobank.com_schema_extensions.owl`
  - `http_itsobank.com_schema.owl`

**Note:** The system requires that an extension namespace URI, in the `content-pack.xml` manifest for a Fabric Content Archive, match the base namespace in the corresponding OWL file and match the namespace prefix of the identity of any Class, Property, or Instance declared in that OWL file.

5. The new Fabric Content Archive zip file `ITSOBankOntPrjxxx-owl.zip` is ready to be imported into Fabric:
  - a. In the **Fabric console**, click **Governance Manager** → **Import/Export**.
  - b. Browse and click the file **ITSOBankOntPrjxx-owl.zip** that was generated from the previous step.
  - c. Click the **Import File** button, as shown in Figure 9-33 on page 234.

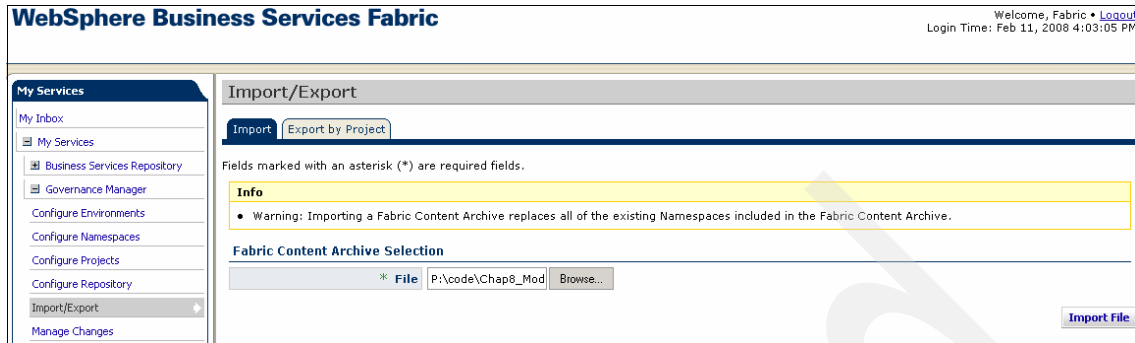


Figure 9-33 Import/Export

Figure 9-34 shows the successful import of the file.

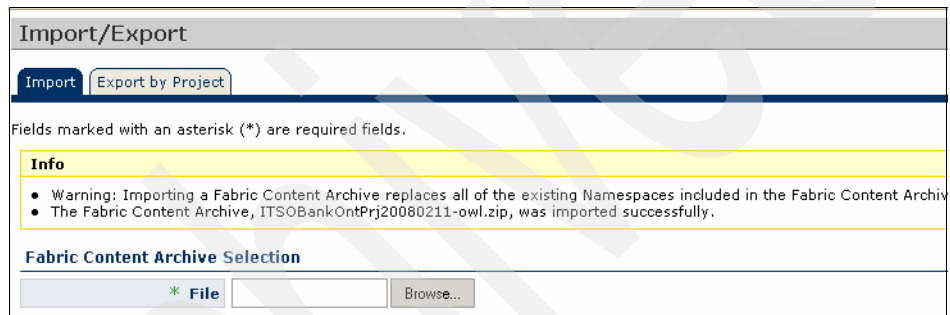


Figure 9-34 Successful import message

You successfully imported your business model extension into Business Service Repository.

## 9.5.4 Creating a Fabric project

In this step, you create the business service project for your CBA. A business service project contains business service instance data that is created using Composition Studio while developing CBAs.

1. Click **Governance Manager** → **Configure Projects**, and specify:
  - a. Type **ITS0BankCBAPrj** as the Project Name.
  - b. Select **Business Service** as the Project Type.
  - c. Select **ITS0BankOrg** as the Team Organization.
  - d. Click **Create Project**, as shown in Figure 9-35 on page 235.





- b. To import the ITSOWSRRPrj namespace:
  - i. Select **ITSOBankCBAPrj**, and click the **Namespaces** tab.
  - ii. Select **ITSOWSRRPrj** as the Project, and select the **ITSOWSRRPrj Sponsored NS 1** namespace, as shown in Figure 9-37.
  - iii. Click **Import Namespaces**.

Project: ITSOBankCBAPrj [Back to Projects](#)

Project Details Project Team **Namespaces**

**Import Namespaces**

Fields marked with an asterisk (\*) are required fields.

**Info**

- Select a **Project** and **Namespaces** to import.

**Project Selection**

\* Project: ITSOWSRRPrj

**Namespace Selection**

1 rows

<input checked="" type="checkbox"/>	Namespace	Type	Description
<input checked="" type="checkbox"/>	ITSOWSRRPrj Sponsored NS 1	Sponsored	

[Import Namespaces](#) [Cancel](#)

Figure 9-37 Import WSRR Namespace

3. Create instance and enrollment namespaces. All of the metadata instances that are created in Composition Studio by users goes to the instance namespace, while the enrollment namespace stores enrollment and subscription information. A CBA project should have at least one instance and one enrollment namespace. To create the Instance namespace:
  - a. In the Namespaces tab, click the **Create a Namespace** button, as shown in Figure 9-38 on page 237, and specify:
    - i. Display Name: Specify ITS0BankINS
    - ii. Namespace Type: Instance
    - iii. Namespace URI: `http://itsobank.com/inst#` as the Namespace URI
    - iv. Click **Create Namespace**, as shown in Figure 9-39 on page 237.

**Configure Projects**

Project: ITSOBankCBAPrj [Back to Project](#)

Project Details | Project Team | **Namespaces**

**Owned Namespaces**

0 rows

Namespace ↑	Type	Description
-------------	------	-------------

[Create a Namespace](#)

Figure 9-38 Create Namespace

**Configure Projects**

Project: ITSOBankCBAPrj [Back to Project](#)

Project Details | Project Team | **Namespaces**

**Create a Namespace**

Fields marked with an asterisk (\*) are required fields.

**Namespace Details**

\* Display Name: ITSOBankINS Description:

**Namespace Specification**

\* Namespace Type: Instance

\* Namespace URI: http://itsobank.com/inst#

[Create Namespace](#) [Create and Start Another](#) [Cancel](#)

Figure 9-39 Create Instance namespace

- b. To create the Enroll namespace, specify:
  - i. Display Name: ITSOBankEnrollNS
  - ii. Namespace Type: **Enrollment**
  - iii. Namespace URI: http://itsobank.com/enroll#
  - iv. Click **Create Namespace**, as shown in Figure 9-40 on page 238.

**Configure Projects**

**Project** ITSOBankCBAPrj [Back to Projects](#)

[Project Details](#) [Project Team](#) **[Namespaces](#)**

**Create a Namespace**

Fields marked with an asterisk (\*) are required fields.

**Namespace Details**

* <b>Display Name</b>	ITSOBankEnrollINS	<b>Description</b>	
-----------------------	-------------------	--------------------	--

**Namespace Specification**

* <b>Namespace Type</b>	Enrollment
* <b>Namespace URI</b>	http://itsobank.com/enroll#

[Create Namespace](#) [Create and Start Another](#) [Cancel](#)

Local intranet

Figure 9-40 Create Enroll namespace

Figure 9-41 on page 239 shows the created instance and enrollment namespaces.

Configure Projects

ProjectITSOBanCBAPrjBack to Projects

Project DetailsProject TeamNamespaces

Owned Namespaces

2 rows

Namespace ↑	Type	Description
ITSOBankINS	Instance	
ITSOBankEnrollINS	Enrollment	

Create a Namespace

Imported Namespaces

6 rows

Namespace ↑	Type	Description	Primary Project	Remove
Organizations	Instance	Stores the organization structure defined using IBM Business Services Subscriber Manager	Organizations, Users, and Roles	<input type="checkbox"/>
Local Users	Instance	Stores users that were created using IBM Business Services Subscriber Manager	Organizations, Users, and Roles	<input type="checkbox"/>
ITSOBankSchemaNS	Schema		ITSOBankOntPri	<input type="checkbox"/>
External Users	Instance	Represents users from external registries	Organizations, Users, and Roles	<input type="checkbox"/>

Local intranet

Figure 9-41 Created Instance and Enroll namespaces

## 9.5.5 Configuring environments

In Fabric, an environment represents a system where endpoints can be deployed, for example, a service can be deployed in a development, test, training, or production environment. A Business Service Repository instance can have endpoints in all of these environments, but if the Business Service Repository is being used for testing, only endpoints in the test environment are considered.

In the Governance Manager, you can create named environments using the Configure Environments page. Using the Configure Repository service, an administrator can select which named environments that the current Business Service Repository instance is free to access. Finally, you can mark endpoints as being deployed in any of these named environments.

If an environment URI is provided in the context, the system automatically filters candidate endpoints down to those in the given environment. If an environment is not listed, the system uses the settings from the Configure Repository to filter endpoints from any environment that is not marked as supported by the current Business Service Repository. The fully qualified URI of an environment is

available on the Configure Environments page in the Fabric Administrative Console.

The system comes with one environment named Default. If scoping selection of endpoints to a specific environment is not yet a concern, using this environment is a shortcut. You can configure what endpoints that your Repository supports using the Configure Repository business service in Governance Manger. If there is only one known environment when an endpoint is created, that environment is automatically selected. You can ignore the complexity of scoping endpoint availability by environment, if you have only one.

For the ITSObank, we create two environments, *Development* and *Production*. You can export and distribute the configurations for these environments to developers so that they can reuse it.

**Note:** If there are multiple developers working on assembling the CBA, you need to first create a master set of environments on one Fabric server environment and then export the environment instances as an FCA project to be used on other Fabric server environments.

The system creates a unique ID when you create an environment variable. If each developer creates their own environment variables, say development, then when you merge them while promoting your CBA from development to test or production environments you end up having two environments that have the same name as development with different environment IDs. This is not advisable.

To configure environments:

1. Create a development environment configuration:
  - a. In the Fabric console, click **Governance Manager** → **Configure Environments** → **Create an Environment**, and specify:
    - i. Environment Name: Development
    - ii. Description: Development Environment, as shown in Figure 9-42 on page 241.
  - b. Click the **Create Environment** button.

**WebSphere Business Services Fabric**

Welcome, Fabric • [Logout](#)  
 Login Time: Feb 12, 2008  
 5:03:40 PM

**My Services**

- My Inbox
- My Services
  - Business Services Repository
  - Governance Manager
  - Configure Environments
  - Configure Namespaces
  - Configure Projects
  - Configure Repository
  - Import/Export
  - Manage Changes

**Create an Environment**

Fields marked with an asterisk (\*) are required fields.

**Environment Details**

* <b>Environment Name</b>	Development
<b>Description</b>	Development Environment

[Create Environment](#) [Cancel](#)

Figure 9-42 Create an Environment

2. Create a new environment configuration for Production:
  - a. In the Fabric console, click **Governance Manager** → **Configure Environments** → **Create an Environment**, and specify:
    - i. Environment Name: Production
    - i. Description: Production Environment
  - b. Click **Create Environment** button.
3. Export the environment configuration:
  - a. Click **Governance Manger** → **Import/Export**, and specify:
    - i. Select **Full Content** for Content.
    - ii. Select **Fabric Governance** as the Project.
    - iii. Select **Environments**.
  - b. As shown in Figure 9-43 on page 242, click **Export To File** to export the environment configuration.

Address <http://localhost:9082/fabric/app/?wicket:interface=:48:1:> Go Links

- Business Services Repository
- Governance Manager
- Configure Environments
- Configure Namespaces
- Configure Projects
- Configure Repository
- Import/Export
- Manage Changes
- Performance Manager
- Subscriber Manager
- Service Portfolio
- Help

Fields marked with an asterisk (\*) are required fields.

**Content Type**

\* Content Type Full Content

**Project Selection**

\* Project Fabric Governance

**Namespace Selection**

1 rows

<input checked="" type="checkbox"/>	Name	Type	Description
<input checked="" type="checkbox"/>	Environments	Instance	User-created environments are stored in this namespace. This namespace has to be exported to other WebSphere Business Services Fabric installations when exporting content that refers to these environments.

Export To File

Figure 9-43 Export environment configuration

This exported compressed file is distributed to developers to reuse the environment configuration.

## 9.5.6 Replicating the project in Composition Studio

In this section, we replicate the business service project in Composition Studio. Replicating the project creates a local copy of Business Service Repository (which uses a Derby database) in your workspace. All of the changes that you make go to your local Business Service Repository.

To replicate the project in Composition Studio:

1. Open **Composition Studio**, and switch to the **Business Services** perspective.
2. Click **File** → **New**, and select **Fabric Project**, as shown in Figure 9-44 on page 243.



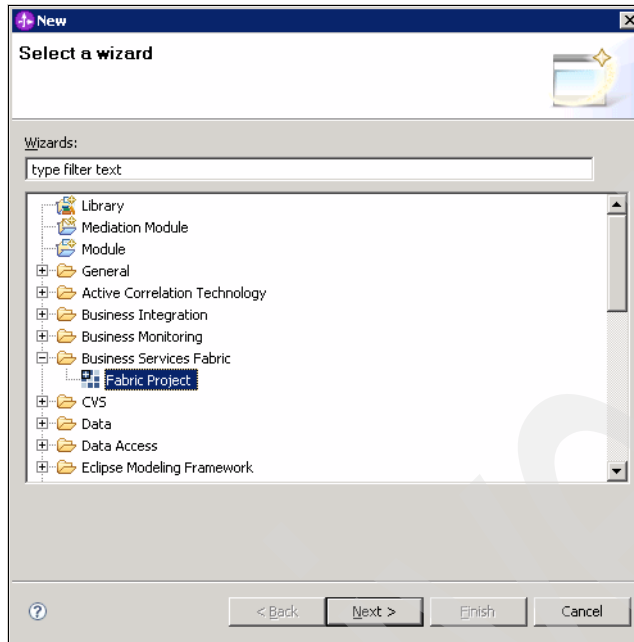


Figure 9-44 New Fabric project in Composition Studio

3. Name the Fabric project ITS0BankCBAPrj, as shown in Figure 9-45.

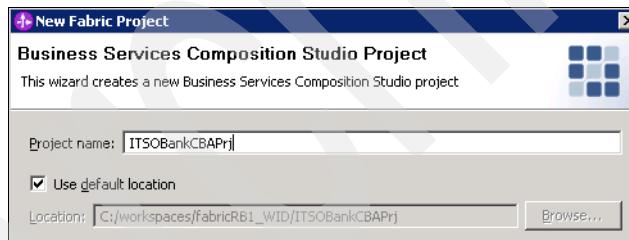


Figure 9-45 Name the Fabric project

4. Configure the Business Services Repository connection, as shown in Figure 9-46 on page 244. Specify user as jerrys and password as password.

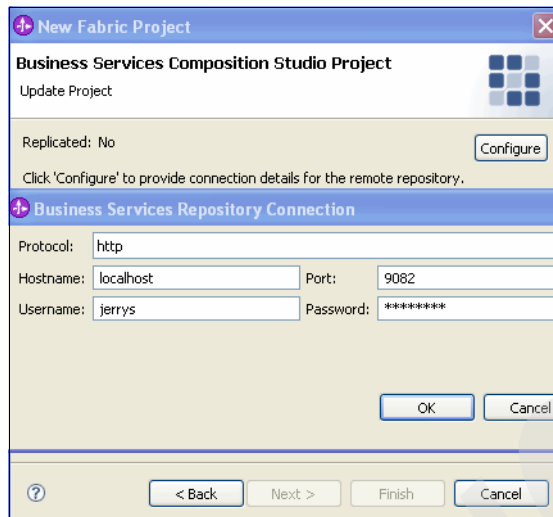


Figure 9-46 Specify user credentials

5. Run the replication, as shown in Figure 9-47.

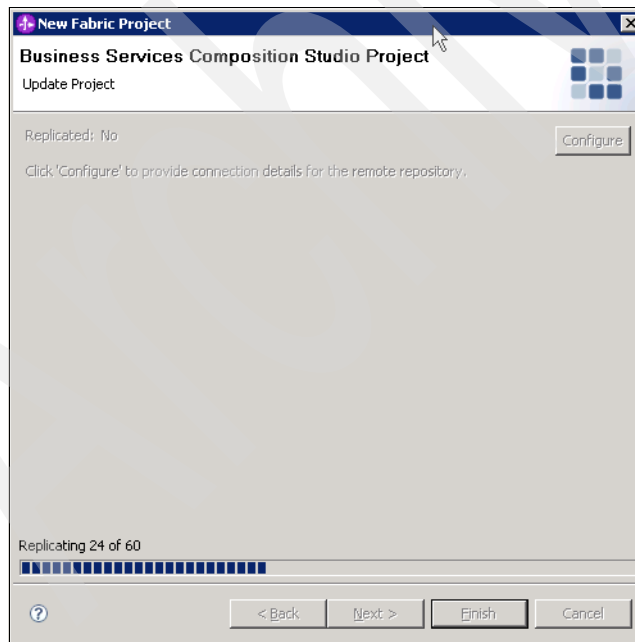


Figure 9-47 Replicate Business Services Project

6. After the replication is complete, as shown in Figure 9-48, click **Next**.

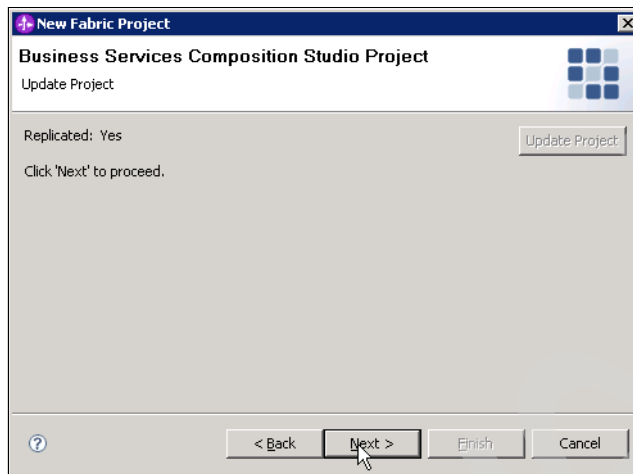


Figure 9-48 Update project

7. From the Fabric project pull-down list, select the **ITSOBankCBAPrj** project, as shown in Figure 9-49. If a user is associated with the project, only that user can see the respective projects in the projects list. As you recollect, a project is tied to any organization, and users are associated through organizations using roles. Click **Finish** to complete the creation of the Fabric project using the Business Services repository.

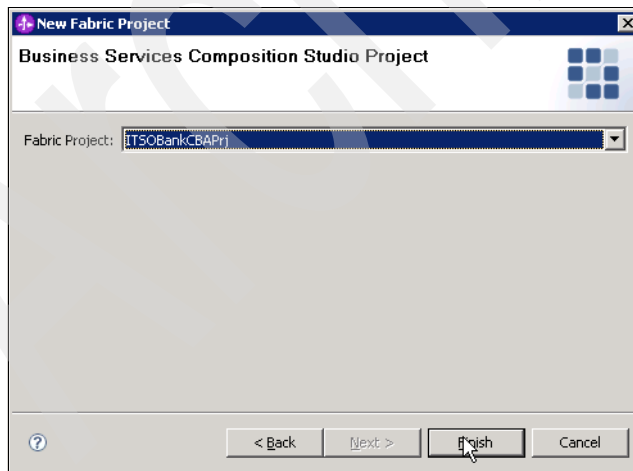


Figure 9-49 Fabric Project creation

8. You see the ITSOBankCBAPrj project in the Business Services view, as shown in Figure 9-50. Expand the Interfaces and Endpoint node to see the metadata that was imported from ITSOWSRRPrj project. Double-click any of the interfaces, say BusinessHoursVINLookup\_VINLookupHttpPort, to see the editor open up with all of the information grayed out (read only mode), which is because you imported the metadata from a WSRR project, and you cannot modify the imported metadata in the ITSOBankCBAPrj project.

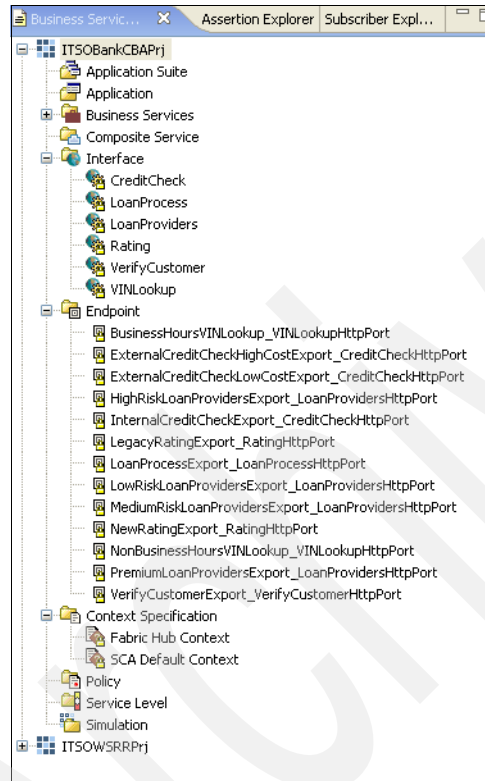


Figure 9-50 View ITSOBankCBAPrj project

9. To make changes to the ITSOWSRRPrj project:
  - a. Import the project into your workspace.
  - b. Repeat steps 1- 6.
  - c. Name the project ITSOWSRRPrj, select **ITSOWSRRPrj** from the project list, and click **Finish**.

You will see the ITSOWSRRPrj in the Business Service view. During this step you do not need to provide your username again for replication because you already supplied this earlier.

## 9.5.7 Creating business context for your CBA

In this section, you create the business context for your CBA. Here we provide the high-level steps that you perform:

1. Create Business Service Domain objects. The Business Service Domain includes business services, applications, and application suites.
2. Create Service Technical Domain objects. The Service Technical Domain is composed of interfaces, composite services, Dynamic Assembly components, and endpoints. Here are the high-level steps:
  - a. Import the SCA module (business process) created from WebSphere Integration Developer as a composite service. This creates the composite service meta definitions, interface definitions, and references from SCA modules, and creates the required metadata in Composition Studio.
  - b. Link the business service to IT implementations (technical services).
  - c. Create a context specification and associate it with the Dynamic Assembler metadata that you created in Step 1.
  - d. Create endpoints information for the composite service and interfaces that you imported from Step 1. Target endpoints to respective environments.
3. Apply capabilities to endpoints by adding required assertions.
4. Create business policies to select the right service implementation based on the context, content, and contract.

**Note:** The process variability points of your business process are extracted out from your process and handled declaratively using business policies. Each of the process variability points are normally handled by creating business policies.

5. Simulate policies using the Policy Simulator to ensure the right endpoint implementation is being picked up.

### Creating the Business Services Domain metadata

In this section, we create the business service domain objects for the Loan Processing application. The Business Service Domain includes business services, applications, and application suites:

- **Business services:** A business service is a business function whose execution can be adapted at runtime based on business policies and user context.

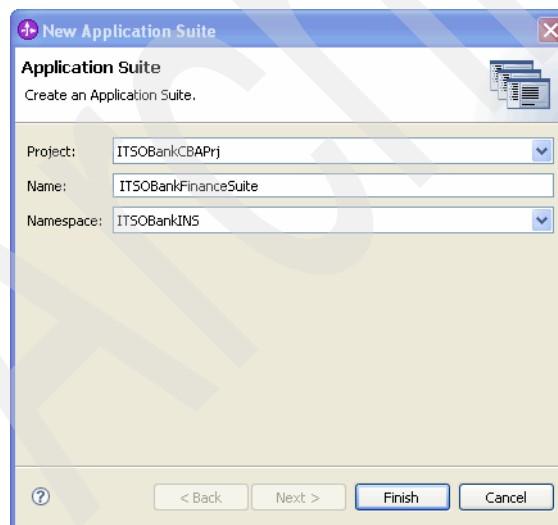
- ▶ **Applications:** An application or composite business application is a logical collection of related business services.
- ▶ **Application suites:** An application suite is a logical grouping of related applications.

Business Service Domain objects represent a rich set of metadata that lets you provide business context around your technical services. While creating the business service, you define how the business services is delivered (over which channels), what implementation is used for the realization, and which roles will have access to it. After you create the business service, you can enroll the business service by an organization and subscribed by the users belonging to the required roles within that organization, for instance, a Bank providing various services, such as Auto Loans and Home Loans, can be enrolled by various vendors or bank branches, and users (Loan officers or Agents) can subscribe to either the Auto Loan or Home Loan business services to use these services.

### ***Creating an application suite***

To create an application suite:

1. Right-click the **ITSOBankCBAPrj** project, and select **New** → **Application Suite**.
2. On the New Application suite window, complete the Project, Name, and Namespaces fields, as shown in Figure 9-51, and click **Finish**.



*Figure 9-51 Create Application Suite*

## Creating an application

To create an application:

1. Right-click the **ITSOBankCBAPrj** project, and select **New** → **Application**.
2. On the New Application window, provide the details, as shown in Figure 9-52.

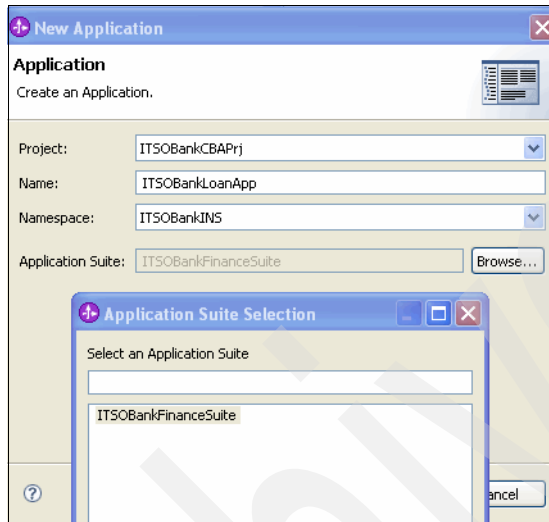


Figure 9-52 Create application

3. Click **Browse**, and select the application suite that you created in the previous step as the parent of this application, as shown in Figure 9-52. Click **Ok**.
4. Click **Finish**.

## Creating a process service

To create a process service:

1. Right-click the **ITSOBankCBAPrj** project, and click **New** → **Business Service**. Select **Process Service**. Click **Next**.

**Note:** A business service can be classified by three types: Process service, Visibility service, and Optimization service. These are just classifications of business services, and a business service should map to one of these types.

2. Provide the details for the process service, as shown in Figure 9-53. Browse and select the parent application that you created in the previous step, as shown in Figure 9-53. Click **OK**. Click **Next**.

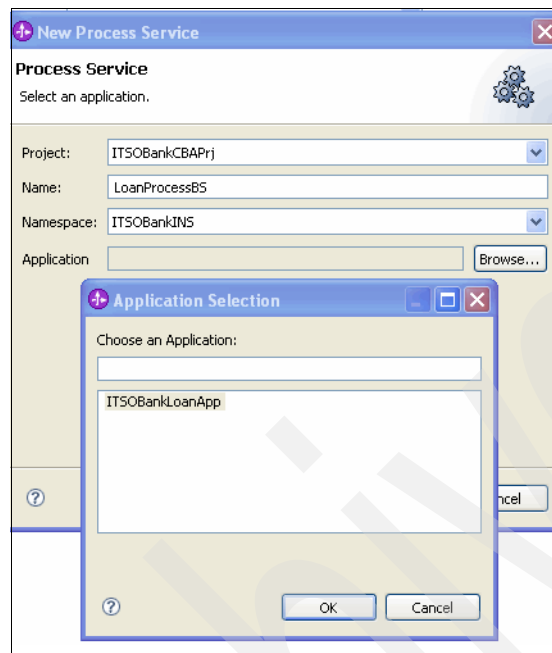


Figure 9-53 Create Process Service

3. Add a channel for the process service. Click **Add Channel**.
  - a. Select **Portal Channel**, as shown in Figure 9-54 on page 251. In the Display Name field, type LoanProcessPortalChannel. Click **OK**. Click **Next**.



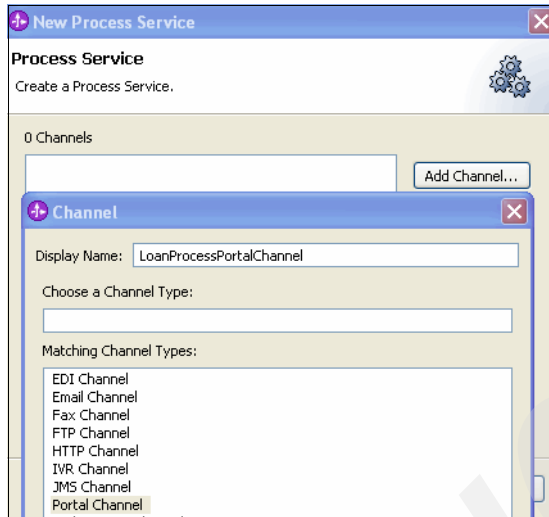


Figure 9-54 Add Portal Channel

4. Add one more channel of type **Web service channel**, and name it LoanProcessB2BChannel1. Click **Next**.
5. Click **Add Role Type**. In the Role Type field, select **Customer, Loan Agent, and Loan Officer**, as shown in Figure 9-55. These roles can access the business service. Click **OK**. Click **Finish**.

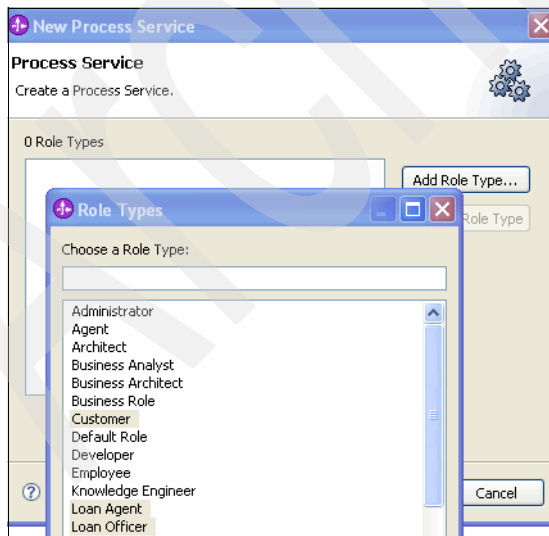


Figure 9-55 Add Role Type

## Creating Service Technical Domain objects

The Service Technical Domain is composed of interfaces, composite services, Dynamic Assembly components, and endpoints. Creating the service technical domain objects using the steps in the next sections.

### Import the SCA module

The SCA module provides the business process implementation that you created in WebSphere Integration Developer. The next step is to import this technical implementation and provide business context around it:

1. To import the SCA module, right-click anywhere on the **Business services** perspective, and click **New** → **Composite Service**.
2. In the Project Name field, select **ITSOBankCBAPrj**. In the Namespace field, select **ITSOBankINS**. In the SCA project field, select **ITSO**, as shown in Figure 9-56.

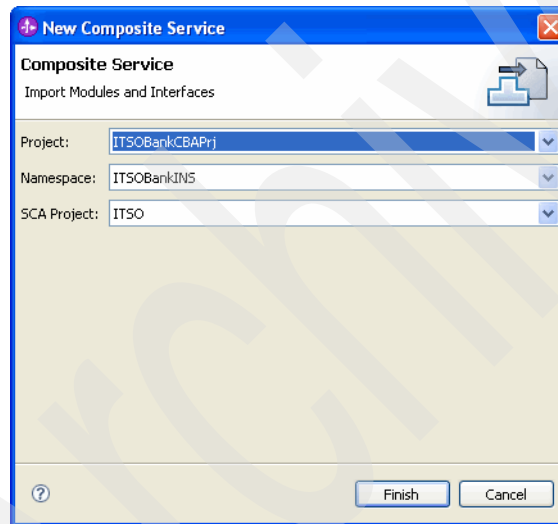


Figure 9-56 Select SCA project

3. Click **Finish**. You will see the warning message shown in Figure 9-57 on page 253. The service interface already exists in your CBA project, which was synchronized from WSRR through the ITSOWSRRPrj project. The SCA module implementation sourced the WSDLs to WSRR. Click **OK**.

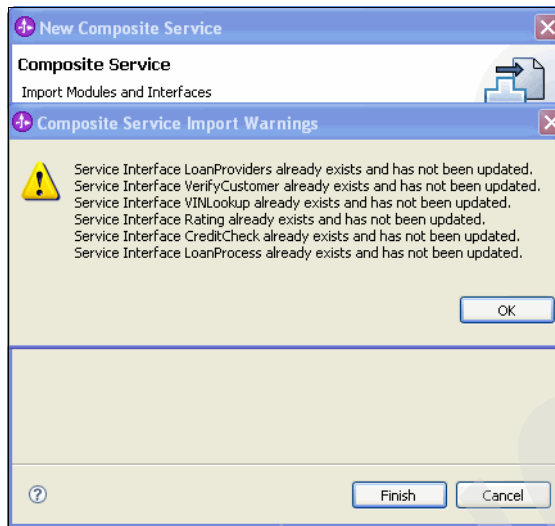


Figure 9-57 Service interface exists warning

The metadata shown in Figure 9-58 on page 254 is generated after importing the SCA module.

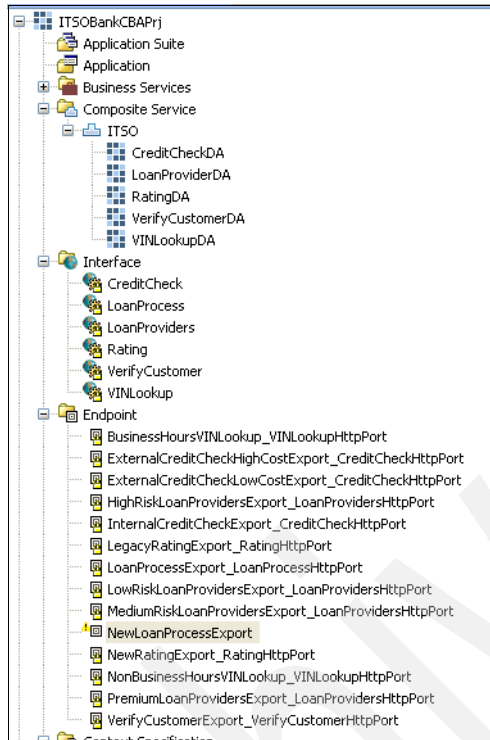


Figure 9-58 SCA Import metadata

## Analyzing the metadata created during the Composite Service import

Based on the assembly diagram for ITSO Loan process, the ITSO Composite Service metadata was created in Compositor Studio.

**Note:** Double-click the metadata, which opens the corresponding editor, and you can view the metadata in detail, for instance, if you double-click ITSO, the Composite Service editor opens up, and you can click the **Exported references and Components** tab information to see references and the component being used by the ITSO composite service.

In the assembly diagram, the Loan Process references the various Dynamic Assembler components for each of the referenced Dynamic Assembler components. The following Dynamic Assembler component metadata is created (Table 9-3 on page 255).

Table 9-3 SCA metadata mapping

SCA Dynamic Assembler component in Assembly editor	Dynamic Assembly component in Composition Studio
CreditCheckDA	CreditCheckDA
LoanProviderDA	LoanProviderDA
RatingDA	RatingDA
VerifyCustomerDA	VerifyCustomerDA
VINLookupDA	VINLookupDA

The Dynamic Assembler component imports the respective service interfaces, for example, CreditCheckDA imports the Credit Check interface. When you import the SCA module, the metadata for interfaces imported by the Dynamic Assembler component is also created. Because the interface metadata already exists in your ITSOBankCBAPrj project, it gives you a warning message that says that the service import exists and will not be updated.

The endpoint metadata is also created based on the binding information that is associated with the service interfaces.

### ***Creating a context specification and associating it with the Dynamic Assembler Component***

The context specification is a declaration of context dimension that is relevant for a dynamic assembly decision. You can declare which model dimensions are employed, for instance, environment or an organization, and which content-based assertions are used. If a dimension is marked as required, the system fails at dynamic assembly time if a value for the dimension is not specified in the context.

The context specifications serves as a contract between the policy-based dynamic assembly scenarios configured and tested in Composition Studio and the context extractor logic that a developer must set up that configures Dynamic Assembler components in WebSphere Integration Developer.

The Dynamic Assembler component metadata in Composition Studio needs to be linked to a context specification. The system provides an SCA Default Context that can be tied to the Dynamic Assembler component but in that case the contract is strict typed, and you cannot restrict at runtime if a particular context needs to be supplied.

Its a recommended practice to create a context specification that best models the extractor code that you wrote in WebSphere Integration Developer, while assembling the business process.

For the ITSOBank, Table 9-4 provides the context extractor that was coded for the respective Dynamic Assembler component while assembling the business process.

*Table 9-4 Dynamic Assembler component and context extractor used for ITSOBank Loan Process in Assembly editor.*

Dynamic Assembler component in Assembly Editor	Model and content assertion required	Context Extractor
VerifyCustomerDA	Model = Environment Content Based Assertion= Bank Id	VerifyCustomerContextExtractor
LoanProviderDA	Model = Environment Content Based Assertion= <i>CUSTOMER_TYPE_ASSERTION</i> and <i>RATING_SCORE_ASSERTION</i>	LoanProviderContextExtractor
RatingDA	Model = Environment Content Based Assertion= <i>CUSTOMER_TYPE_ASSERTION</i>	RatingDAContextExtractor
VINLookupDA	Model = Environment	
CreditCheckDA	Model = Environment Content Based Assertion= <i>CUSTOMER_TYPE_ASSERTION</i> and <i>LOAN_AMOUNT_ASSERTION</i>	CreditCheckContextExtractor

Based on the above table we will create 5 context specification objects in Composition Studio.

**Note:** You do not need to create a context specification for each of the context extractor codes. If context that is required by the Dynamic Assembler components are similar, you can create a single context extractor object and tie it to respective Dynamic Assembler component object.

To create a context specification object in Composition Studio:

1. Click the **Context Specification** node, and select **New** → **Context Specification**.
2. On the New context specification window, in the Name field, type CreditCheckCS, as shown in Figure 9-59, and click **Finish**.

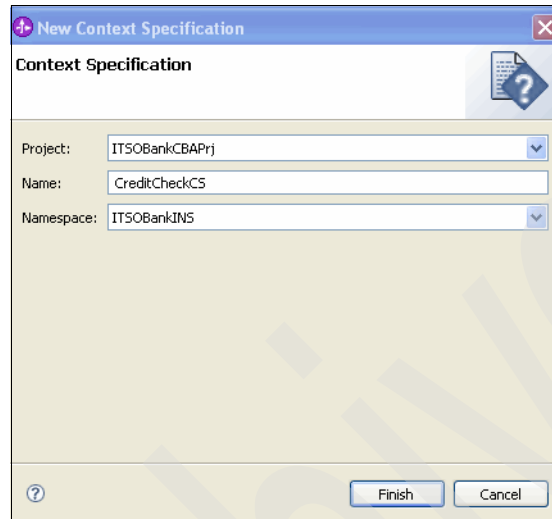


Figure 9-59 CreditCheckCS Specification

3. The CreditCheckCS object is created under the Context Specification node. Double-click the **CreditCheckCS**. The editor opens up. Select the **Dimensions** tab. The CreditCheckCS maps to the CreditCheckContextExtractor code, which requires Environment as the Model dimension and Customer Type and Loan Amount as a content-based assertion. For adding these dimensions, click **Add** on Model Dimensions, and select **Environment** from the Model dimension dialog box, as shown in Figure 9-60 on page 258. Click **OK**.

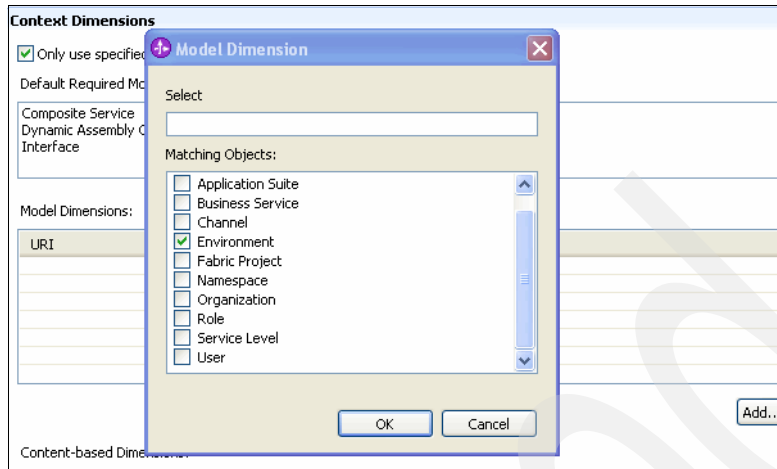


Figure 9-60 Model dimension

- a. Click **Add** for the content-based assertions, and select the **Customer Type Assertion** and **Loan Amount Assertion**. Save the editor using Ctrl+S.
4. Link the CreditCheckCS to the CreditCheckDA object. Click the CreditCheckDA object. The editor opens up. Click **Browse**. As shown in Figure 9-61 on page 259, select **CreditCheckCS**, and click **OK**, and save the editor.



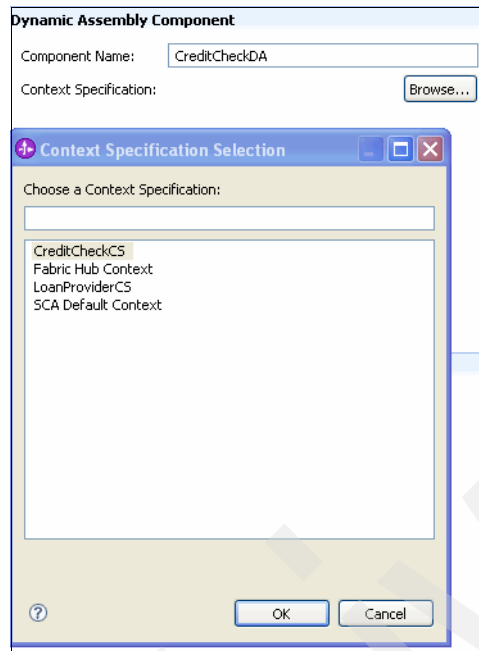


Figure 9-61 Link DA to context specification

5. Create context specification objects with the following model dimensions and content based assertions, and link it to the Dynamic Assembler using information in Table 9-5.

Table 9-5 Context specification objects

Name	Model dimension	Content-based dimensions	Link to DA component
LoanProviderCS	Environment	Customer Type Assertion and Rating Score Assertion	LoanProviderDA
VerifyCustomerCS	Environment	Bank Identifier	VerifyCustomerDA
RatingCS	Environment	Customer Type Assertion	RatingDA
LoanProcessDefaultCS	Environment	None.	VINLookupDA

## Creating endpoints

In this step, we create endpoint objects. We imported the WSDL information from WSRR and based on the binding information that is associated with WSDL the endpoint metadata was also created in the ITSOWSRRPrj project. As the ITSOPBankCBAPrj project imports the ITSOWSRRPrj project, we can see the endpoint metadata as read only. Typically you use the endpoint information for reference only and create new ones in your CBA project. In this way, multiple CBA projects can reference the WSDLs and endpoints from a WSRR project without modifying its content.

You can create endpoint object in two ways:

- ▶ If your implementation is based on SCA, you can import the SCA module using the Import Composite service that creates the interfaces and endpoint metadata.
- ▶ You can use the Create Endpoint wizard by right clicking anywhere on the Business Service view and selecting **New** → **Endpoint**. In the endpoint wizard, specify the corresponding Address type, for instance, HTTP or SCA, and provide the URL of the endpoint.

We used the second option and manually created the endpoints.

To create the endpoint:

1. Right-click **ITSOPBankCBAPrj**, and select **New** → **Composite Service**. Select **ITSO\_impl** from the SCA project list, and click **Finish**. Clicking Finish imports the SCA implementation module and creates interface and endpoint information from WSDL files that are present in the ITSO\_impl module. Because the interface already exists, you will get a warning message “Interface exists” that you can ignore.

**Note:** ITSO\_impl provides dummy implementations for Credit check, Loan provider, VIN lookup, credit verification, and Rating services that the Loan process uses. Multiple endpoints are provided for each service.

2. You will see two endpoints with the same name, as shown in Figure 9-62 on page 261, for example LegacyRatingExport\_RatingHttpPort, with one of them being read only. This is because one endpoint comes from the ISTOWSRRPrj project and the other endpoint comes from the SCA module implementation that you imported in Step 1. This is the expected behavior while you develop the SCA module (ISTO\_imp) that we sourced WSDLs from WSRR and provided the implementation.

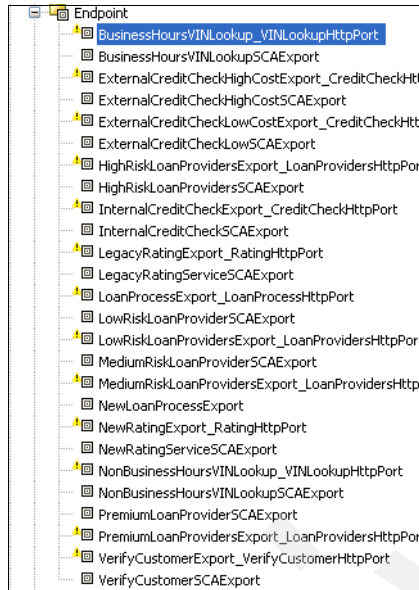


Figure 9-62 End point meta data

**Note:** If you double-click any of the endpoints, the endpoint editor opens. Click the Interfaces tab to see the interfaces that the endpoint supports. Multiple endpoints can support the same interface and they may vary in terms of their implementation.

After the endpoint objects are created, we need to target the endpoints to the corresponding environment. As part of the ITSO\_impl, we provided dummy implementations for production and development environments. For development, we used an SCA implementation, and for production we used a Web service implementation. The endpoints that contain HTTP are for the production environment, while endpoints that contain SCA are for the development environment. Based on which environment your application is running, the corresponding environment is picked at runtime by providing the right environment context.

### ***Associating environments to endpoints***

Next, we add the environment to all of the endpoints. For endpoints ending in HTTP, (which are not marked as read-only) we add the production environment. For endpoints that contain SCA, we add the development environment:

1. To add the environment and endpoint to BusinessHoursVINLookupSCAExport, double-click

**BusinessHoursVINLookupSCAExport.** The editor opens up. Click the **Environments** tab, and click the **Add Existing** button, as shown in Figure 9-63. Select the **Development** environment, and click **OK**. Save the editor.

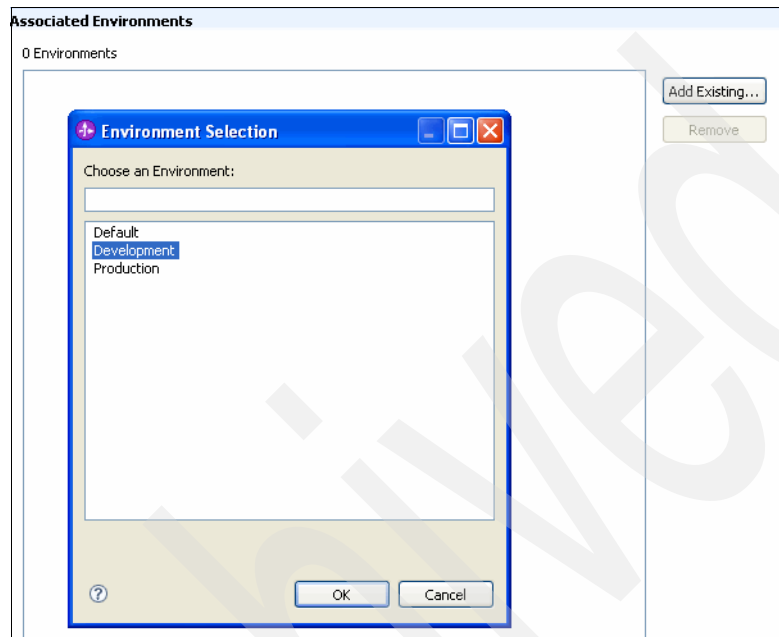


Figure 9-63 Development endpoint

The environment that is shown in the environment section list comes from the environments that we created in Configure Environments in the Governance Manager.

**Note:** Fabric, at runtime, uses only those endpoints and policies that are associated with the environment. You can specify the environment as one of the context parameters that is used at runtime to pick up an endpoint that is associated with that environment.

2. Target the remaining endpoints that contain SCA to the development environment and endpoints that contain HTTP (which are not marked as read-only) to the production environment.

## 9.5.8 Adding capabilities to endpoints

In this section, we target capabilities through endpoints by adding assertions. Each endpoint specifies its capabilities, such as how much response time it supports, what is the availability of the endpoints, and what is operation time.

For the ITSOPBank scenario, let us look at Table 9-6, which shows what assertions need to be applied.

Table 9-6 Capabilities for endpoints

Interfaces	Endpoint	Capability	Assertion applied to endpoints
CreditCheck	ExternalCreditCheckHighCostSCAExport and ExternalCreditCheckHighCostExport_CreditCheckHttpPort	Costs 10\$for credit check	Credit Check Assertion with value 10 with Required option checked.
CreditCheck	ExternalCreditCheckLowSCAExport and ExternalCreditCheckLowCostExport_CreditCheckHttpPort	Costs 10\$for credit check	Credit Check assertion with value 5 with Required option checked.
CreditCheck	InternalCreditCheckSCAExport and InternalCreditCheckExport_CreditCheckHttpPort	Costs 2\$for credit check	Credit Check assertion with value 2 with Required option checked
VINLookup	BusinessHoursVINLookupSCAExport and BusinessHoursVINLookup_VINLookupHttpPort	Supports Business Hours:9:00am to 5:00 pm from. Monday to Friday	Hours of Operation with Required option checked
VINLookup	NonBusinessHoursVINLookupSCAExport and NonBusinessHoursVINLookup_VINLookupHttpPort	Supports Non Business Hours:5:00pm to 9:00 am from Monday to Friday and weekends.	Hours of Operation with Required option checked
LoanProviders	LowRiskLoanProviderSCAExport and LowRiskLoanProvidersExport_LoanProvidersHttpPort	Serves Low Risk loans	Rating Score Assertion with value Low with Required option checked.

Interfaces	Endpoint	Capability	Assertion applied to endpoints
LoanProviders	MediumRiskLoanProviderSCAExport and MediumRiskLoanProviderSExport_LoanProvidersHttpPort	Serves Medium Risk loans	Rating Score Assertion with value Medium with Required option checked
LoanProviders	PremiumLoanProviderSCAExport and PremiumLoanProvidersExport_LoanProvidersHttpPort	Serves Premium bank customers with Low credit rating	Rating Score Assertion with value Low with Required option checked and Customer Type Assertion with value Premium with Required option checked.
LoanProviders	HighRiskLoanProvidersSCAExport and HighRiskLoanProvidersExport_LoanProvidersHttpPort	Serves High Risk loans	Rating Score Assertion with value High with Required option checked
VerifyCustomer	VerifyCustomerSCAExport and VerifyCustomerExport_VerifyCustomerHttpPort	Supports verification for ITSO Bank customers	Bank Identifier with value of ITSO Bank with Required option checked
Rating	LegacyRatingServiceSCAExport and LegacyRatingExport_RatingHttpPort	Supports Regular or Premium customer and will be operation till May 15th. After May 15th the New Rating service needs to be invoked	Deprecation Assertion set with deprecation date Required option <b>not</b> checked and Supports version as 1 with Required option checked.
Rating	NewRatingServiceSCAExport an NewRatingExport_RatingHttpPort	Supports All customers	Supports version as 2 with Required option checked.

**Note:** There are two endpoints that are provided in Table 9-6 for each of the capabilities: one for development and one for production. Based on the environment context that is supplied at runtime, the Dynamic Assembler picks the required endpoint that is targeted to a development or production environment.

With the capabilities defined, the next task is to add capabilities to the endpoints.

## Adding capabilities to the Credit Check endpoint

To add capabilities to the Credit Check endpoint:

1. Double-click **ExternalCreditCheckHighCostSCAExport**. The editor opens up.
2. Click the **Assertions** tab, and select **Add**. On the Assertion Type dialog box, expand **Interoperability Assertion**, expand **Content Based Assertion**, and select **Credit Check Cost Assertion**, as shown in Figure 9-64. Click **OK**.

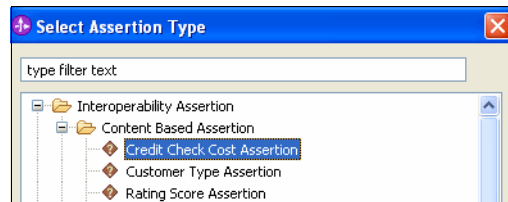


Figure 9-64 Credit check assertion

3. On the credit check assertion value dialog box, specify the value as 10, as shown in Figure 9-65. Select the **Required** option. You will see the credit check assertion being added to the Assertions tab. Save the editor.

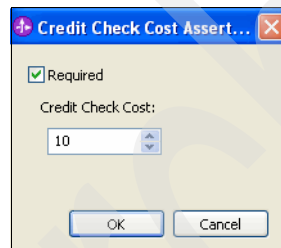


Figure 9-65 Specify cost value

**Note:** Marking the assertion as *required* implies that the endpoint is selected only if the assertion is matched by policy during policy evaluation, for example, the `ExternalCreditCheckHighCostSCAExport` is picked up only if the credit cost is equal to 10.

4. Carry out the steps for the following endpoints using the values specified in Table 9-6 on page 263:
  - `ExternalCreditCheckLowSCAExport`
  - `ExternalCreditCheckLowCostExport_CreditCheckHttpPort`
  - `ExternalCreditCheckHighCostExport_CreditCheckHttpPort`

- InternalCreditCheckSCAExport
- InternalCreditCheckExport\_CreditCheckHttpPort

5. Mark each of these assertions as **Required**.

## Adding capabilities to the VINLookup endpoints

To add capabilities to the VINLookup endpoints:

1. Double-click **BusinessHoursVINLookupSCAExport**. The editor opens up.
2. Click the **Assertions** tab, and select **Add**. On the Assertion Type dialog box, expand **Manageability Assertion**, and select **Hours of Assertion**, as shown in Figure 9-66. Click **OK**.

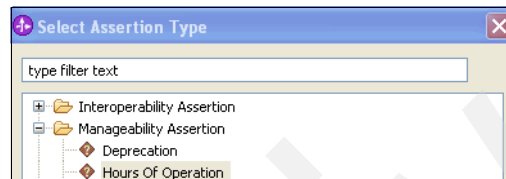


Figure 9-66 Select assertion type

3. On the Hours of Assertion value dialog box, specify the values shown in Figure 9-67. Click **OK**. Save the editor.

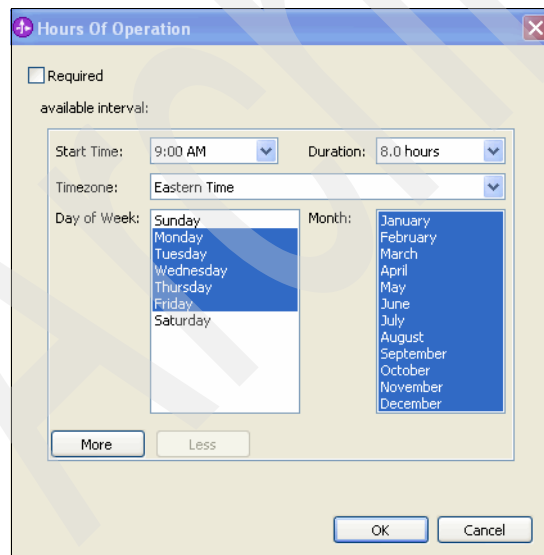


Figure 9-67 Specify hours of operation



4. Repeat the same steps for the BusinessHoursVINLookup\_VINLookupHttpPort endpoint using the values specified in Table 9-6 on page 263.
5. For the NonBusinessHoursVINLookupSCAExport and NonBusinessHoursVINLookup\_VINLookupHttpPort endpoints, specify the following hours of operation, as shown in Figure 9-68.

Start Time: 5:00 PM Duration: 16.0 hours

Timezone: Adak (United States)

Day of Week: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

Month: January, February, March, April, May, June, July, August, September, October, November, December

More Less

Figure 9-68 Specify non business hours for week days

6. We also need to supply weekend hours of operation, which you can specify by clicking **More**, and specifying the values in Figure 9-69.

Start Time: 9:00 AM Duration: 24.0 hours

Timezone: Eastern Time

Day of Week: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

Month: January, February, March, April, May, June, July, August, September, October, November, December

More Less

Figure 9-69 Specify non weekend hours

7. The Hours of operations assertion is added to the Assertions tab. Mark the assertion as required by selecting the **Required** option. Save the editor.

## Adding capabilities to the Loan Provider endpoints

To add capabilities to the Loan Provider endpoints:

1. Double-click **LowRiskLoanProviderSCAExport**. The editor opens up.

2. Click the **Assertions** tab, and select **Add**. On the Assertion Type dialog box, expand **Interoperability Assertion**, expand **Content Based Assertion**, and select **Rating Score Assertion**, as shown in Figure 9-70.

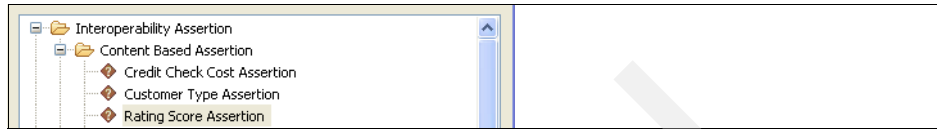


Figure 9-70 Rating score

3. On the Rating score assertion value box, specify the value as **Low**, and select the **Required** option, as shown in Figure 9-71. Save the editor.



Figure 9-71 Rating value

4. Add the assertions for the following endpoints using the values specified in Table 9-6 on page 263:
  - LowRiskLoanProvidersExport\_LoanProvidersHttpPort
  - MediumRiskLoanProviderSCAExport
  - MediumRiskLoanProvidersExport\_LoanProvidersHttpPort
  - PremiumLoanProviderSCAExport
  - PremiumLoanProvidersExport\_LoanProvidersHttpPort
  - HighRiskLoanProvidersSCAExport
  - HighRiskLoanProvidersExport\_LoanProvidersHttpPort
5. Mark each of these assertions as **Required**.

## Adding capabilities to the Verify Customer endpoints

To add capabilities to the Verify Customer endpoints:

1. Double-click **VerifyCustomerSCAExport**. The editor opens up.
2. Click the **Assertions** tab, and select **Add**. On the Assertion Type dialog box, expand **Interoperability Assertion**, expand **Content Based Assertion**, and select **Bank Identifier Assertion**. On the pop-up dialog box, select **ITSO** in the Bank Identifier Assertion dialog box, and click select the **Required** option. Save the editor.

3. Repeat the steps for the VerifyCustomerExport\_VerifyCustomerHttpPort endpoint using the values that are specified in Table 9-6 on page 263. Mark the assertion as **Required**.

## Adding capabilities to the Rating endpoints

To add capabilities to the Rating endpoints:

1. Double-click **LegacyRatingServiceSCAExport**. The editor opens up.
2. Click the **Assertions** tab, and select **Add**. On the Assertion Type dialog box, expand **Interoperability Assertion**, and add the **Supports Version**. Enter a value of 1 in the supported Version text field, as shown in Figure 9-72. Select the **Required** option, which specifies that the endpoint supports version 1.0 of the interface. Because we have two realizations of interfaces, one for Legacy and one for New Rating, we distinguish this by the Supports Version field and add a deprecation date to the Legacy endpoint. After the Deprecation Date, the NewRating endpoint is picked up.

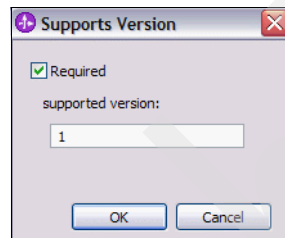


Figure 9-72 Support version assertion

3. Add the service deprecation assertion, which asserts when an endpoint is deprecated. Click **Add Assertion**, expand **Manageability Assertion**, and click **Deprecation**. On the Deprecation window, provide the values that are shown in Figure 9-73 on page 270.

Figure 9-73 Deprecation value

4. Save the editor.
5. Repeat the steps for the LegacyRatingExport\_RatingHttpPort endpoint using the values specified in Table 9-6 on page 263.
6. Repeat the steps for the NewRatingServiceSCAExport and NewRatingExport\_RatingHttpPort. Add the Support Version assertion with a value of 2 in the Supported Version text field and mark it as **Required**.

We added capabilities to the endpoints. The next step is to provide the required capabilities at runtime, which you provide by creating respective policies that select an endpoint based on the context (such as organization and user), content (such as customer type and amount) and contract (such as cost identifier and maximum transaction time) of the service request.

## 9.5.9 Creating business policies

Policies provide a set of conditions that must be true for the policy to execute. For each of the variability points, you define business policies. For the ITSOBank scenario, we defined the business policies that are specified in Table 9-7 on page 271.

Table 9-7 Business policies

Policy name	Maps to points of variability in ITSOLoan process	Target	Context	Content	Contract
ExternalCreditCheckHighLoanNonPremiumPolicy	If Loan Amount $\geq$ 20000 and Customer Type $\neq$ Premium then select a Credit rating service with Cost 10	Interface = Credit Check		Loan Amount $\geq$ 20000 and Customer Type $\neq$ Premium	Credit Check Cost Assertion =10 with Required option checked
ExternalCreditCheckNewCustomerPolicy	If Loan Amount $<$ 20000 and Customer Type = New then select a Credit rating service with Cost 10	Interface = Credit Check		Loan Amount $<$ 20000 and Customer Type = New	Credit Check Cost Assertion =10 with Required option checked
ExternalCreditCheckPremiumCustomerPolicy	If Loan Amount $\geq$ 20000 and Customer Type = Premium then select a Credit rating service with Cost 5	Interface = Credit Check		Loan Amount $\geq$ 20000 and Customer Type = Premium	Credit Check Cost Assertion =5 with Required option checked
ExternalCreditCheckRegularCustomerPolicy	If Loan Amount $<$ 20000 and Customer Type = Regular, then select a Credit rating service with Cost 5	Interface = Credit Check		Loan Amount $<$ 20000 and Customer Type = Regular	Credit Check Cost Assertion =5 with Required option checked

Policy name	Maps to points of variability in ITSOLoan process	Target	Context	Content	Contract
InternalCreditCheckPolicy	If Loan Amount < 20000 and Customer Type = Premium, then select Internal Credit Rating service with Cost 2	Interface = Credit Check		Loan Amount <20000 and CustomerType = Premium	Credit Check Cost Assertion =2 with Required option checked

Policy name	Maps to points of variability in ITSOLoan process	Target	Context	Content	Contract
LoanProvidePolicy	<ul style="list-style-type: none"> <li>▶ If Rating = High then select HighRiskLoanProvider</li> <li>▶ If Rating = Medium then select MediumRiskLoanProvider</li> <li>▶ If Rating = Low and (Customer type = New or Customer type = Regular) then LowRiskLoanProvider</li> <li>▶ if Rating = Low and Customer type = Premium then PremiumLoanProvider</li> </ul>	Interface = LoanProviders			<ul style="list-style-type: none"> <li>▶ Rating Score with Required and Fill From context checked</li> <li>▶ Customer Type Assertion with Fill From context checked</li> </ul>

Policy name	Maps to points of variability in ITSOLoan process	Target	Context	Content	Contract
RatingPolicy	<ul style="list-style-type: none"> <li>► If Customer Type= New, select New Rating service</li> <li>► If Customer Type = Premium or Regular and Date is &lt; 15th May, select Legacy Rating service</li> </ul> <p>Make sure that you set the policy end date as 15th May 2008.</p>	Interface = RatingPolicy		Customer Type != New	Support Version Assertion with Required checked and Value as 1.
VerifyCustomerPolicy	If Bank Id =ITSO, select VerifyCustomer service from ITSO Bank.	Interface = VerifyCustomer			Bank Identifier with Required and Fill from context checked.



**Note:** The **Fill from context** option causes an assertion's properties to be filled from the property values that are supplied in the context, which implies that context contains a policy assertion.

## Creating Fabric policies

We will start by creating the ExternalCreditCheckHighLoanNonPremiumPolicy policy:

1. Right-click **Policy**, and select **New** → **Policy**.
2. Select the project name as **ITSOBankCBAPrj**. Click **Browse**, and on the target dialog box, click **Interfaces** under the **Technical** node. Select **CreditCheck**, as shown in Figure 9-74. Specify the name of the policy as ExternalCreditCheckHighLoanNonPremiumPolicy. Click **Next**.

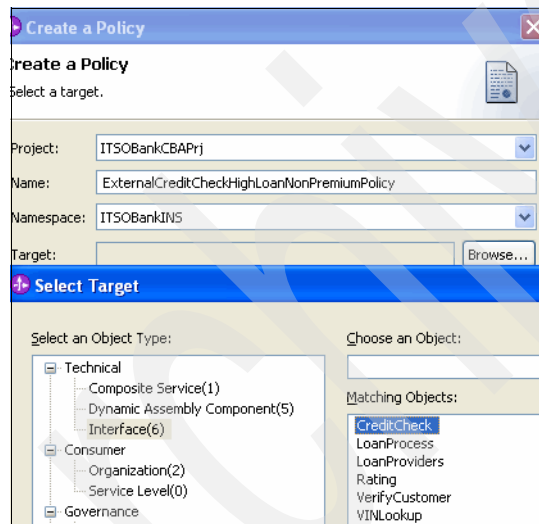


Figure 9-74 Create a policy

3. On the Policy Context window, click **Next**, and then click **Add**:
  - a. On the Content Selection dialog box, in the Content Type field, select **Loan Amount**, and in the Comparison field, select **is greater than** as shown in Figure 9-75 on page 276. Click **Add content**.

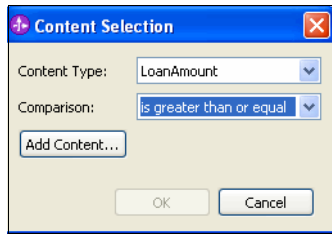


Figure 9-75 Loan amount assertion

4. On the Content dialog, in the LoanAmount field, select 20000, as shown in Figure 9-76. Click **OK**.

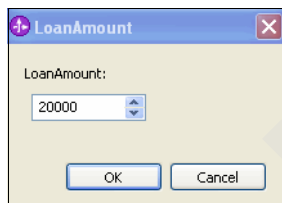


Figure 9-76 Specify content value

5. On the content selection dialog box, click **OK**. In this way you added a content selection to the policy.
6. Repeat the previous three steps for adding a Customer Type content type by selecting **Customer Type** as Content Type, Comparison as **is equal to**, and selecting Premium from CustomerType value, as shown in Figure 9-77.

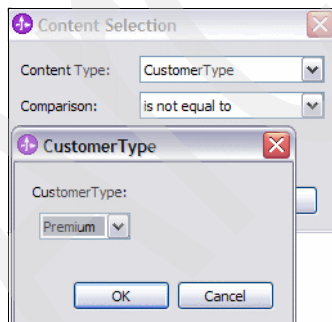


Figure 9-77 Customer Type selection

7. Click **Finish** to create the policy.

8. Next we need to add the assertions that need to be applied if the following policy condition is matched. Double-click **ExternalCreditCheckHighLoanNonPremiumPolicy**, and go to the **Assertions** tab. Click **Add**, and click **Interoperability Assertion**. Click **Content Based Assertion**, and select **Credit Check Assertion**, as shown in Figure 9-78, and click **OK**.

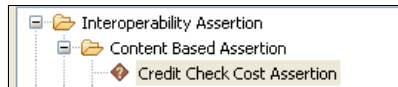


Figure 9-78 Credit Check Assertion

9. On the value dialog box, in the Credit Check Cost field, select 10, and select **Required**, as shown in Figure 9-79. Click **OK**. Save the editor.

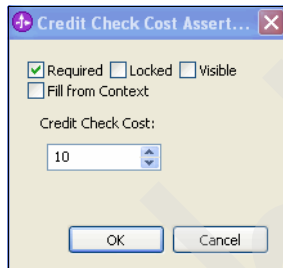


Figure 9-79 Credit cost value

With this we added a policy that states that if the context is the CreditCheck Web service, the content has a loan amount greater than or equal to 20000, and the customer type is not Premium, then apply a contract - credit check with value 10.

At runtime, if the service request contains the corresponding context and content matches the **ExternalCreditCheckHighLoanNonPremiumPolicy** policy, then the credit check assertion with value 10 is applied. The Dynamic Assembler then looks for all endpoints that satisfy the corresponding context. In this case, the context is the credit check interface, so it will look for endpoints that satisfy the credit check interface. It then looks for all endpoints that satisfy the assertion - credit check with value 10 - and will select that endpoint.

**Note:** If policy computation results in satisfying two endpoints, the one that has the least cost modifier has a preference. If the cost modifier is the same, the Dynamic Assembler throws an exception. You can see the exception while simulating the policy using policy simulation.

10. Create the policies for the following policies based on the information specified in Table 9-7 on page 271:

- ▶ ExternalCreditCheckNewCustomerPolicy
- ▶ ExternalCreditCheckPremiumCustomerPolicy
- ▶ ExternalCreditCheckRegularCustomerPolicy
- ▶ InternalCreditCheckPolicy
- ▶ LoanProvidePolicy
- ▶ RatingPolicyVerifyCustomerPolicy

### 9.5.10 Using policy simulation

We created the policies. The next step is to verify that the correct endpoints are being picked up by the policies and the policy evaluation is working fine.

Policy simulation provides design time governance that ensures that policies are working as expected. As a best practice, verify all of the policies before you submit the changes to the Business Service Repository.

**Note:** Policy simulation does not invoke the endpoint as part of the simulation. Policy simulation only tests if the right endpoint is picked up based on the evaluated policy. Make sure that your endpoint is running before you invoke the process; otherwise, you will receive an error during runtime.

The Dynamic Assembler performs the same steps as the Policy simulation with the addition of invoking the endpoint.

#### Creating a policy simulation

For each point of variability, create a policy simulation to ensure that the correct endpoints are selected as a result of policy computation. You will now create a policy simulation object for ExternalCreditCheckHighLoanNonPremiumPolicy:

1. Right-click **ITSOBankCBAPrj**, and click **New** → **Simulation**.
2. On the Dynamic Assembly Simulation window, enter the name as ExternalCreditCheckHighLoanNonPremiumPolicy, in the Namespace field, select Namespace **ITSOBankNS**, and select **CreditCheckDA** as the Dynamic Assembly component, as shown in Figure 9-80 on page 279.
3. Click **Finish**. The Policy Simulator editor opens up.

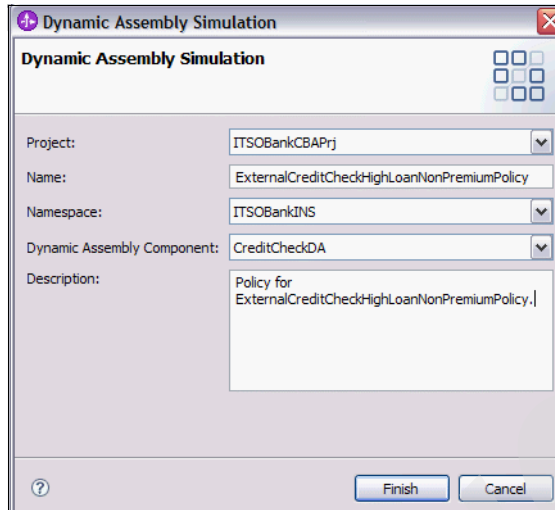


Figure 9-80 Create New Policy Simulation

4. Based on the context specification that is associated with the Dynamic Assembler component, the Required context is pre-populated; for example, CreditCheckCS is associated with CreditCheckDA and based on the CreditCheckCS context, the Environment, Customer Type, and Loan Amount context are pre populated, as shown in Figure 9-81. Also, because we selected the **Only use specified context at runtime** for the CreditCheckCS, the optional parameters are grayed out.



Figure 9-81 Required and Optional Context based on Context Specification

5. Click the **Edit** button next to CustomerType, and provide values of CustomerType as **New** or **Regular**, as shown in Figure 9-82 on page 280, and click **OK** in the Customer Type dialog box.

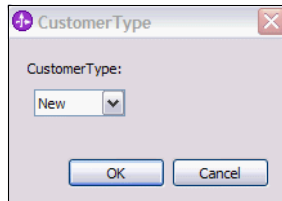


Figure 9-82 Customer Type Assertion value

6. Click the **Edit** button next to LoanAmount, and provide a value greater than 20000, say 25000, and click **OK** in the LoanAmount dialog box.
7. Click **Browse**, and in the Environment field, select **Development**. Save the editor.
8. Click **Run** to run the policy simulation:
  - a. Running the simulation shows the Results tab in the Simulator Editor, as shown in Figure 9-83. The details next to each of the step lists the results that are associated with each step.

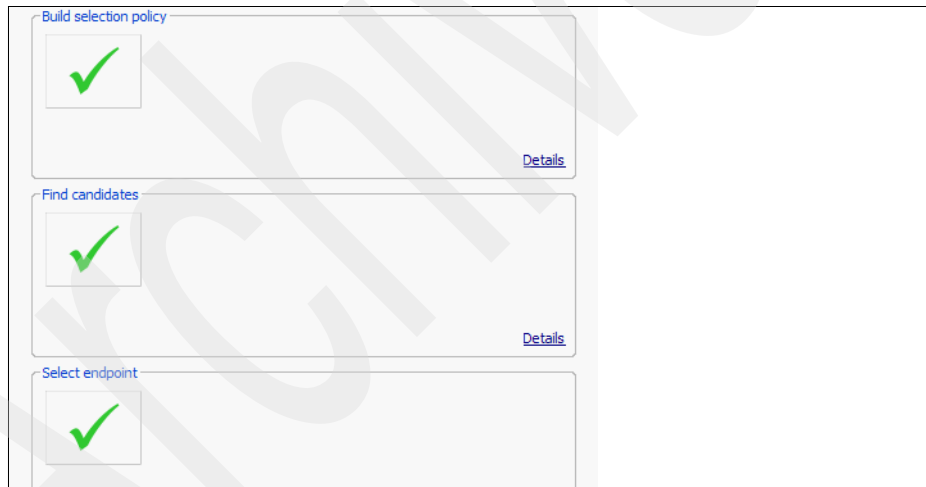


Figure 9-83 Simulation results

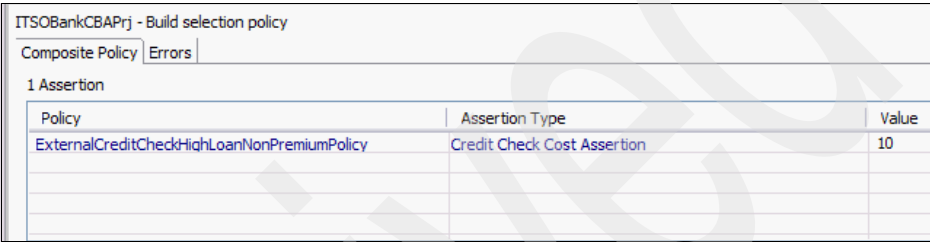
The Policy Simulator performs four operations (and by the Dynamic Assembler at runtime):

- Build Context: Builds the detail context based on the context supplied, for instance, based on the subscription ID and interface, the entire context information associated with that user can be populated like roles, business service enrolled and so on. If the required context is missing, the details

will list the missing context dimensions in a failure reasons list. In this case, as we have specified the context, this step is skipped.

- Building the selection policy based on supplied context.

The simulator lists all of the assertions in the composite selection policy along with the policy that contributed each assertion. For the ExternalCreditCheckHighLoanNonPremiumPolicy policy, if you click **Details** next to Build selection policy, you will see the ExternalCreditCheckHighLoanNonPremiumPolicy policy being selected in the Simulation operations window, as shown in Figure 9-84, with the Credit Cost Check Assertion of value 10 being applied.



ITSOBankCBAPrj - Build selection policy		
Composite Policy Errors		
1 Assertion		
Policy	Assertion Type	Value
ExternalCreditCheckHighLoanNonPremiumPolicy	Credit Check Cost Assertion	10

Figure 9-84 Selection Policy Details

Failure reasons can appear that indicate policy conflicts or policy-based guards that were triggered. To simulate this, you can create one more similar policy. This would be result in policy conflict.

- Finding candidate endpoints

In this step, the list of candidate endpoints are determined based on the selected policy. Based on interface (port type) provided in the context (this is set automatically by the Dynamic Assembler which knows which interface is being invoked at runtime), all matching endpoints are found. Then based on assertions that are defined on the selected policy, the endpoints are found whose assertion value matches the assertion defined on the selected policy. For the ExternalCreditCheckHighLoanNonPremiumPolicy policy, if you click **Details** next to Find candidates, you will see the ExternalCreditCheckHighCostSCAExport being selected in the Candidate endpoint list in the Simulation operations window, as shown in Figure 9-85 on page 282.

ITSOBankCBAPrj - Find candidates	
Candidate Endpoints	Rejected Candidates Errors
1 Candidate Endpoint	
Endpoint	Tier
ExternalCreditCheckHighCostSCAExport	1

Figure 9-85 Candidate endpoints

The ExternalCreditCheckHighLoanNonPremiumPolicy was selected as the candidate endpoint because it was associated with credit check interface and had the Credit Cost Assertion set to 10, which was required by the ExternalCreditCheckHighLoanNonPremiumPolicy policy.

The Tier 1 next to each endpoint denotes the cost that is associated with each endpoint. In endpoint metadata, there is cost field, and the one with the least cost is picked up while selecting an endpoint.

**Note:** For simulating Tier 1 data, you can create an endpoint with the same metadata as ExternalCreditCheckHighCostSCAExport and set the Cost Modifier to Day and cost as 20 in the endpoint editor.

If two endpoints have the same tier number, the Dynamic Assembler will pick up the endpoint with the lowest cost.

If you click **Rejected Candidates**, you will find two endpoints: InternalCreditCheckSCAExport and ExternalCreditCheckLowSCAExport, as shown in Figure 9-86. Both InternalCreditCheckSCAExport and ExternalCreditCheckLowSCAExport are associated with the CreditCheck interface, but did not satisfy the required contract.

Candidate Endpoints	Rejected Candidates	Errors
2 Rejected Endpoints		
Endpoint	Reason	
InternalCreditCheckSCAExport	Eliminated by Selection Policy:	
ExternalCreditCheckLowSCAExport	Eliminated by Selection Policy:	

Figure 9-86 Rejected endpoint

If any endpoints are not selected, it shows up in the rejected candidate list along with the reason.

- If there are no candidate endpoints, then the Dynamic Assembler throws an endpoint not found exception. If endpoints are not deployed in the required environment, the Dynamic Assembler throws an exception stating that no suitable endpoints were deployed to the environment.



– Selected endpoint

The policy simulator shows the selected endpoint based on candidate endpoints. For the ExternalCreditCheckHighLoanNonPremiumPolicy, if you click **Details** next to Selected endpoint, you will see the ExternalCreditCheckHighCostSCAExport being selected as the selected endpoint in the Simulation operations window, as shown in Figure 9-87.

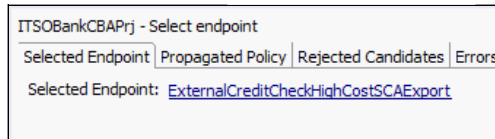


Figure 9-87 Selected endpoint

9. Any rejected candidate endpoints are also shown. The endpoints which are marked as inactive or not available for that duration (based on hours of operability assertion) show up in rejected endpoints:

**Note:** To simulate the above step, create a endpoint with same metadata as ExternalCreditCheckHighCostSCAExport and mark the status as Inactive in the endpoint editor.

- a. The propagated policy shows a list of policies that are propagated. The policy propagated is a subset of selection policy shown in the Build Selection policy. The policy is propagated as the SOAP header. The propagated policy may be used by clients receiving the message back to perform some computation.
- b. The errors pane shows an error if no endpoints were selected.

Similarly, you can create a Policy Simulation object for rest of the variability points to ensure the correct endpoints are being picked up. Table 9-8 on page 284 shows policy simulation objects that we created for the ITSOBank scenario.

Table 9-8 Policy simulation objects

Name	Dynamic Assembler component	Environment	Assertions	Endpoint result
VINLookupSimulationBusinessHours	VINLookupDA	Development	None Provide Simulation Context in editor. Date: Any week day, like Feb. 22nd Time: Business Hours (9:00 -5:00 pm), such as, 12:00 p. m.	BusinessHoursVINLookupSCAExport
VINlookupSimulationNonBusinessHours	VINLookupDA	Development	None Provide Simulation Context in editor. Date: Any week day, like Feb. 22nd Time: Non Business Hours(5:00pm to 9:00 am), such as 8:00 p. m.	NonBusinessHoursVINLookupSCAExport

Name	Dynamic Assembler component	Environment	Assertions	Endpoint result
VINlookupSimulationWeekends	VINLookupDA	Development	None  Provide Simulation Context in editor Date: Any week day, like Feb. 22nd Time: Non Business Hours(5:00pm to 9:00 am) like 8:00pm	NonBusiness HoursVINLookupSCAExport
RatingSimulationLegacy	RatingDA	Development	CustomerType = Regular Provide Simulation Context in editor Date: Any date before May 15th	LegacyRatingServiceSCAExport
LegacyRatingSimulationAfter15May	RatingDA	Development	CustomerType =Regular/New/Premium Provide Simulation Context in editor Date: Any date after May 15th	RatingServiceSCAExport
RatingSimulationNewCustomer	RatingDA	Development	CustomerType = New	NewRatingServiceSCAExport
PremiumLoanProviderSimulation	LoanProviderDA	Development	CustomerType =Premium Rating Score = Low	PremiumLoanProviderSCAExport

Name	Dynamic Assembler component	Environment	Assertions	Endpoint result
MediumRiskLoanProviderSimulation	LoanProviderDA	Development	CustomerType=New/Regular/Premium Rating Score=Medium	MediumRiskLoanProviderSCAExport
LowRiskLoanProviderSimulation	LoanProviderDA	Development	CustomerType=New/Regular Rating Score=Low	LowRiskLoanProviderSCAExport
HighRiskLoanProviderSimulation	LoanProviderDA	Development	CustomerType=New/Regular/Premium Rating Score=Low	HighRiskLoanProvidersCAExport
ExternalCreditCheckNewCustomerLessThan2000AmountSimulation	CreditCheck	Development	CustomerType=New Loan Amount=15000	ExternalCreditCheckHighCostSCAExport
ExternalCreditCheckNewCustomerGreaterThan2000AmountSimulation	CreditCheck	Development	CustomerType=New Loan Amount=25000	ExternalCreditCheckHighCostSCAExport
ExternalCreditCheckRegularCustomerGreaterThan2000AmountSimulation	CreditCheck	Development	CustomerType=Regular Loan Amount=25000	ExternalCreditCheckHighCostSCAExport
ExternalCreditCheckPremiumGreaterThan2000AmountSimulation	CreditCheck	Development	CustomerType=Premium Loan Amount=25000	ExternalCreditCheckLowSCAExport

Name	Dynamic Assembler component	Environment	Assertions	Endpoint result
ExternalCreditCheckRegularLessThan2000AmountSimulation	CreditCheck	Development	CustomerType=Regular LoanAmount=15000	ExternalCreditCheckLowSCAExport
InternalCreditCheckPremiumLessThan2000Simulation	CreditCheck	Development	CustomerType=Premium LoanAmount=15000	InternalCreditCheckSCAExport
VerifyCustomerAssertionITSPolicy	VerifyCustomer	Development	BankIdentifier=ITSO	VerifyCustomerSCAExport

**Note:** You could create similar policy simulation objects for the production environment. In that case, the end points that contain HTTP are selected instead of SCA, for instance, for the VerifyCustomerAssertionITSPolicy policy, the VerifyCustomerExport\_VerifyCustomerHttpPort endpoint is selected.

You completed the assembly of Fabric solution.

## 9.6 Summary

In this chapter, you assembled the Fabric solution in WebSphere Integration Developer by implementing the Dynamic Assembler components and extending the Dynamic Assembler by implementing various Context Extractors. Also, you learned how to inject the Fabric context that carries the required information such that the Dynamic Assembler can invoke the appropriate policies and start the required endpoint.

Next, you created the subscriber model, synchronized WSRR with Fabric to source the WSDL information in Fabric, created the ontology project that extends the Fabric ontology for custom content-based assertions, roles, and channels and created the business service project for the ITSOBank Loan process using the Fabric Administration Console.

Using the Fabric Composition Studio, you imported the SCA implementation and created the business context by defining the business service, endpoint models, context specifications and business policies and verified the policy execution by performing the policy simulation. In the next chapter, you review how to deploy and manage the Fabric solution.

# Deploying Fabric solutions

In this chapter, we describe how to deploy and test Fabric solutions and manage service enrollments and entitlements to business services.

In this chapter, you will deploy the application that we built in Chapter 9, “Assembling Fabric solutions” on page 193.

This chapter contains the following sections:

- ▶ 10.1, “Deploying the Fabric solution” on page 290
- ▶ 10.2, “Publishing the business service metadata” on page 290
- ▶ 10.3, “Deploying the SCA module” on page 295
- ▶ 10.4, “Enrollments and subscriptions management” on page 297
- ▶ 10.5, “Testing the SCA process” on page 303

## 10.1 Deploying the Fabric solution

After we choreograph the generalize the business process in WebSphere Integration Developer, create the business service metadata, and test the Policy Simulations in Composition Studio, the next step is to deploy the solution.

The business process gets deployed as enterprise archives on WebSphere Process Server, while the business services' metadata gets published to the Business Services Repository.

After the Fabric solution is deployed, you can test your Fabric solution using the Test Module functionality in WebSphere Integration Developer. In this chapter we look at how to carry out these tasks.

## 10.2 Publishing the business service metadata

You can view the metadata changes that you created for the ITSO business service project in the *Activate changes* bucket in the Repository Explorer view. The metadata changes reside in your local workspace and need to be published to the server Business Service Repository through a governance process.

**Note:** When you replicate the business service project in Composition Studio, the system creates a local Derby database in your workspace that holds your business service metadata. Each developer works on their local copy and submits the metadata changes to the Business Service Repository through a governance process. The Governance Manager resolves any conflicts by rejecting the changes by the developer if the changes consist of any stale metadata. In that case, the developer synchronizes with the Business Service Repository by updating their project and then working on it.

To publish the business service metadata:

1. Before you publish the metadata, verify that the author for each of the objects is set to ITSOBankOrg instead of the user. If not, double-click each of the objects, and change the author to ITSOBankOrg. Targeting the metadata to an organization ensures that metadata can be exported to other environments. If instead we target the objects to the user, the user must reside on those other environments as well.
2. You might also need to change the endpoint address for production environments that are based on the WebSphere port configuration. The endpoints containing the Http string are used in a production environment. Double-click each of the endpoints that contain the Http string. In the endpoint



editor, click **Protocol**, and make sure that the address is set accordingly to your WebSphere configuration. Copy the Address URL, and paste it into a browser. If you see a message “Hi there, this is a Web service”, this implies that the service is running.

3. To publish the changes, in the Repository Changes view, click the **Active Changes** node of the ITSOBANKCBAPrj project, as shown in Figure 10-1, and click **Submit Changelist**.

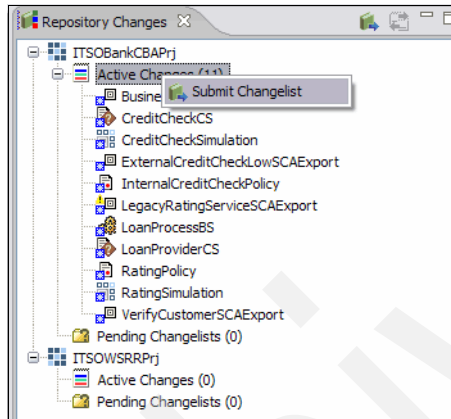


Figure 10-1 Submitting active changes

4. On the Submit changes panel, select **ITSOBANKCBAPrj**, as shown in Figure 10-2, and click **Next**.

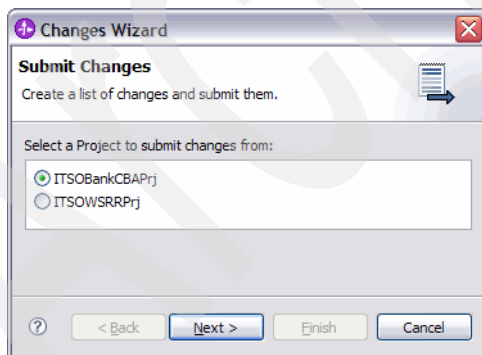


Figure 10-2 Selecting the CBA Project to submit changelist

5. On the select Changelist panel, click the **Add All** button to add all available changes in the change list, as shown in Figure 10-3 on page 292. Click **Finish**.

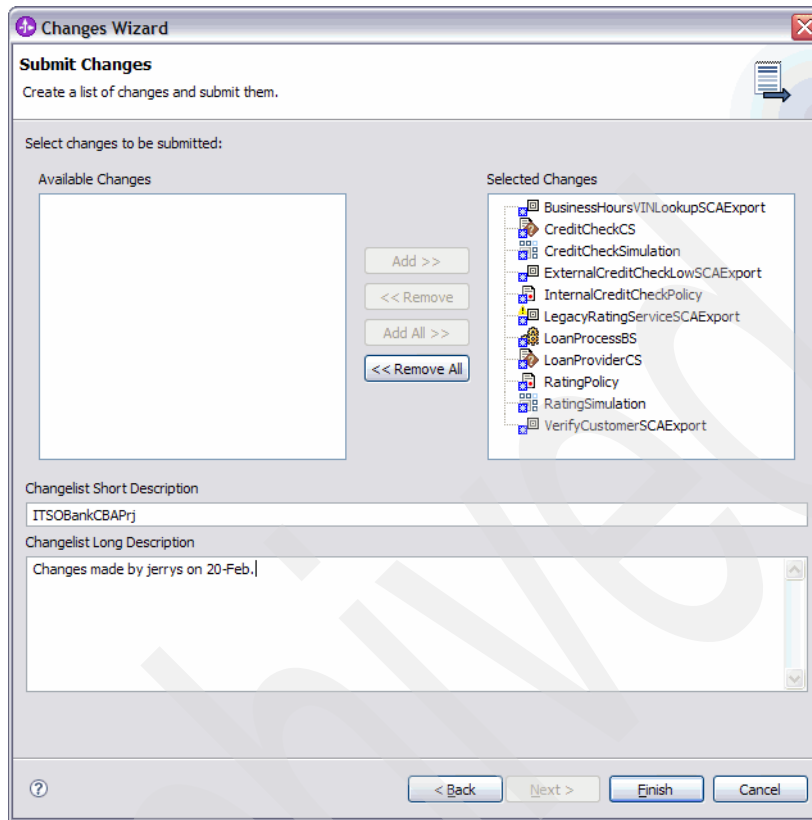


Figure 10-3 Selecting the changes to be submitted

6. Click **Submit** to submit the metadata changes to the Governance Manager, which provides you with the Governance Manager confirmation number, as shown in Figure 10-4 on page 293. You can use this number for tracking the status of your change list.

**Note:** Because we are using the Unit Test Environment, we will skip the governance process, and your metadata changes are auto approved and published to the Business Service Repository. If you were using the Foundation Pack, then a user having Administrator access needs to logon to the Fabric console and use the Governance Manager to view, approve, and publish the change list or reject it.

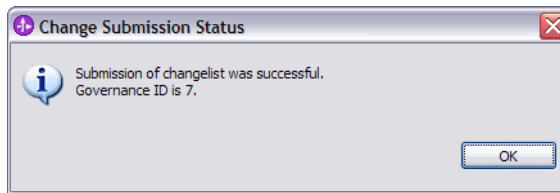


Figure 10-4 Governance Manager submission confirmation

7. After you submit the changelist, notice that your metadata that is in the changelist becomes read only (grayed out), as shown in Figure 10-5, and are batched into a Pending ChangeLists bucket.

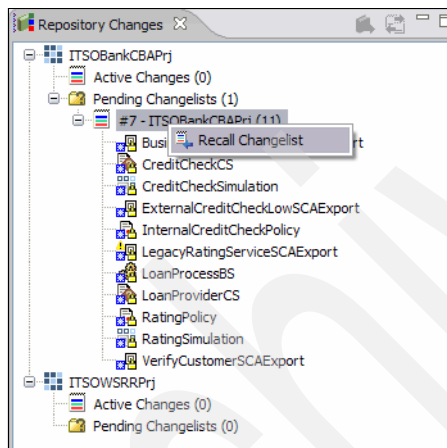


Figure 10-5 Pending changes

8. You cannot edit your changes because the changelist is in a governance process. If you need to make any changes to the submitted changelist, you can recall the change list by clicking the changelist under the Pending ChangeLists bucket, as shown in Figure 10-5, which causes the changelist to be rejected.

After the changes are published to the Business Service Repository, you can right-click the ITSOBankCBAPrj project, and click **Update**. The Update project window opens up, as shown in Figure 10-6 on page 294. Click **Finish** to update and synchronize your local copy of your Business Service Repository with the server Business Service Repository. You will see a window that provides the status of the changelist as Published and a message that says Project synchronized from version x to y.

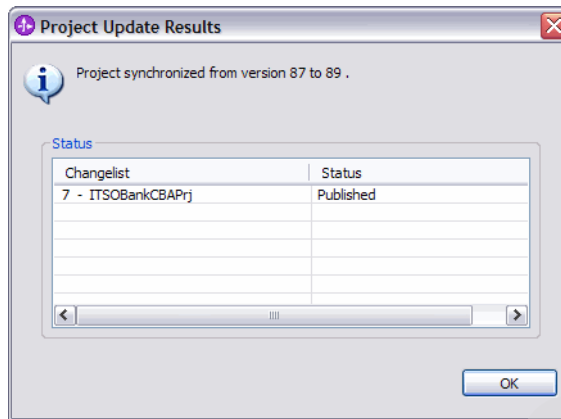


Figure 10-6 Changelist synchronization

- Logon to the Fabric Administration console, and click **My Inbox**. The following three events are displayed in the Inbox messages: Changelist submitted, Changelist accepted by Administrator, and Changelist published, as shown in Figure 10-7. In the Unit Test Environment, the governance process is skipped, and the changelist is auto approved and published. The Composition Studio user also receives e-mails about the status of the changelist after each step.



Figure 10-7 Inbox messages for changelist submission

## 10.3 Deploying the SCA module

SCA modules are deployed as enterprise archives. Switch to the Business Services perspective in WebSphere Integration Developer, and you will see the three SCA modules in your workspace, as shown in Table 10-1.

Table 10-1 Module functionary

Module name	Functionally provided
ITSOApp	Provides the Loan Process implementation.
ITSO_impl	Provides a dummy implementation of services that are used in the Loan process implementation export as SCA and Web service bindings.
WebSphereEnvUtilApp	Comprises of a WebSphereEnvUtil module and a WebSphereEnvUtilLIB library. The module is used for retrieving WebSphere environment variables, which determines where the application is running (development or production) and the Fabric subscription ID using the Fabric Subscriber SDK APIs based on username and business service.

WebSphereEnvUtil uses the Fabric Subscriber SDK APIs. Before you deploy the applications, you need to provide the correct URL of the Fabric Subscriber SDK service, based on which port your WebSphere Process Server is running.

To change the URL:

1. Under WebSphereEnvUtil, double-click **Assembly Diagram**, and click **SubscriberManagerImport1** in the Assembly Editor.
2. In the Properties window, change the address information that is associated with Binding to  
`http://localhost:yourwpsport/fabric-catalog/services/subscriberManager`
3. Save the editor and build the module again.

**Note:** You can also open the SubscriberManager WSDL in a text editor, and manually change the host name and port number.

When you type in the URL

`http://localhost:yourwpsport/fabric-catalog/services/subscriberManager` at the browser, you will see the following message, which verifies that the service is running:

Interface:

`com.webify.wsf.sdk.subscriber.remote.SubscriberManagerService`

Next we need to set the environment where the application is running. Fabric uses the environment as one of the context to determine which endpoint to select (for example development or production).

To set the environment property:

1. Logon to the WebSphere Administration console.
2. Click **Environments** → **Namespace** → **Name Space Bindings**.
3. Select scope to server by selecting **Node=xxx,Server=server1**, and click **New**.
4. Specify the Binding type as **String**, and click **Next**.
5. Type `FABRIC_ENVIRONMENT_PARAM` in the Binding identifier and Name in the Name space field.
6. Provide a String value with your Fabric development environment instance ID. To get this value:
  - a. Logon to the Fabric Administration console.
  - b. Navigate to **My Services** → **Governance Manager** → **Configure Environments**, and click **Development**.
  - c. Copy the Environment ID and paste it in the String value field for `FABRIC_ENVIRONMENT_PARAM`. Click **Next**
7. Click **Finish**. Click **Save** to store the WebSphere configuration. Restart the server.
8. To deploy the SCA modules:
  - a. Right-click **WebSphere Business Services Fabric Server**.
  - b. Click **Add and Remove projects**, and add the `ITSO_impl`, `ITSOApp`, and `WebSphereEnvUtilApp` applications by clicking **Add All**, as shown in Figure 10-8 on page 297.
  - c. Click **Finish**. The applications are deployed to the Fabric server.

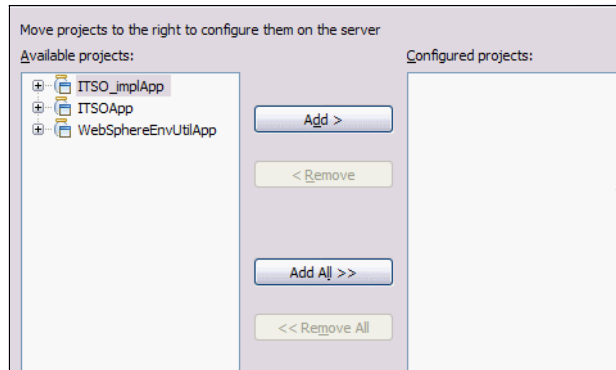


Figure 10-8 Deploy SCA module to Fabric UTE

## 10.4 Enrollments and subscriptions management

Use the following steps to enroll the Loan Process Service to the ITSOBank organization:

1. Logon to the Fabric Administration console using fabricadmin (or any userID with Administration privileges). Click **Subscriber Manager** → **Manage Subscribers**, and click **ITSOBankOrg** as shown in Figure 10-9.

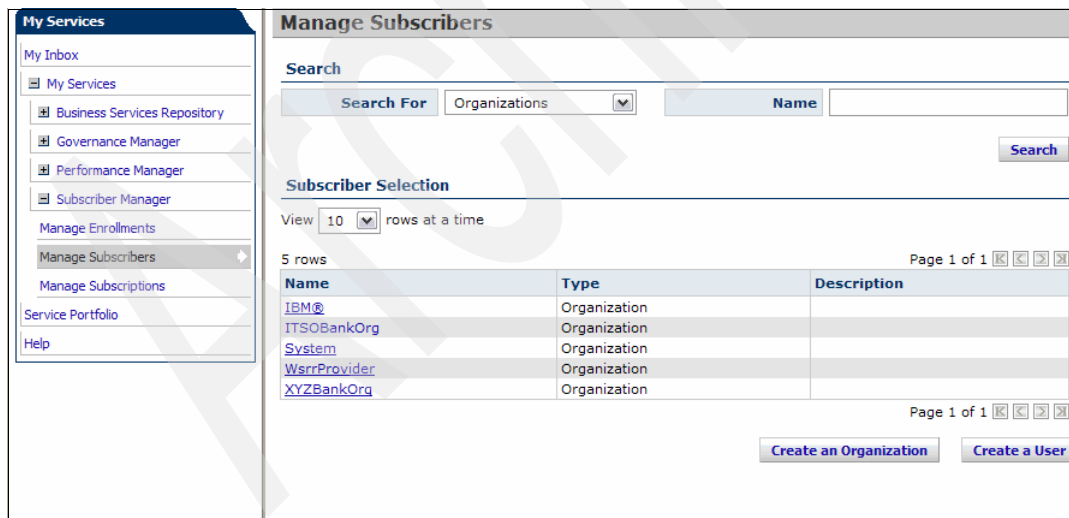


Figure 10-9 Manage subscriptions and enrollments

2. Click the **Enrollments** tab, and click **ITSOBankLoanApp**, as shown in Figure 10-10. Click **Save Enrollments**.

The screenshot shows the 'Manage Subscribers' interface for the organization 'ITSOBankOrg'. The 'Enrollments' tab is active, displaying a list of services that can be enrolled. The 'ITSOBankLoanApp' and 'LoanProcessBS' services are selected, indicated by checked checkboxes and a red highlight. The 'Save Enrollments' button is visible at the bottom right.

Name	Type
<input type="checkbox"/> Business Services Repository	Application
<input type="checkbox"/> Search Repository	Business Service
<input type="checkbox"/> Governance Manager	Application
<input type="checkbox"/> Import/Export	Business Service
<input type="checkbox"/> Configure Namespaces	Business Service
<input type="checkbox"/> Configure Environments	Business Service
<input type="checkbox"/> Configure Projects	Business Service
<input type="checkbox"/> Manage Changes	Business Service
<input type="checkbox"/> Configure Repository	Business Service
<input checked="" type="checkbox"/> ITSOBankLoanApp	Application
<input checked="" type="checkbox"/> LoanProcessBS	Business Service
<input type="checkbox"/> Performance Manager	Application
<input type="checkbox"/> Service Performance	Business Service
<input type="checkbox"/> Service Utilization	Business Service
<input type="checkbox"/> Service Invocation Summary	Business Service
<input type="checkbox"/> Subscriber Manager	Application
<input type="checkbox"/> Manage Subscriptions	Business Service
<input type="checkbox"/> Manage Enrollments	Business Service
<input type="checkbox"/> Manage Subscribers	Business Service

Figure 10-10 Enrolling Organization to Business Service

The Loan Officer and Customer roles can access LoanProcessBS. We provided this metadata while creating the process service in the Assemble phase. Only users that belong to this role can access and subscribe to the LoanProcessBS.

For the ITSO Bank scenario, the users' and group information resides in the LDAP. The next step is to source the users from the LDAP into the subscription manager.

**Note:** Subscriber manager sources only the user information from LDAP (or an external repository). There is no direct mapping from a group that is defined in LDAP to WebSphere Business Services Fabric. The user that is sourced from LDAP needs to be assigned the required Role in WebSphere Business Services Fabric.



To source the users:

1. Click **Manage Subscribers**, and click the **ITSOBankOrg** Organization. Click the **Users** tab, and click **Grant User Roles**. A page is displayed that shows all the available users in the organization, as shown in Figure 10-11.

**My Services**

- My Inbox
- My Services
  - Business Services Repository
  - Governance Manager
  - Performance Manager
  - Subscriber Manager
  - Manage Enrollments
  - Manage Subscribers
  - Manage Subscriptions
- Service Portfolio
- Help

**Manage Subscribers**

Organization: ITSOBankOrg [Back to Search](#)

Organization Details | Sub Organizations | **Users** | Enrollments

**Grant Users Roles**

Fields marked with an asterisk (\*) are required fields.

**User Selection**

Search Type: Last Name Search For:  [Search](#)

**Available Users**

- admin, admin (admin)
- Daniel, Bonnie (bdaniel)
- Campbell, David (dcampbell)
- Edwards, Doug (dedwards)
- James, Jeffrey (james)
- Smith, Joe (jsmith)
- Carlesberg, Linda (lcarlesberg)
- Burnnet, Mary (mburnnet)
- Charles, Mary (mcharles)
- Edwards, Ron (redwards)

**\* Selected Users**

**Role Selection**

**Available Roles**

- Administrator
- Agent
- Architect
- Business Analyst
- Business Architect
- Customer
- Default Role
- Developer
- Employee
- Knowledge Engineer

**\* Selected Roles**

[Grant Users Roles](#) [Cancel](#)

Figure 10-11 Grant User Roles

**Note:** The search uses the WebSphere Process Server VMM APIs to look for users.

2. In the Search Type field, select **Last Name** as the option, and type **Smith** in the Search For field. Click **Search**.
3. The Search fetches the user from the user repository and displays the user **Joe Smith** in the Available Users list. Select the user, and add it to the Selected Users section by clicking the arrow (->) button, as shown in Figure 10-12 on page 300.

The screenshot displays the 'Manage Subscribers' interface. On the left is a 'My Services' sidebar with links like 'My Inbox', 'My Services', 'Business Services Repository', 'Governance Manager', 'Performance Manager', 'Subscriber Manager', 'Manage Enrollments', 'Manage Subscribers' (highlighted), 'Manage Subscriptions', 'Service Portfolio', and 'Help'. The main area is titled 'Manage Subscribers' and shows the 'Organization' as 'ITSOBankOrg'. Below this are tabs for 'Organization Details', 'Sub Organizations', 'Users', and 'Enrollments'. A dark blue header reads 'Grant Users Roles'. A note states: 'Fields marked with an asterisk (\*) are required fields.' Under 'User Selection', there's a 'Search Type' dropdown set to 'Last Name' and a 'Search For' field containing 'Smith'. Below the search are two panels: 'Available Users' (empty) and '\* Selected Users' (containing 'Smith, Joe (jsmith)'). Between these panels are right and left arrow buttons.

Figure 10-12 Selecting user to grant role

4. From the Available Roles list, click **Customer Role**, and click the arrow (->) button to add it to the Selected Roles list. Click **Grant User Roles**, as shown in Figure 10-13 on page 301. In this process, you linked the user Joe Smith to the Customer role.

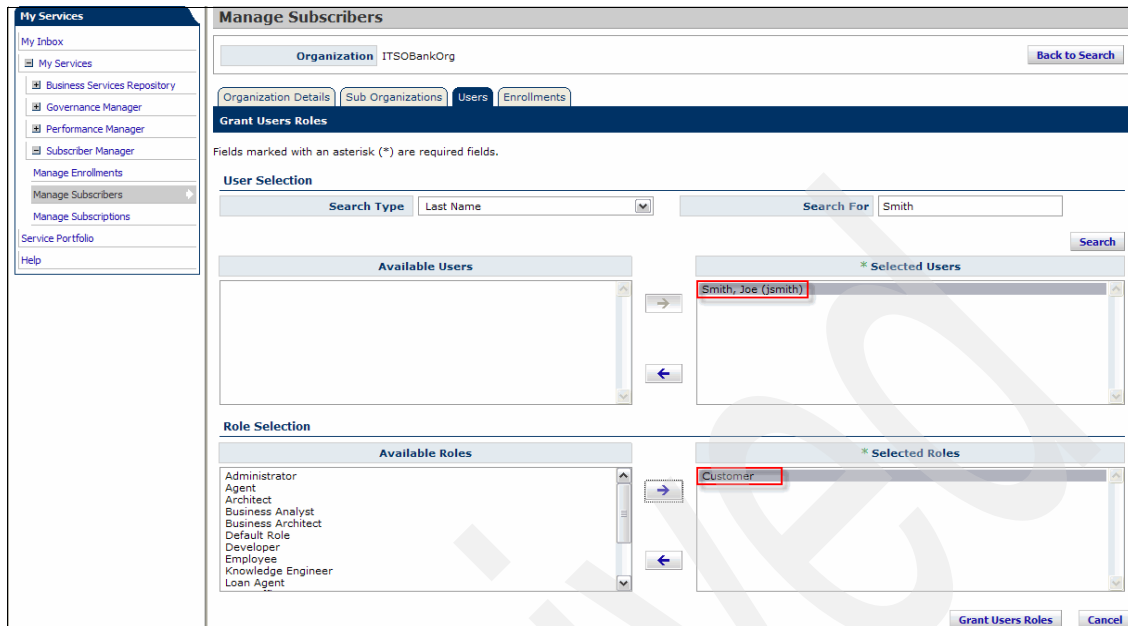


Figure 10-13 Grant user roles

Similarly, you can source other users and assign them to the Customer or Loan Officer role.

Now users have access to the business service, the next step is to subscribe users to the Loan Process Service.

To subscribe users to the Loan Process Service:

1. Click **Manage Subscribers**, and click the **ITSOBankOrg** organization. Click the **Users** tab, and on the User panel, select **Joe Smith** as the user ID.
2. On the **Subscription** tab, select the portal **LoanProcessPortalChannel** channel that is associated with the **LoanProcessBS** business service, as shown in Figure 10-14 on page 302. Click **Save Subscriptions**. There are two channels that are available for the Loan Process Service, and you selected only the required channel through which a customer can access the Loan Process Service. With this process, you subscribed to the service.

My Services

[My Inbox](#)

[My Services](#)

[Business Services Repository](#)

[Governance Manager](#)

[Performance Manager](#)

[Subscriber Manager](#)

[Manage Enrollments](#)

[Manage Subscribers](#)

[Manage Subscriptions](#)

[Service Portfolio](#)

[Help](#)

Manage Subscribers

User Joe Smith (jsmith) - joe.smith@mail.com

Back to Search

User Details

Organizations

Subscriptions

Organization Selection

Organization ITSOBankOrg

View Subscriptions

Subscription Selection

A list of Services that the current User can be Subscribed to for the selected Organization.

Name	Type
<input checked="" type="checkbox"/> ITSOBankLoanApp	Application
<input checked="" type="checkbox"/> LoanProcessBS	Business Service
<input type="checkbox"/> LoanProcessB2BChannel	Web Service Channel
<input checked="" type="checkbox"/> LoanProcessPortalChannel	Portal Channel

Save Subscriptions

Figure 10-14 User subscription to a business service

- To view the subscription ID, click **View Subscriptions**. The subscription for the Loan Process Service is displayed, as shown in Figure 10-15 on page 303. The subscription ID is a unique ID that represents a combination of a business service, role, and channel that WebSphere Business Service Fabric uses at runtime to determine what user accesses the business service and over what channel. If the subscription ID is not provided at runtime, WebSphere Business Service Fabric cannot identify the user accessing the business service, and in absence of a subscription ID, you can see any performance-related data in Performance Manager that is associated with that transaction.

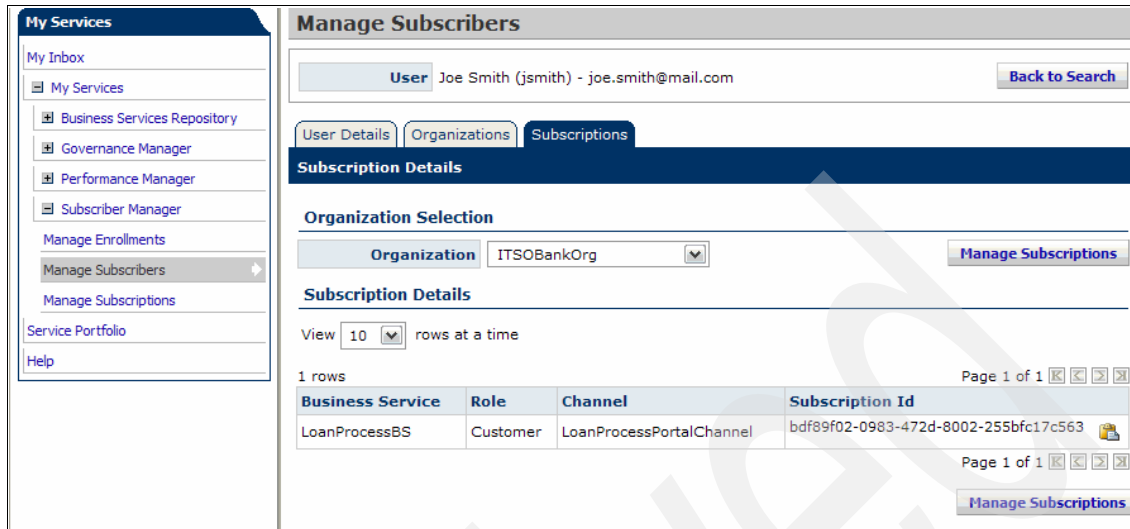


Figure 10-15 View user subscriptions

**Note:** You can retrieve subscriptions for a user and a business service through WebSphere Business Service Fabric SDK APIs. The ITSOBank process uses the WebSphere Business Service Fabric SDK APIs to retrieve the subscription ID.

## 10.5 Testing the SCA process

We use the WebSphere Business Services Fabric Unit Test Environment (UTE) server to test our application.

To test the SCA process:

1. Switch to the Business Integration perspective of WebSphere Integration Developer.
2. Open the assembly diagram for the ITSO module.
3. Right-click the assembly diagram, and select **Test Module**, as shown in Figure 10-16 on page 304.

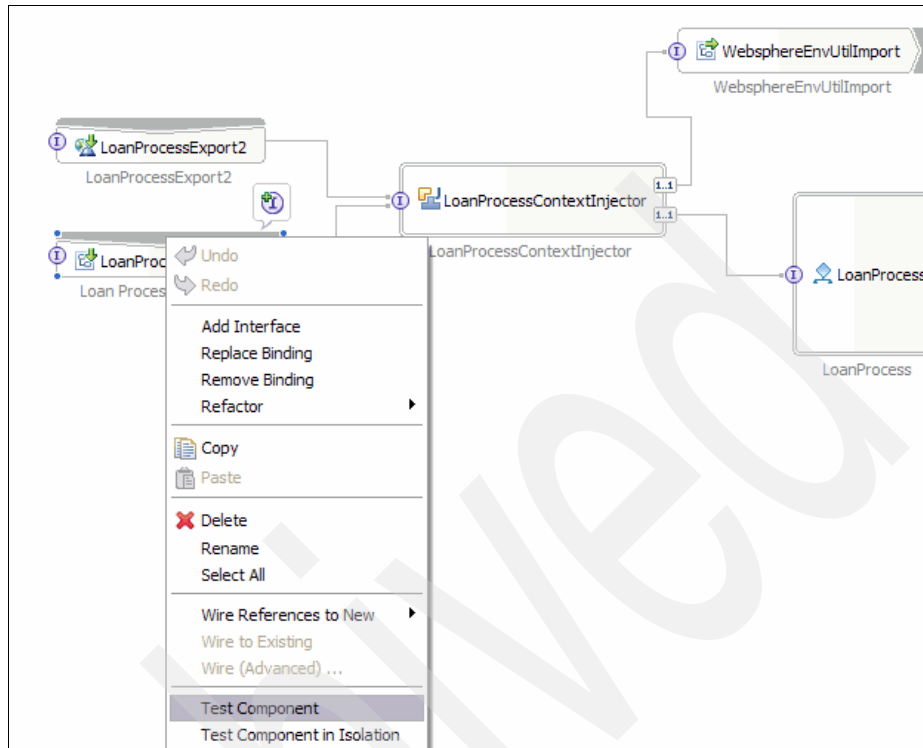


Figure 10-16 Testing the Loan Process using the Test Component

4. Provide the input data, as shown in Figure 10-17.

**General Properties**

**Detailed Properties**

Configuration: Default Module Test

Module: ITSO

Component: LoanProcessSCAExport

Interface: LoanProcess

Operation: processLoan

☒ Invoke export using binding

Initial request parameters

Name	Type	Value
Input	LoanApplication	✓
CustomerIdentific	string	✓ 100
CustomerAddress	string	✓ abc
VIN	string	✓ 12345678901234567
LoanAmountRequ	float	✓ 15000
BankId	string	✓ ITSO

Figure 10-17 Test input data

5. Click the **Continue** button. Select the **IBM WebSphere Business Service Fabric Server**. Provide the username and password as jsmith/password. jsmith is the user ID that we sourced from LDAP earlier and assigned a Customer role with access to the LoanProcessBS business service.

The sequence of the process flow is displayed. The business service should be successfully executed and return a loan response, as shown in Figure 10-18.

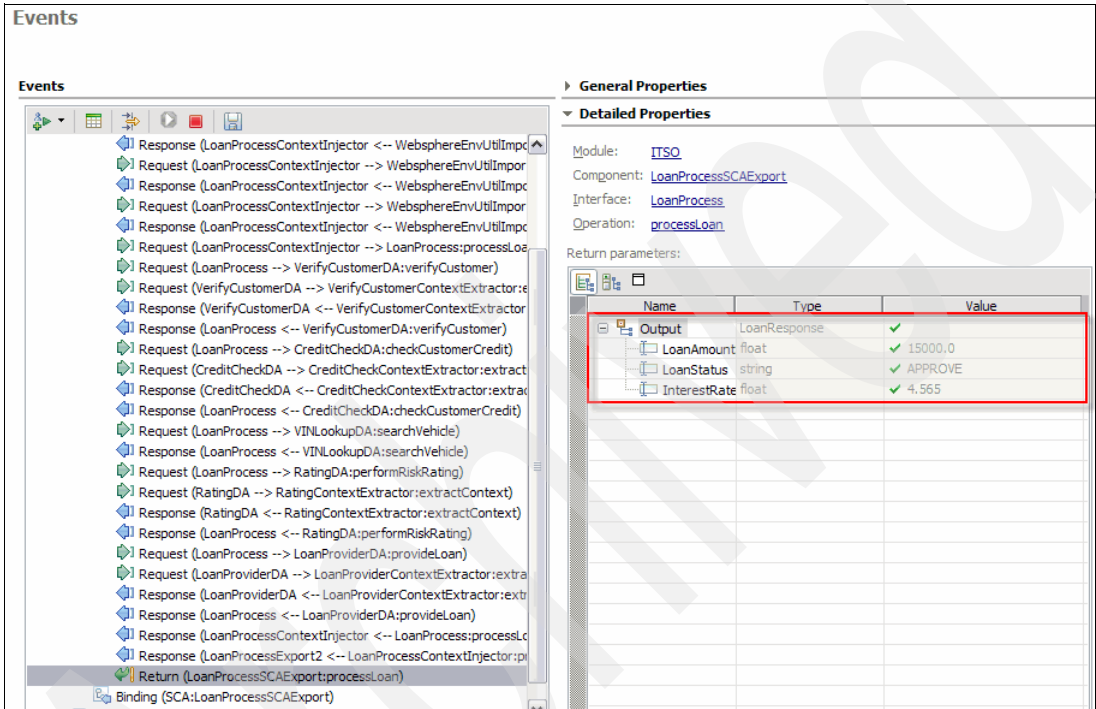


Figure 10-18 Loan Response

6. To view the performance data that is associated with the service, in the Fabric Administration Console, go to **My services** → **Performance Manager** → **Service Invocation Summary**, as shown in Figure 10-19 on page 306.

My Services

My Inbox

My Services

Business Services Repository

Governance Manager

Performance Manager

Service Invocation Summary

Service Performance

Service Utilization

Subscriber Manager

Service Portfolio

Help

Service Invocation Summary

Browse

Browse By

Application Suite

Application Suite

All

ITSOBankFinanceSuite

Duration

Today

Browse

Service Invocation Selection

View 10 rows at a time

1 rows

Page 1 of 1

Transaction ID	Service Name	Invoked On	Response Time	Channel
ID:T60FF22-2688-1203523352296-0:0	LoanProcessBS	Feb 20, 2008 4:02:52 PM	750 ms	Portal Channel

Page 1 of 1

Figure 10-19 Service Invocation Summary in Performance Manager

- Click the Transaction ID link to view the invocation that was made using the Dynamic Assembler, as shown in Figure 10-20 on page 307.



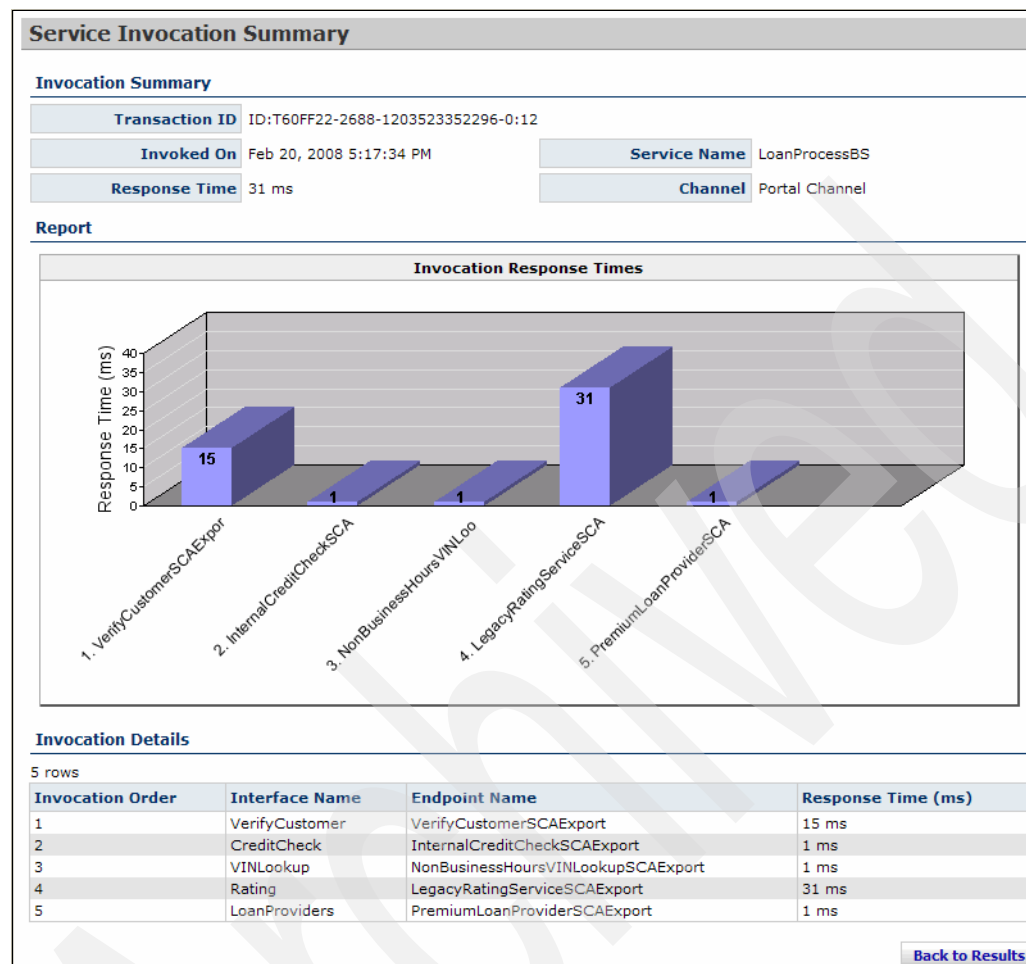


Figure 10-20 Service invocation graph with response details

You can see that five invocations were made through the Dynamic Assembler. The Dynamic Assembler picked up endpoints based on context, content, and contract that is associated with the service request.

We provided test cases for testing various scenarios and points of variability to ensure that the right implementation is picked up. The test cases are provided in as.wbiextract files, which you can find in the Physical Resources view, under the ITSO module, in the test folder.

8. The .wbiexetrace files contain the sample data. Double-click the corresponding.wbiexetrace, for example, LoanProcess\_NewCustomer\_CleanVIN\_HighLoanAmount.wbiexetrace, and right-click **Invoke**, and click **ReRun**, as shown in Figure 10-21.

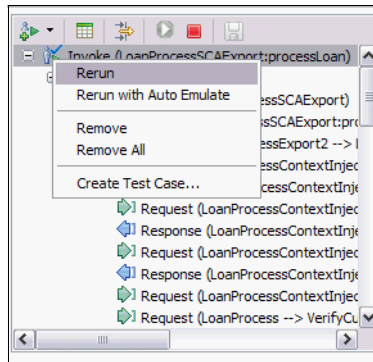


Figure 10-21 Sample wbiexetrace

The application runs with the sample data and with the Dynamic Assembler based on context, content, and contract that is associated with the service request that gets picked up at the required endpoint. Table 10-2 provides the results of invoking the various.wbiexetrace files.

9. You can view the endpoint that was picked up in the Fabric Administration console by navigating to **My services** → **Performance Manager** → **Service Invocation Summary**.

Table 10-2 Execution on .wbiexetrace files

.wbiexetrace files	Context and content provided by the service request	Endpoints picked up
LoanProcess_NewCustomer_CleanVIN_HighLoanAmount.wbiexetrace	Provides the following context <ul style="list-style-type: none"> <li>▶ Customer Type = New</li> <li>▶ Loan Amount &gt; 20000</li> <li>▶ VIN = Clean VIN</li> <li>▶ Rating= High</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckHighCostSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ NewRatingServiceSCAExport</li> <li>▶ HighRiskLoanProvidersSCAExport</li> </ul>
LoanProcess_NewCustomer_CleanVIN_LessLoanAmount.wbiexetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = New</li> <li>▶ Loan Amount &lt; 20000</li> <li>▶ VIN = Clean VIN</li> <li>▶ Rating= High</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckHighCostSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ NewRatingServiceSCAExport</li> <li>▶ HighRiskLoanProvidersSCAExport</li> </ul>

<b>.wbixetrace files</b>	<b>Context and content provided by the service request</b>	<b>Endpoints picked up</b>
LoanProcess_NewCustomer_NotCleanVIN_HighLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = New</li> <li>▶ Loan Amount &gt; 20000</li> <li>▶ Rating= High</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckHighCostSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ HighRiskLoanProvidersSCAExport</li> </ul>
LoanProcess_NewCustomer_NotCleanVIN_LessLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = New</li> <li>▶ Loan Amount &lt; 20000</li> <li>▶ Rating= High</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckHighCostSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ NewRatingServiceSCAExport</li> <li>▶ HighRiskLoanProvidersSCAExport</li> </ul>
LoanProcess_PremiumCustomer_CleanVIN_HighLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Premium</li> <li>▶ Loan Amount &gt; 20000</li> <li>▶ Rating= High</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckHighCostSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ HighRiskLoanProvidersSCAExport</li> </ul>
LoanProcess_PremiumCustomer_CleanVIN_LessLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Premium</li> <li>▶ Loan Amount &lt;20000</li> <li>▶ Rating= Low</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ InternalCreditCheckSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ PremiumLoanProviderSCAExport</li> </ul>
LoanProcess_PremiumCustomer_NotCleanVIN_HighLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Premium</li> <li>▶ Loan Amount &gt; 20000</li> <li>▶ Rating= Medium</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckLowSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ MediumRiskLoanProviderSCAExport</li> </ul>
LoanProcess_PremiumCustomer_NotCleanVIN_LessLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Premium</li> <li>▶ Loan Amount &lt;20000</li> <li>▶ Rating= Medium</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ InternalCreditCheckSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ MediumRiskLoanProviderSCAExport</li> </ul>

<b>.wbixetrace files</b>	<b>Context and content provided by the service request</b>	<b>Endpoints picked up</b>
LoanProcess_RegularCustomer_CleanVIN_HighLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Premium</li> <li>▶ Loan Amount &lt;20000</li> <li>▶ Rating= Low</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckLowSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ PremiumLoanProviderSCAExport</li> </ul>
LoanProcess_RegularCustomer_CleanVIN_LessLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Regular</li> <li>▶ Loan Amount &lt;20000</li> <li>▶ Rating= Medium</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckLowSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ MediumRiskLoanProviderSCAExport</li> </ul>
LoanProcess_RegularCustomer_NotCleanVIN_HighLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Regular</li> <li>▶ Loan Amount &gt;20000</li> <li>▶ Rating= High</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckHighCostSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ HighRiskLoanProvidersSCAExport</li> </ul>
LoanProcess_RegularCustomer_NotCleanVIN_LessLoanAmount.wbixetrace	Provides the following context: <ul style="list-style-type: none"> <li>▶ Customer Type = Regular</li> <li>▶ Loan Amount &lt;20000</li> <li>▶ Rating= Medium</li> </ul>	<ul style="list-style-type: none"> <li>▶ VerifyCustomerSCAExport</li> <li>▶ ExternalCreditCheckLowSCAExport</li> <li>▶ BusinessHoursVINLookupSCAExport</li> <li>▶ LegacyRatingServiceSCAExport</li> <li>▶ MediumRiskLoanProviderSCAExport</li> </ul>

As shown in Table 10-2 on page 308, The Dynamic Assembler picks up the required endpoint based on the context and content that the service request provided and the contract that is applied based on policy execution.

The process that is modeled with Fabric is more generalized and can handle incremental changes without you modifying and deploying the process. You can add new endpoints and context, and you can create business policies in Fabric to handle incremental changes.

In Chapter 12, “Adapting to changing business needs” on page 325 you will look at how to manage business changes with Fabric without making process modifications.

Archived



## Migrating Fabric projects across environments

After you develop your Fabric solution in the Fabric Unit Test Environment (UTE), as described in Chapter 10, “Deploying Fabric solutions” on page 289, the next step is to migrate your Fabric solution to a production-ready environment. In this chapter, we describe how to migrate a Fabric solution, in general, across environments.

Migrating a Fabric solution consist of two steps:

- ▶ Migrating the business process implemented as the SCA module
- ▶ Migrating the WebSphere Business Services meta data

## 11.1 Environment setup overview

Before promoting the Fabric project to a production (or any) environment, let us look at a typical setup for a standalone WebSphere Business Services Fabric environment. For the ITSOBank scenario, we used the Foundation Pack Topology, as described in 6.2, “Typical deployment topology for the Business Services Foundation Pack” on page 100.

Use the following steps to configure this environment:

1. Install WebSphere Process Server V6.1, and create a profile with security enabled.
2. Install Foundation Pack on the profile.
3. Install WebSphere Services Registry and Repository (WSRR) on that profile. We could use the same WSRR database that we used with the UTE environment. If you plan to use the same WSRR database, select **Use existing database** when the system prompts you for database configuration as part of the WSRR installation. We created a new WSRR database as part of the ITSO production installation.
4. Integrate the profile with LDAP using WebSphere VMM. We can use the same LDAP server that we used for the UTE environment or integrate with another LDAP server. For the ITSOBank, we used the same LDAP server.
5. Logon to the Fabric console using the admin user ID (we used fabricadmin). If you can view the subscriptions page after setup, that implies that Foundation Pack is successfully configured. For troubleshooting, refer to 6.6, “Verifying the Business Services Foundation Pack installation” on page 111.

**Note:** When you create business service metadata in Composition Studio, by default, the author of the metadata is set to the user who is logged into Composition Studio.

If you export this business service data to other environments and if the user is not available on that environment, you cannot import the business service data. In that case, it is ideal to set the author of the metadata to the organization, as we did for the ITSO Bank scenario, and export the organization's project when moving across environments. We discuss how to perform this process in this chapter.



## 11.2 Migrating the business process

Migrating the business process involves exporting the SCA module as an EAR file and deploying the enterprise archive on the WebSphere Process Server environment.

In a development environment, you typically use dummy service realizations that are based on actual production (or third-party) interfaces. Using the Dynamic Assembler to assemble the services that you had, also removes any static bindings of services to your processes; therefore, you do not need to change your business process to add services (endpoint) address references that are targeted to the production environment. Based on the environment context, the Dynamic Assembler can find suitable policies and endpoints that are targeted to an environment.

To migrate the process:

1. Export the following SCA projects as an enterprise archive by clicking **SCA projects** and selecting **Export** → **J2EE** → **Ear file**:
  - ITSO
  - ITSO\_Impl
  - WebSphereEnvUtil
2. For the third-party service references that the **LoanProcessBS** business service uses, we provided dummy Web-service implementations.
3. Deploy the exported enterprise archives (EAR) files through the Integrated Solution Console.

**Note:** In the development Fabric, UTE environments used SCA bindings for the third-party services implementation, while in production we use Web services binding.

4. To verify if the implementations are running, open a browser, and type in the one of the URLs in Table 11-1 on page 316, which contains the list of endpoints and corresponding URLs for the loan process. You should receive a message Hi there, this is a Web Service, as shown in Figure 11-1 on page 316.

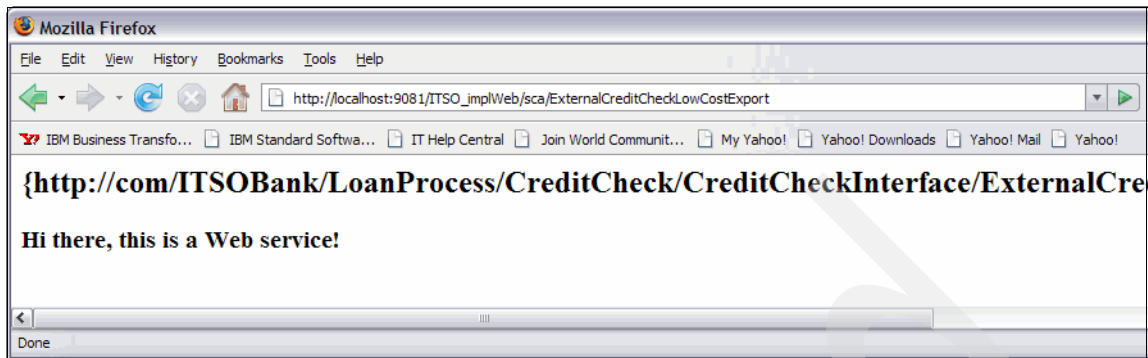


Figure 11-1 Verifying the Web Service deployment

**Note:** In Table 11-1, the {http\_port} that is mentioned in the URL is the port on which the default host is available. You can get the port address from:

**Server Instance → Communications → Ports → WC\_defaulthost**

Table 11-1 Endpoints and corresponding binding URLs

Endpoints	Endpoint URLs
VerifyCustomerExport_VerifyCustomerHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/VerifyCustomerExport
InternalCreditCheckExport_CreditCheckHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/InternalCreditCheckExport
ExternalCreditCheckLowCostExport_CreditCheckHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/ExternalCreditCheckLowCostExport
ExternalCreditCheckHighCostExport_CreditCheckHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/ExternalCreditCheckHighCostExport
BusinessHoursVINLookupExport_VINLookupHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/BusinessHoursVINLookupExport
NonBusinessHoursVINLookupExport_VINLookupHttpPort	http://localhost:{http_port}/ITSO_libWeb/sca/NonBusinessHoursVINLookupExport
LegacyRatingExport_RatingHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/LegacyRatingExport
NewRatingExport_RatingHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/NewRatingExport

Endpoints	Endpoint URLs
PremiumLoanProvidersExport_LoanProvider sHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/Premium LoanProvidersExport
MediumRiskLoanProvidersExport_LoanProvi dersHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/MediumR iskLoanProvidersExport
LowRiskLoanProvidersExport_LoanProvider sHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/LowRisk LoanProvidersExport
HighRiskLoanProvidersExport_LoanProvider sHttpPort	http://localhost:{http_port}/ITSO_implWeb/sca/HighRisk LoanProvidersExport

### 11.2.1 Configuring environment variables

To determine in which environment the application is deployed, we configured an environment variable.

To configure an environment variable for the production environment:

1. Logon to the Integrated Solutions Console of WebSphere Process Server as an administrator.
2. Click **Environments** → **Namespace** → **Name Space Bindings**.
3. Select the scope to server by selecting **Node=xxx,Server=server1**. Click **New**.
4. Specify the Binding type as **String**, and click **Next**.
5. In the Binding identifier and Name in name space field, type `FABRIC_ENVIRONMENT_PARAM`, and provide a string value with your Fabric development environment instance ID. To get this value:
  - a. Logon to the Fabric Administration console.
  - b. Navigate to **My Services** → **Governance Manager** → **Configure Environments**, and click **Production**.
  - c. Copy the Environment ID and paste it in the String value field for `FABRIC_ENVIRONMENT_PARAM`. Click **Next**.
6. Click **Finish**, and click **Save** to store the WebSphere configuration. Restart the server.

The application uses this configuration to provide the environment context to the Fabric Dynamic Assembler at runtime.

Our next step is to promote the business service metadata.

## 11.3 Promoting business service metadata

Promoting the business service metadata consists of exporting the business service metadata from one environment and importing it back to another environment. For the ITSOBank scenario, we promote business service metadata from the Fabric UTE out to the production environment.

### 11.3.1 Exporting the metadata from UTE

Use the following steps to promote the business service metadata:

1. Logon to the UTE Fabric Console and carry out the following steps.

**Note:** In a multi user environment where developers are working on assembling business service metadata, create the environment instances first, and then share them across the environments, so that each of the developers work on the same environment instances. It is similar to creating a master set of environments and using the master set across environments. If two developers create an environment named development in their own Fabric UTE server, and if one imports the other environment instances, they get two development environments in their UTE server because the unique identifier for each of them is different. Before you start a Fabric development, it is a good practice to create a master set of environments and use it across environments.

2. Export the environment instances:
  - a. Logon to the UTE Fabric console, and go to **Governance Manager** → **Import/Export**, and select the **Export by Project** tab.
  - b. From the Project pull-down list, select **Fabric Governance**, and in the Namespace section, click **Environments**.
  - c. Click **Export to File**, as shown in Figure 11-2 on page 319. Save the file FabricGovernancexxx-owl.zip to any folder (for example c:\production).

<div> <div>* Content Type</div> <div>Full Content</div> </div>			
Project Selection			
<div> <div>* Project</div> <div>Fabric Governance</div> </div>			
Namespace Selection			
4 rows			
<input type="checkbox"/>	Name	Type	Description
<input type="checkbox"/>	Default Fabric Configuration	Instance	Provides objects used in default configuration of WebSphere Business Services Fabric. Since this namespace is provided on all system deployments, there is no need to export this namespace
<input checked="" type="checkbox"/>	Environments	Instance	User-created environments are stored in this namespace. This namespace has to be exported to other WebSphere Business Services Fabric installations when exporting content that refers to these environments.

Figure 11-2 Export environment

3. Export the ITSBankOntPrj project:

- To export the environment instance, logon to the UTE Fabric console, and go to **Governance Manager** → **Import/Export**, and select the **Export by Project** tab.
- From the Project pull-down list, select **ITSBankOntPrj**, and in the Namespace Selection section, click **ITSOBankSchemaNS** and **ITSOBankSchemaExtN**.
- Click **Export to File**, as shown in Figure 11-3, and save the file ITSBankOntPrjxxx-owl.zip to any folder, for example c:\production.

fields marked with an asterisk (\*) are required fields.

Content Type			
<div> <div>* Content Type</div> <div>Full Content</div> </div>			
Project Selection			
<div> <div>* Project</div> <div>ITSOBankCBAProj</div> </div>			
Namespace Selection			
2 rows			
<input checked="" type="checkbox"/>	Name	Type	Description
<input checked="" type="checkbox"/>	ITSOBankEnrollINS	Enrollment	
<input checked="" type="checkbox"/>	ITSOBankINS	Instance	

Figure 11-3 Export Ontology project

4. Export the ITSOBankCBAPrj Project:
  - a. Logon to the UTE Fabric console, and go to **Governance Manager** → **Import/Export**, and select the **Export by Project** tab.
  - b. From the Project field pull-down menu, select **ITSOBankCBAPrj**. In the Namespace Selection section, click **ITSOBankINS** (Instance type), and click **Export to File**, as shown in Figure 11-4, and save the file ITSOBankCBAPrj-owl.zip to any folder, for example c:\production.

**Note:** The enrollment namespace consists of enrollment and subscription information. The users on the production environment are different, and if you export the enrollment namespace, you cannot import the ITSOBankCBAPrj because the user might not exist on the production system. You do not need to export the enrollment namespace, and create the enrollments and subscription on the production environment.

Fields marked with an asterisk (\*) are required fields.

**Content Type**

\* **Content Type** Full Content

**Project Selection**

\* **Project** ITSOBankCBAPrj

**Namespace Selection**

2 rows

<input type="checkbox"/>	Name	Type	Description
<input type="checkbox"/>	ITSOBankEnrollINS	Enrollment	
<input checked="" type="checkbox"/>	ITSOBankINS	Instance	

Figure 11-4 Export CBA Project

5. Export organization, roles, and users:
  - a. To export the environment instance, log on to the UTE Fabric console and go to **Governance Manager** → **Import/Export**, and select the **Export by Project** tab.
  - b. In the Project field, select **Organization, Users and Roles**, and from the Namespace selection list, click **Organizations** → **Export to File**. Save the file OrganizationsUsersandRolesxxx.zip to any folder, for example c:\production.

**Note:** We only export the organization instances. The users in a UTE environment might be different in production environments. If the users are the same in both of the environments, you can export the external users and roles that are associated with it.

### 11.3.2 Importing the business service metadata into the production environment

The next thing we need to do is to import the FCA archives into the required environment.

To import the business service metadata:

1. Logon to the Production Fabric Console.
2. Import the Environment Instances:
  - a. Go to **Governance Manager** → **Import/Export**.
  - b. On the **Import** tab, select **Browse**, and select FabricGovernancexxx-owl.zip from the c:\production folder, as shown in Figure 11-5 on page 322.
  - c. Click **Import File**. After importing the FCA, you should see the following message: The Fabric Content Archive, FabricGovernancexxx-owl.zip, was imported successfully.

**Note:** Importing the FCA replaces all existing namespaces in that namespace, which means that if you upload an Ontology extension with two extensions and then later you add one more, you need to import all the three extensions.

With the exception of certain namespace-like environments, organization, and system instances, the application merges the namespace rather than replace it.

Fields marked with an asterisk (\*) are required fields.

**Info**

- Warning: Importing a Fabric Content Archive replaces all of the existing content.

**Fabric Content Archive Selection**

\* File c:\production\Fabric... Browse...

Figure 11-5 Importing business service metadata

3. Import the ITSO Bank Ontology project:
    - a. Click **Governance Manager** → **Import/Export**, and on the **Import** tab, select **Browse**.
    - b. Select ITSOBankOntPrjxxx-owl.zip from the c:\production folder.
    - c. Click **Import File**. You should receive a successful import message.
  4. Import the organizations, roles, and users into FCA:
    - a. Go to **Governance Manager** → **Import/Export**.
    - b. On the **Import** tab, select **Browse**, and select the OrganizationsUsersandRolesxxx.zip from the c:\production folder.
    - c. Click **Import File**.
  5. Create a WSRR Federation project, and provide the same namespace settings (source name and sponsored namespace) in Federation settings that you provided in Fabric UTE environment. For our scenario, we used the same WSRR database. If you setup a new database, first import the WSDLs that the ITSOBank Loan process uses.
- Note:** When you export a WSRR project, the federating setting metadata is not exported.
6. After the WSRR synchronization completes, verify that the interface metadata was created:
    - a. Navigate to **Business Services Repository** → **Search Repository**.
    - b. Select only **Interfaces**.
    - c. On the Resources tab, click **Search**. You should see seven rows, as shown in Figure 11-6 on page 323. With the exception of Event Consumer, the rest of the interface metadata is created by sourcing the WSDLs from WSRR.



<input checked="" type="checkbox"/> Interfaces	
<input type="checkbox"/> Composite Services	
<input type="checkbox"/> Endpoints	

Resource Selection	
View	10 rows at a time
7 rows	
Name	Type
<a href="#">CreditCheck</a>	WSDL Interface
<a href="#">Event Consumer</a>	WSDL Interface
<a href="#">LoanProcess</a>	WSDL Interface
<a href="#">LoanProviders</a>	WSDL Interface
<a href="#">Rating</a>	WSDL Interface
<a href="#">VerifyCustomer</a>	WSDL Interface
<a href="#">VINLookup</a>	WSDL Interface

Figure 11-6 Viewing interfaces' data from WSRR

7. Import the ITSOBANKCBAPrj project:
  - a. Go to **Governance Manager** → **Import/Export**.
  - b. On the **Import** tab, select **Browse**.
  - c. From the c:\production folder, select ITSOBANKCBAPrj-owl.zip, and click **Import File**. You should receive a successful import message.

**Note:** The ITSOBANKCBAPrj has a dependency on the WSDL metadata that is sourced from WSRR. The WSDL metadata belongs to a namespace, and if you provided a different namespace while creating the WSRR project, the references are not resolved, and the Import FCA will fail.

Source users from the external repository, and assign them to the ITSOBANK organization using respective roles. At this point, the users that are sourced are typically Fabric users who need to assemble the ITSO CBA Project in Composition Studio and any users who need to access the Fabric Administration console.

8. Replicate the ITSOBANKCBAPrj project in Composition Studio, and provide the production endpoint address information. Change and publish the production endpoint URL information (endpoint containing Http) on our WebSphere port settings. Because we already performed this as part of UTE, we can skip this.
9. Provide enrollments and subscriptions to the Loan Process business service.
10. Create an enrollment namespace for the ITSOBANKCBAPrj project. After the subscriptions are defined, the user can submit a loan process request.



# Adapting to changing business needs

In this chapter, we describe the changes that are required in Fabric solution to adapt to a change in the business scenario. We highlight the Fabric solution strengths in adapting to changing business needs.

Through an example, the chapter covers the steps that are needed to effectively implement the changes in the original business scenario.

We divided the chapter into the following sections:

- ▶ 12.1, “Business change” on page 326
- ▶ 12.2, “Change in the process” on page 326
- ▶ 12.3, “Managing business change” on page 328
- ▶ 12.4, “Modeling the change” on page 329
- ▶ 12.5, “Assembling the change” on page 334
- ▶ 12.6, “Deploying and testing” on page 354

## 12.1 Business change

Organizations go through changes in the business process due to many factors, for example, restructure, realignment of business, mergers, acquisitions, and so on. In this book, we take the merger and acquisitions scenario as an example to discuss the challenge of implementing business change in Fabric. The fictional change in the business scenario is due to a merger action from ITSOBank.

In this scenario, the ITSOBank buys another bank called XYZBank. After the merger, XYZBank is brought in line with the ITSOBank and adopts ITSOBank processes, except for its lending platform, which was a key strategy behind the merger.

XYZBank has a set of existing customers and its own processes to verify customers and to perform an internal credit check. So, there is still a need to use XYZBank's Verify Customer and Internal Credit Check processes for its existing customer base, particularly because the lending platform is the one upon which ITSOBank standardizes. At the same time, ITSOBank's verify customer and internal credit check services must be used for ITSOBank's existing customers. Any new customers who join after the merger can be served by the ITSOBank's services. The business challenge is to combine ITSOBank and XYZBank's processes and use the relevant service endpoints, which are based on the bank type, and observe which system becomes the system of record. For example purposes, we discuss the credit check portion of the process.

To tackle these changes in the business scenario, we need to branch off the Verify Customer and Internal Credit Search processes. The bank identifier (with a value of either ITSO or XYZ) determines the relevant process to be called, for example, if the bank identifier is ITSO, the ITSOBank Internal Credit Check service is called. If the identifier is XYZ, the XYZBank Internal Credit Check service is utilized for credit check purposes.

## 12.2 Change in the process

Due to the changes in the business scenario, we needed to modify the processes for the new merged business. Historically, implementing change at this point requires considerable effort and costs significant time and resources to implement the change. But, in a Fabric solution, we can implement changes quicker because the process is already streamlined. In this case, first we need to model the business changes and then implement it in Fabric.

## 12.2.1 Without Fabric

If the original solution was implemented without Fabric, and in a static business process, then it would be a challenge to modify the static business process. In this case, we need to re-model the business process and recreate the business process choreography too, which means changing the original WS-BPEL code. This implementation suffers from poor adaptability to the changes in the business scenario. Also, the governance of existing services and the cost of re-writing the original code becomes an obstacle in implementing the business change.

## 12.2.2 With Fabric

A solution implemented using Fabric offers dynamic business-service personalization and delivery, which reduces hard-coded binding changes with policy-based process customization. In Fabric, dynamic assembly utilizes metadata and policies to enforce assembly guidance at runtime, which reduces the static nature of the solution and increases the adaptability to a business change. Furthermore, you can rapidly update a solution that you create in Fabric in response to changes in the business scenario, design, market dynamics, and regulatory constraints. Fabric provides flexibility by using dynamic enablers.

In a Fabric solution, you can incorporate changes by configuring existing or adding new business policies and assertions that were created in the original solution. The original process implementation remains the same.

For the example scenario, the changes that the new business services of XYZBank introduced can be implemented by adding an extra instance for the already identified Bank Identifier assertion. This assertion already has an instance of ITS0. An additional instance of XYZ can be added to the ontology model and imported in Fabric.

Figure 12-1 on page 328 and Figure 12-2 on page 328 show the new models after the introduction of additional business services from XYZBank in the original business scenario for ITS0Bank:

- Verify Customer

Figure 12-1 on page 328 shows the model capturing the additional business process to use XYZBank's Verify Customer scenario. A decision is introduced to call the correct endpoint based on the Bank Identifier, so as to call the relevant Verify customer process that is based on the result.

If it is an ITS0Bank customer, then call the Verify ITS0Bank Customer process, and if it is an XYZBank customer, call the process to Verify XYZBank Customer. This is managed by using the Bank Identifier assertion.

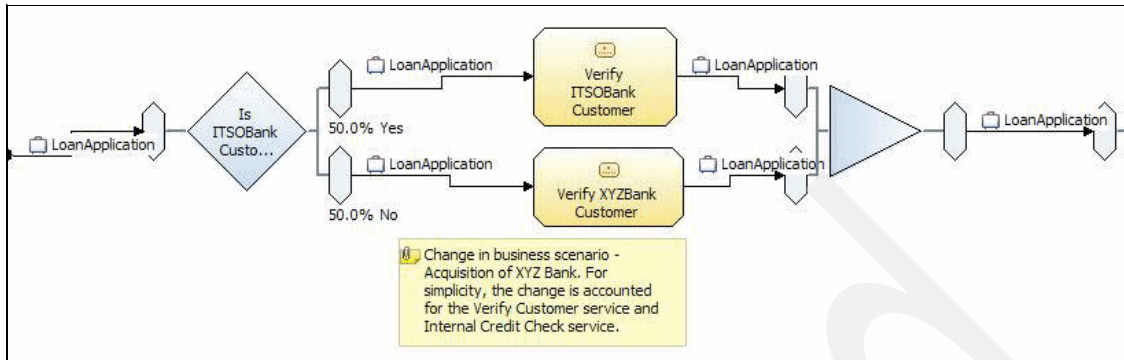


Figure 12-1 Verify Customer scenario

#### ► Internal Credit Check

Figure 12-2 shows the new model for the Internal Credit Check scenario. After the merger of two banks, ITSOBank's credit check becomes the priority for external credit checks. But, for internal customer credit checks, XYZBank's process needs to be used because it contains the information for its existing customer.

A decision point is introduced before calling the internal credit check process. In the case of an ITSOBank customer, ITSOBank's internal credit check service is utilized, and XYZBank's customers' internal credit check can be run using the XYZBank's internal credit check service. The decision is made based on the Bank identifier assertion before running the relevant internal credit check process.

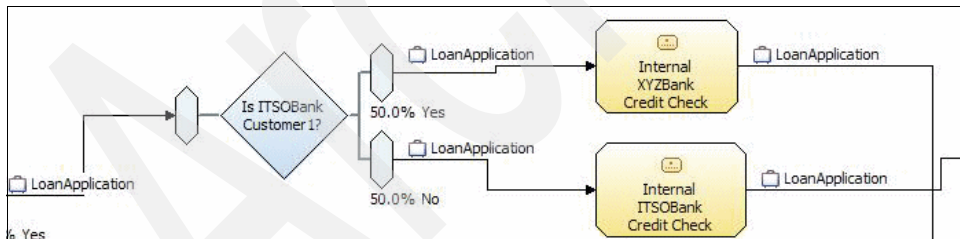


Figure 12-2 Internal Credit Check scenario

## 12.3 Managing business change

Managing this business change in Fabric requires minimal effort compared to the standard process implementation.

To implement this business change in Fabric:

1. Model:
  - a. Identify variability points.
  - b. Add an XYZ instance value to the Bank Identifier assertion.
  - c. Run the transformation.
  - d. Export the ontology model.
2. Assemble:
  - a. Import the ontology model in Fabric.
  - b. Replicate the new model in Composition Studio.
  - c. Add new endpoints and assertions for XYZBank.
  - d. Add or modify policies, if required.
  - e. Associate environments.
  - f. Run simulation on the test server.
3. Deploy:
  - a. Deploy metadata of the solution.
  - b. Test the process in Fabric.

In the next sections, we discuss these steps.

## 12.4 Modeling the change

The only change that is required for the original model is to introduce a new instance for the Bank Identifier assertion. We need to add this instance as a value XYZ to the original Bank Identifier assertion, which enables us to create a decision policy for the ITSOBank and XYZBank endpoints.

### 12.4.1 Identifying variability points

Based on the requirements, we can use Bank Identifier assertion to select the correct endpoint for Verify Customer and Internal Credit Search processes.

Enable the following values for the Bank Identifier assertion:

- ▶ For ITSO Bank: ITSO
- ▶ For XYZ Bank: XYZ

We need to add the instance for XYZ for the enumeration value.

## 12.4.2 Ontology modeling

While modeling the original scenario, the assertions in Figure 12-3 were created. For the Change In Business scenario, do not modify these assertions. You only need to create an instance of Bank Identifier with value XYZ.

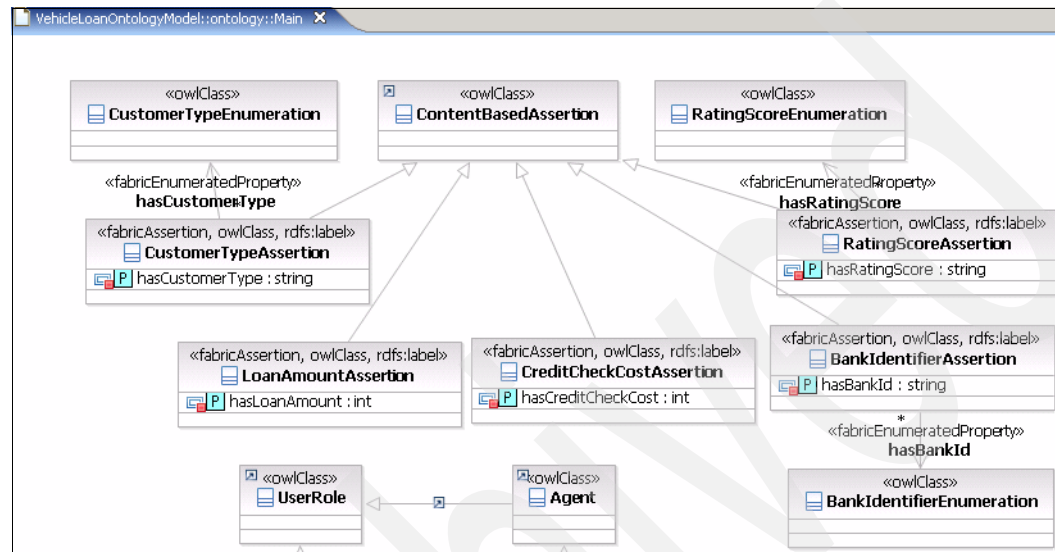


Figure 12-3 Modeled assertions

### Model change

To add an instance of BankIdentifierAssertion with the value of XYZ:

1. In Rational Software Architect's Fabric Modeling Tool, from the Palette menu, click **Class Instance**, as shown in Figure 12-4 on page 331.



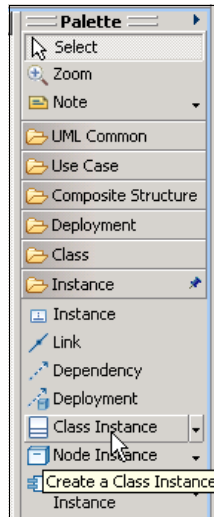


Figure 12-4 Create new class instance

2. Click the canvas, and choose **Existing Element**, as shown in Figure 12-5.

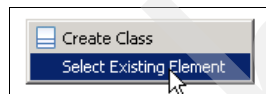


Figure 12-5 Select existing element

3. In the dialog, click **BankIdentifierEnumeration**, as shown in Figure 12-6 on page 332.

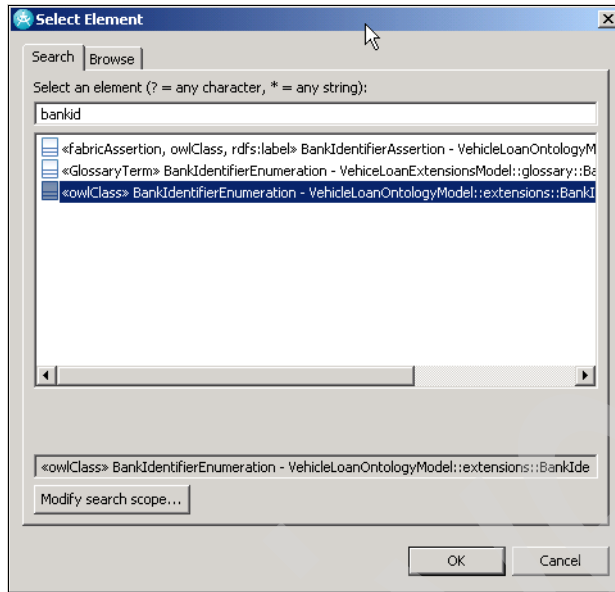


Figure 12-6 Select BankId class

4. Go to the Properties view, and change the name to BankIdentifierEnumeration:XYZ, as shown in Figure 12-7.

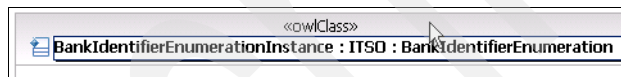


Figure 12-7 Name class

5. Type the value as XYZ, which creates an instance with value XYZ, as shown in Figure 12-8.

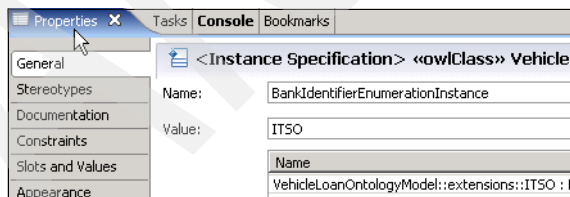


Figure 12-8 Properties

6. Figure 12-9 shows the enumerated values for the Bank Identifier along with other enumerated values.

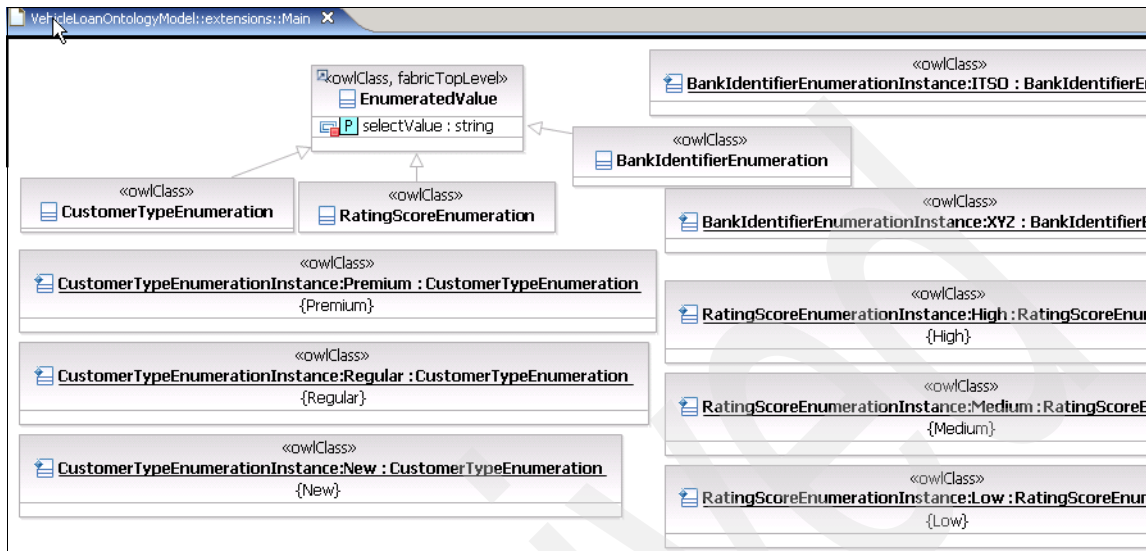


Figure 12-9 New glossary

7. Run ontology transformation to produce the OWL class similar to the Ontology Modeling in original scenario, as shown in Figure 12-10.

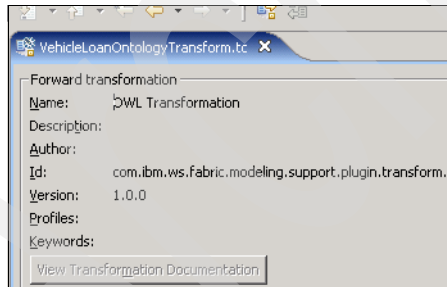


Figure 12-10 Run ontology transformation

## Creating FCA and exporting

Export the changed ontology model as a Fabric Content Archive (FCA) compressed file. Refer to 8.7.7, "Exporting the Fabric Content Archive" on page 192 for details about exporting the model as an FCA file.

**Note:** For this scenario, we call the exported FCA file `fabricitsobankChange.zip`.

## 12.5 Assembling the change

After the modeling is finished, assemble the solution in Fabric. We provide the basic instructions for assembly in Chapter 9, “Assembling Fabric solutions” on page 193, so refer to this chapter for detailed instructions and further graphics for these steps.

In the next few sections, we summarize the steps to take to perform changes to the already assembled solution.

### 12.5.1 Import the new ontology model into Fabric

We need to import the new ontology model into Fabric to use the new instance for XYZ in the Fabric solution.

Import the new ontology model FCA file that was generated from the modelling steps into Fabric. Refer to 9.5.3, “Importing the Ontology extension project” on page 231 for details on how to import an Ontology model FCA file into Fabric.

### 12.5.2 Replicate the project in Composition Studio

After you import the project into the Fabric console, open Composition Studio and replicate the project in Composition Studio. Refer to 9.5.6, “Replicating the project in Composition Studio” on page 242 for details about how to replicate the project in Composition Studio.

### 12.5.3 Adding endpoints

A Fabric endpoint is a service location that can be invoked. An endpoint indicates service interfaces and environments after a service is deployed. The address of an endpoint provides the details of the location where the service can be invoked. An endpoint might have a number of assertions to describe its characteristics or capabilities.

After the new ontology model is available in Fabric project, create additional endpoints to cater for the new scenario:

1. First, add both SCA and Web service endpoints for the Verify Customer XYZBank scenario.
2. Similarly, create endpoints for Internal Credit Check scenario change.

## Adding an SCA endpoint for Verify Customer XYZBank

To add an SCA endpoint for Verify Customer XYZBank:

1. In Composition Studio, in the Business Service Explorer menu, right-click the **ITSObankCBAPrj** → **Endpoint**, and from the Context menu, click **New** → **Endpoint**, as shown in Figure 12-11.

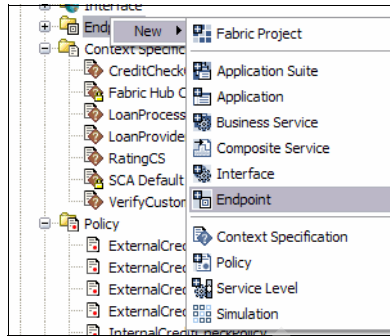


Figure 12-11 Create endpoint menu

2. In the dialog box, type the value `VerifyCustomerXYZBankSCAExport` for Name. The Project, Namespace, and Address values are already populated, as shown in Figure 12-12. Click **Next**.

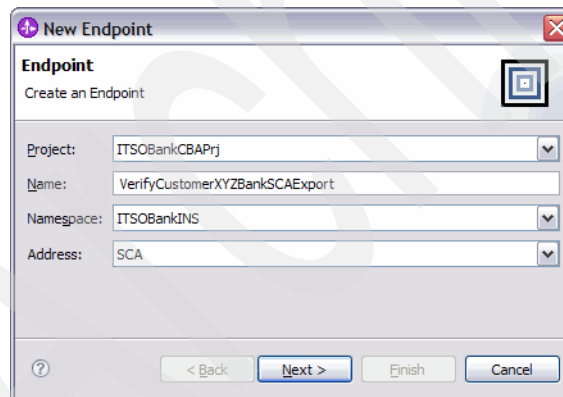


Figure 12-12 Create SCA Endpoint for Verify Customer XYZBank

3. In the SCA Address field, type `sca://ITS0_imp1/VerifyCustomerSCAExport`, as shown in Figure 12-13 on page 336. Click **Finish**.

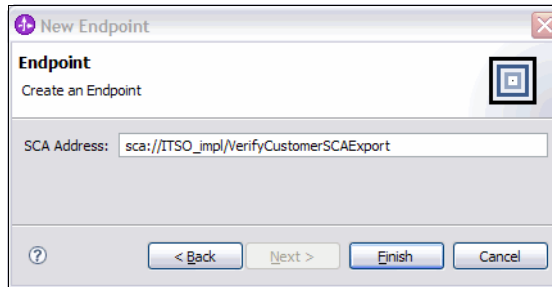


Figure 12-13 Specify SCA address for Verify Customer XYZBank endpoint

4. Save your work by simultaneously pressing **Ctrl+S**.

**Note:** Save your work regularly by using the key combination Ctrl+S or clicking the menu option **File** → **Save**.

## Adding a Web service endpoint for Verify Customer XYZBank

To add a Web Service endpoint for the Verify Customer XYZBank:

1. In Composition Studio, from the Business Service Explorer menu, right-click **ITSObankCBAPrj** → **Endpoint**, and from the Context menu, click **New** → **Endpoint**:
  - In the Name field, type `VerifyCustomerXYZBankExport_VerifyCustomerHttpPort` as shown in Figure 12-14 on page 337.
  - In the Address field, select **HTTP**.

The Project and Namespace fields are already populated, as shown in Figure 12-14 on page 337. Click **Next**.

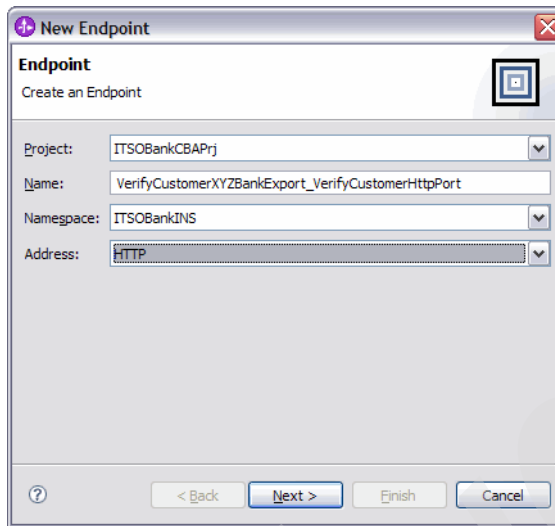


Figure 12-14 Create endpoint for Verify Customer WS

2. In the URL field, type `http://localhost:9081/ITSO_implWeb/sca/VerifyCustomerExport`. In the Message Type field, select **SOAP 1.1**, as shown in Figure 12-15. Click **Finish**.

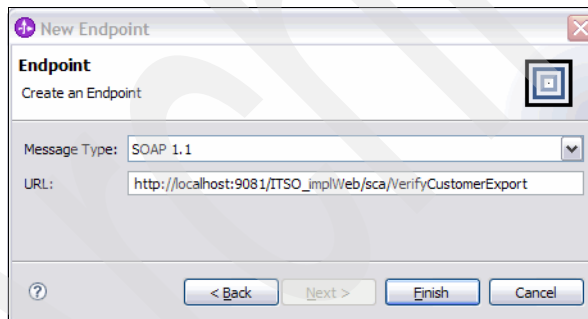


Figure 12-15 Specify URL for Verify Customer Web Service endpoint

**Note:** The URL type is  
`http://<hostname>:<port>/ITSO_implWeb/sca/VerifyCustomerExport`

In Figure 12-15, localhost is chosen for the host name and 9081 is the port for the endpoints, per the configuration during installation. Choose the correct host name and port that is applicable to your installation.

## Creating an SCA endpoint for Internal Credit Check XYZBank

To create an SCA endpoint for Internal Credit Check XYZBank:

1. In Composition Studio, go to the Business Service Explorer menu, and right-click **ITSObankCBAPrj** → **Endpoint**. From the Context menu, click **New** → **Endpoint**. In the Name field, type `InternalCreditCheckXYZBankSCAExport`. Accept the defaults for the Project, Namespace, and Address fields, as shown in Figure 12-16. Click **Next**.

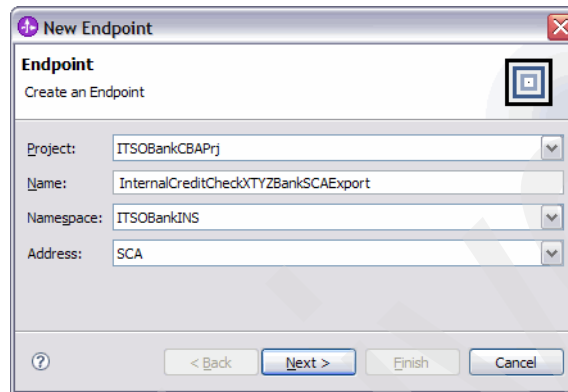


Figure 12-16 Create endpoint

2. As shown in Figure 12-17, in the SCA Address field, type `sca://ITSO_impl/InternalCreditCheckSCAExport`. Click **Finish**.

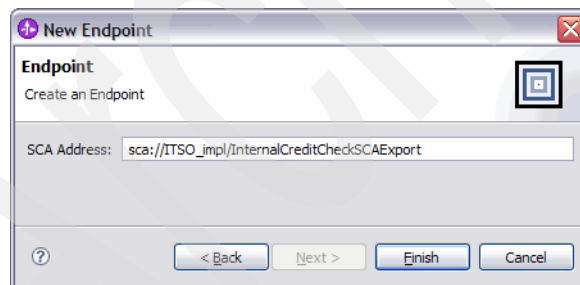


Figure 12-17 Create SCA endpoint for Internal Credit Check

## Adding a Web services endpoint for Internal Credit Check XYZBank

1. In the URL field, type the value `http://localhost:9081/ITSO_implWeb/sca/InternalCreditCheckExport`. Accept the Message Type **SOAP 1.1**, as shown in Figure 12-19 on page 339. Click **Finish**.



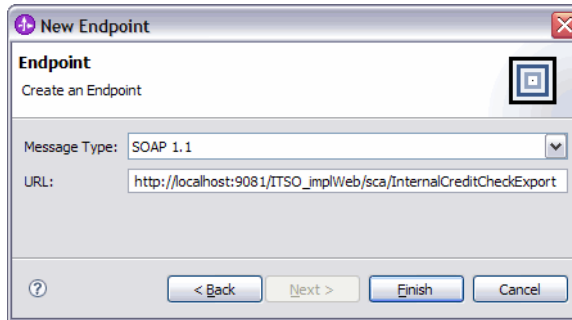


Figure 12-18 Add Endpoint for Internal credit check Web Service

2. In Composition Studio, go to the Business Service Explorer menu, and right-click **ITSOBankCBAPrj** → **Endpoint**. From the Context menu, click **New** → **Endpoint**:
  - In the Name field, type `InternalCreditCheckXYZBankExport_CreditCheckHttpPort`.
  - In the Address field, select **HTTP**.

The Project and Namespace fields are already populated, as shown in Figure 12-19. Click **Next**

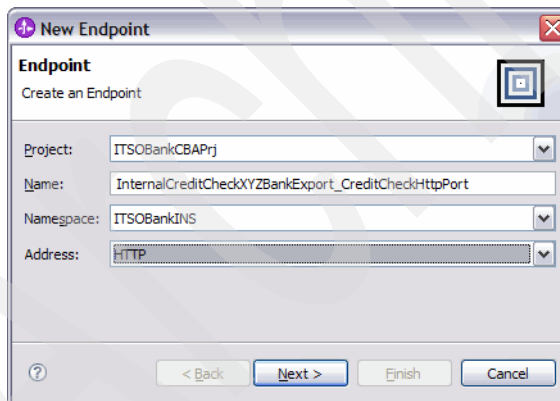


Figure 12-19 Create endpoint

## 12.5.4 Adding an assertion to the endpoints

After you create the additional endpoints, apply an assertion to these endpoints. Here, `BankIdentifierAssertion` is added to the endpoints that were created for the new XYZ Bank scenario.

## Adding an assertion to the SCA endpoint for Verify Customer XYZBank

To add an assertion to the SCA endpoint for Verify Customer XYZBank:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **VerifyCustomerXYZBankSCAExport**.
2. Click the **Assertions** tab, and then click the **Add** button. In the dialog box, expand **Interoperability Assertion** → **Content Based Assertion**, and click **Bank Identifier Assertion**, as shown in Figure 12-20. Click **OK**.

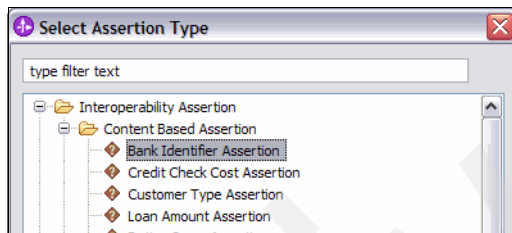


Figure 12-20 Add Bank Identifier Assertion

3. In the subsequent dialog box, select **Required**, and in the BankIdentifier field, select XYZ from the choice box, as shown in Figure 12-21. Click **OK**.

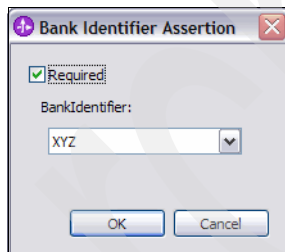


Figure 12-21 Bank Identifier assertion

Figure 12-22 on page 341 shows the added assertion.

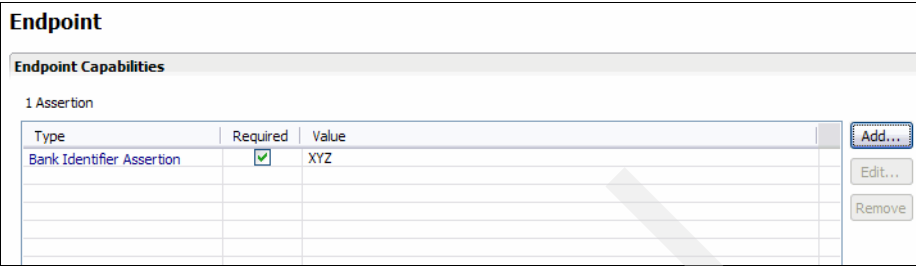


Figure 12-22 Added assertion

### Adding an assertion to the Web services endpoint for Verify Customer XYZBank

To add an assertion to the Web services endpoint for Verify Customer XYZBank:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **VerifyCustomerXYZBankExport\_VerifyCustomerHttpPort**.
2. Click the **Assertions** tab, and then click the **Add** button. In the dialog box, expand **Interoperability Assertion** → **Content Based Assertion**, and click **Bank Identifier Assertion**. Click **OK**. In the BankIdentifier field, select XYZ, and select the **Required** option. Click **OK**.

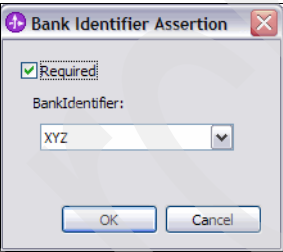


Figure 12-23 Select assertion value

Figure 12-24 on page 342 shows the added assertion.

Endpoint			
Endpoint Capabilities			
1 Assertion			
Type	Required	Value	
Bank Identifier Assertion	<input checked="" type="checkbox"/>	XYZ	<input type="button" value="Add..."/>
			<input type="button" value="Edit..."/>
			<input type="button" value="Remove"/>

Figure 12-24 Added assertion

## Adding an assertion to the SCA Internal Credit Check XYZBank

To add an assertion to the SCA Internal Credit Check XYZBank:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **InternalCreditCheckXYZBankSCAExport**.
2. Click the **Assertions** tab, and then click the **Add** button. In the dialog box, expand **Interoperability Assertion** → **Content Based Assertion**, and click **Bank Identifier Assertion**. Click **OK**. In the BankIdentifier field, select XYZ, and select the **Required** option, as shown in Figure 12-25. Click **OK**.

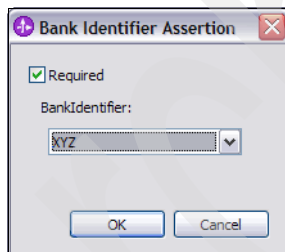


Figure 12-25 Select Bank Identifier Assertion Value

Figure 12-26 on page 343 shows the added assertion and the existing CreditCheckCostAssertion assertion.

Endpoint		
Endpoint Capabilities		
2 Assertions		
Type	Required	Value
Bank Identifier Assertion	<input checked="" type="checkbox"/>	XYZ
Credit Check Cost Assertion	<input checked="" type="checkbox"/>	2

Figure 12-26 Added Assertion Bank Identifier

## Adding an assertion to Web service Internal Credit Check XYZBank

To add an assertion to Web service Internal Credit Check XYZBank

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **InternalCreditCheckXYZBankExport:CreditCheckHttpPort**.
2. Click the **Assertions** tab, and then click the **Add** button. In the dialog box, expand **Interoperability Assertion** → **Content Based Assertion**, and click **Bank Identifier Assertion**, as shown in Figure 12-27. Click **OK**.

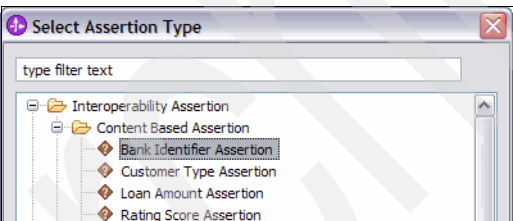


Figure 12-27 Add Assertion

3. From the BankIdentifier field, select XYZ, and select the **Required** option. Click **OK**. Figure 12-28 on page 344 shows the added assertion and the existing CreditCheckCostAssertition assertion.

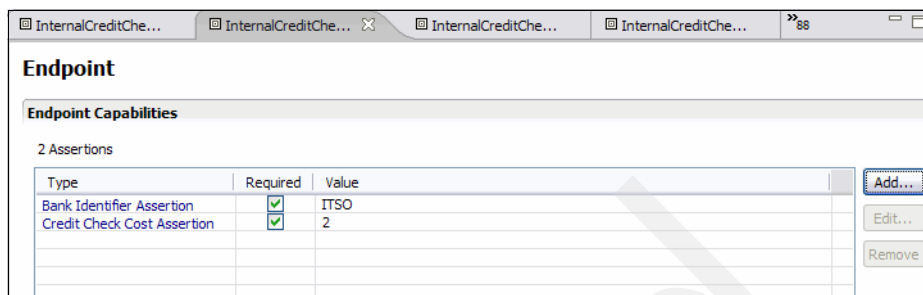


Figure 12-28 Added assertion

## 12.5.5 Associate interfaces

After you add the assertions, add the relevant interfaces to the recently created endpoints for the XYZBank scenario.

### Verifying Customer

To add an interface to the SCA endpoint of Verify Customer:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **VerifyCustomerXYZBankSCAExport**. Click the **Interfaces** tab, and then click the **Add** button. In the dialog box, click the **VerifyCustomer** interface, as shown in Figure 12-29. Click **OK**.

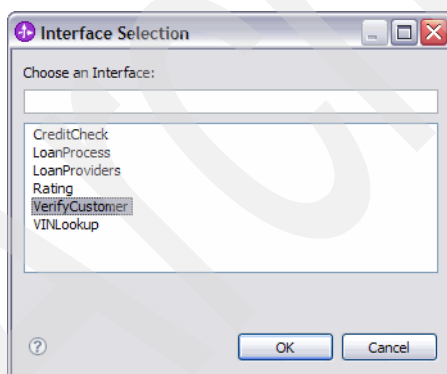


Figure 12-29 Add Interface Verify Customer

Figure 12-30 on page 345 shows the associated interface Verify Customer.

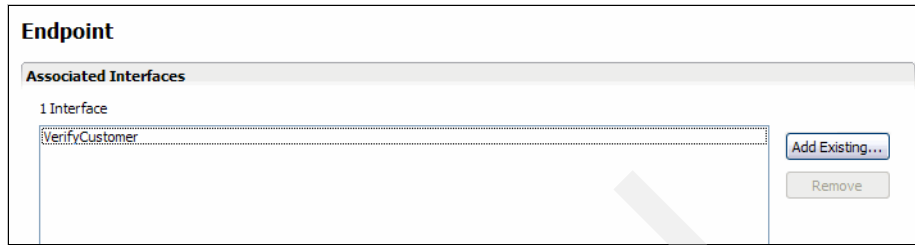


Figure 12-30 Added Interface VerifyCustomer

2. Add the **VerifyCustomer** interface to the Web service endpoint **VerifyCustomerXYZBankExport\_VerifyCustomerHttpPort**.

## Internal Credit Check

To add an interface to the SCA endpoint of Internal Credit Check:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click the **InternalCreditCheckXYZBankSCAExport**. Click the **Interfaces** tab, and then click the **Add** button. In the dialog box, click **CreditCheck** interface, as shown in Figure 12-31. Click **OK**. This step adds the Credit Check interface.

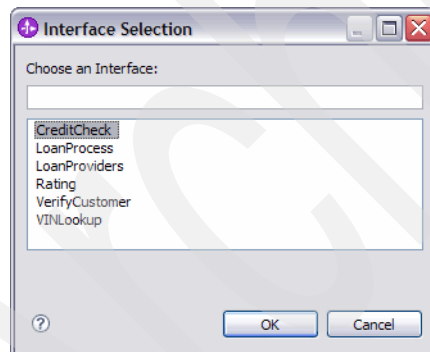


Figure 12-31 Add Credit Check interface

2. Add the **CreditCheck** interface to the Web service endpoint **InternalCreditCheckXYZBankExport:CreditCheckHttpPort**.

## 12.5.6 Associating environments

Now, associate relevant environments to the new endpoints created for the XYZ Bank scenario. Similar to the original assembly of the solution, add SCA endpoints to Development and HTTP endpoints to the Production environment.

## Associating an environment for Verify Customer SCA

To associate an environment for Verify Customer SCA:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **VerifyCustomerXYZBankSCAExport**. Click the **Environments** tab, and then click the **Add Existing** button. In the dialog box, click **Development**, and click **OK**, as shown in Figure 12-32.

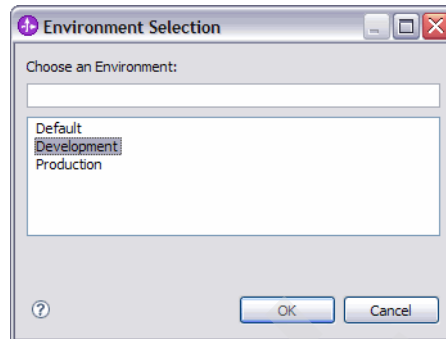


Figure 12-32 Add Environment to Verify Customer

Figure 12-33 shows the associated Development environment for this endpoint.

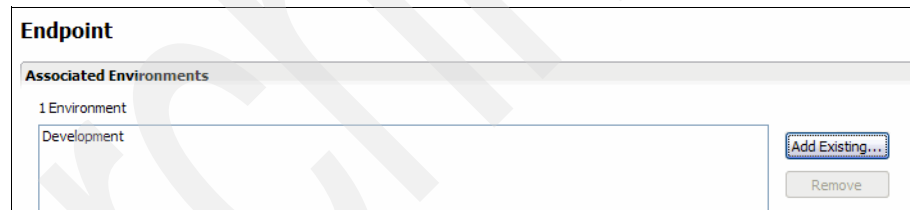


Figure 12-33 Associated environments

## Associating an environment for Verify Customer Web service

To associate an environment for Verify Customer Web service:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **VerifyCustomerXYZBankExport\_VerifyCustomerHttpPort**. Click the **Environments** tab, and then click the **Add Existing** button. In the dialog box, click **Production**, and click **OK**.

Figure 12-34 on page 347 shows the selected environment.



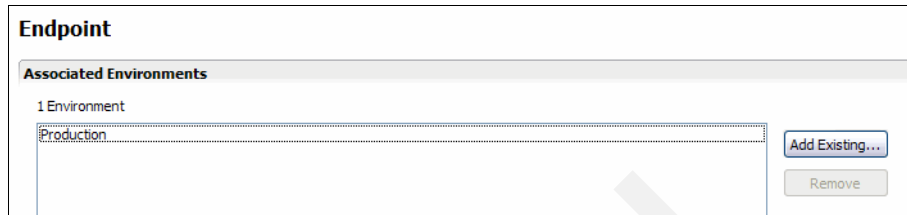


Figure 12-34 Associate Environment for Verify Customer Web Service

## Associating an environment for Internal Credit Check SCA

To associate an environment for Internal Credit Check SCA:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint**, and double-click **InternalCreditCheckXYZBankSCAExport**. Click the **Environments** tab, and then click the **Add Existing** button. In the dialog box, click **Development**, and click **OK**.

Figure 12-35 shows the associated environment for this endpoint.

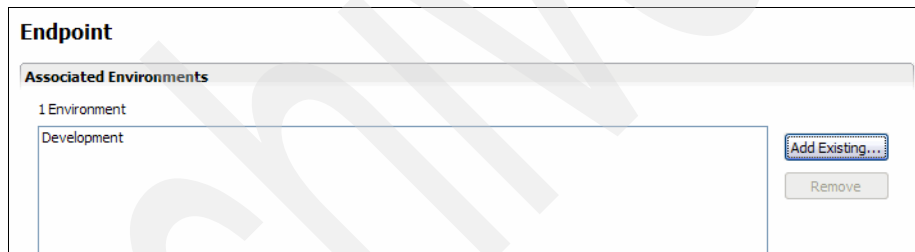


Figure 12-35 Add Environment for Internal credit check SCA

## Associating an environment for Internal Credit Check Web service

To associate an environment for Internal Credit Check Web service:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Endpoint** and double click **InternalCreditCheckXYZBankExport\_CreditCheckHttpPort**. Click on the **Environments** tab, and then click the **Add Existing** button. In the dialog box, click **Production**, and click **OK**.

Figure 12-36 on page 348 shows the associated environment for this endpoint.

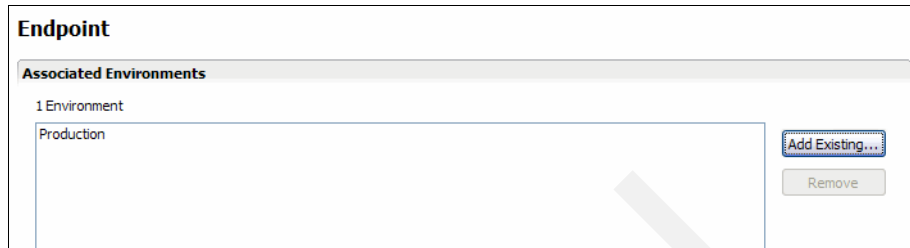


Figure 12-36 Add Environment for Internal Credit Check Web Service

## 12.5.7 Adding policies

The next step is to review policies and add or extend business policies if required. The existing Verify Customer policy does not require a change, since this policy already has the BankIdentifierAssertion assertion. The assertion is valid for both SCA and Web service endpoints for this policy.

### Extending the policy for Internal Credit Check

We already have a policy for Internal Credit Check added in the original solution, but do not have the mechanism to handle the bank identifier. So, we need to extend this policy InternalCreditCheck policy to add the assertion BankIdentifierAssertion:

1. In the Business Service Explorer, expand **ITSOBankCBAPrj** → **Policy**, and double-click **InternalCreditCheckPolicy** as shown in Figure 12-37.

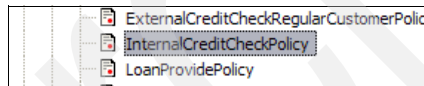


Figure 12-37 Extend Policy for Internal Credit Check

2. Following the directions in 12.5.4, “Adding an assertion to the endpoints” on page 339, add BankIdentifierAssertion to the InternalCreditCheck policy. Select the **Required** and **Fill from Content** options when you add the assertion, as shown in Figure 12-38 on page 349.

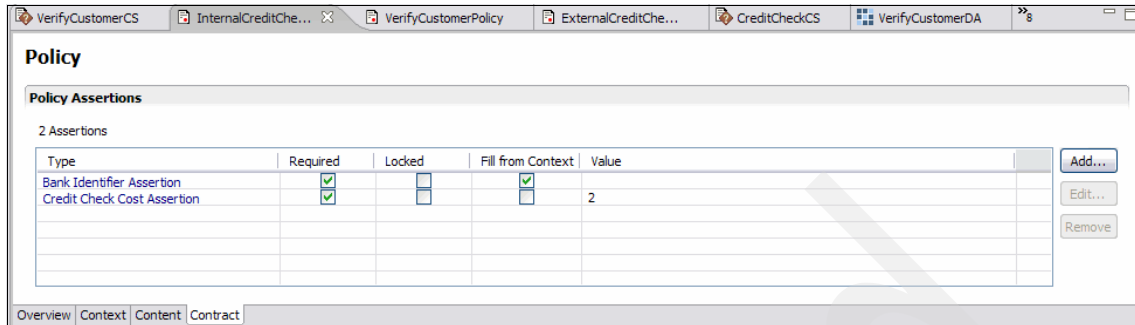


Figure 12-38 Add Assertion to Internal Credit Check Policy

## 12.5.8 Changing the Credit Check context specification

The final change is to modify the context specification for Credit Check:

1. As per Figure 12-39, expand **ITSOBankCBAPrj** → **Context Specification**, and double-click **CreditCheckCS**, which we created for the original scenario.

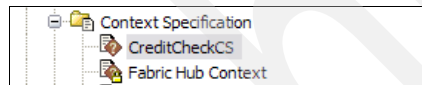


Figure 12-39 Open Credit Check Context Spec

2. Click the **Dimensions** tab. Click the **Add** button, which is located below the Content-based Dimension section. In the resulting dialog box, click **Bank Identifier Assertion**. Click **OK**. Clear the **Required** option.

The bank identifier assertion is added to the policy as shown in Figure 12-40 on page 350.

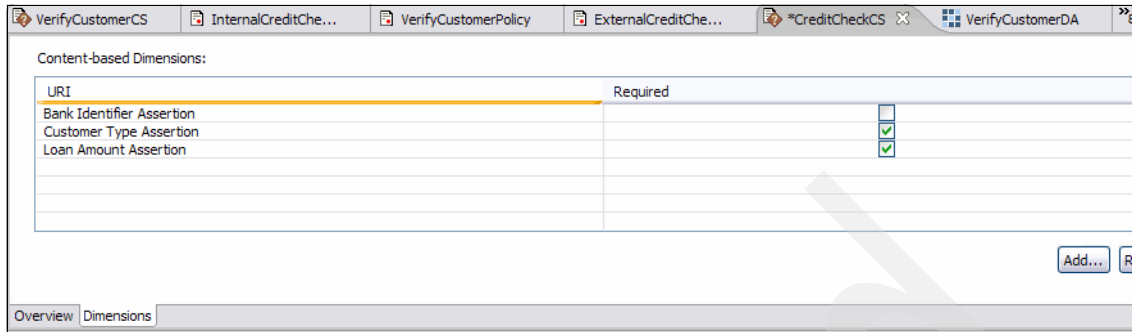


Figure 12-40 Add Content Based Dimensions

3. Press **Ctrl+S** to save your work.

## 12.5.9 Running simulation

The solution is ready, and we can simulate the environment run. The following steps show the simulation that is created for Verify Customer. You can use the same steps to create a simulation for Credit Check.

To run simulation:

1. In the Business Service Explorer, right-click **ITSOBankCBAPrj** → **Simulation**, and from the Context menu, click **New** → **Simulation**, as shown in Figure 12-41.

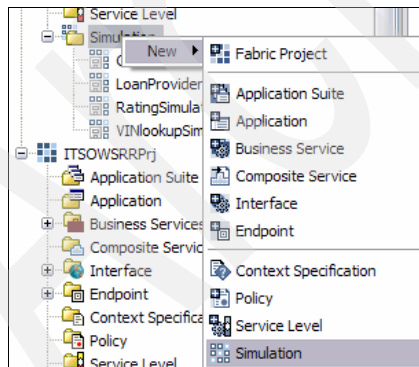


Figure 12-41 Run new simulation

2. In the Name field, type **VerifyCustomerSimulation**. In the Namespace field, select **ITSOBankINS**, and in the Dynamic Assembly field, select **VerifyCustomerDA** as shown in Figure 12-42 on page 351. Click **Finish**.

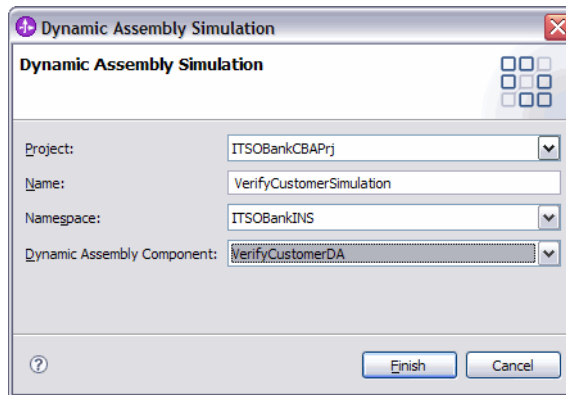


Figure 12-42 VerifyCustomer Simulation

3. Click the **Simulation Input** tab, go to the Required Context section, and click the **Edit** button. In the resulting dialog box, specify **XYZ** as the value for the **BankIdentifier**. Click **OK**, as shown in Figure 12-43.

 A screenshot of the 'Simulation' window, specifically the 'Simulation Input' tab. The window is divided into three main sections: 'General Information', 'Simulation Context', and 'Required Context'. 
 

- General Information:** Contains fields for ID (ue4b0dda2-7636-432b-9d5b-8e920a89ba4d), Namespace (ITSOBankINS), Name (VerifyCustomerSimulation), and Description. It also shows 'Date Created' and 'Date Modified' as '<Not Published to Master Business Services Repository>'. There is an 'Edit...' button next to the ID field.
- Simulation Context:** Contains fields for Context Specification (VerifyCustomerCS), Composite Service (ITSO), Dynamic Assembly Component (VerifyCustomerDA), Date (Feb 19, 2008), Time (4:30 PM), and Time Zone ((GMT-05:00) Eastern Time).
- Required Context:** Contains a table with one row: Interface (VerifyCustomer), Environment (Development), and Bank Identifier Assertion (BankIdentifier: XYZ). There is an 'Edit...' button next to the Bank Identifier Assertion field.
- Optional Context:** A section at the bottom with buttons for 'Add Model Dimension...' and 'Add Content-based Dimension...'.

 At the bottom of the window is a tab bar with 'Simulation Input' and 'Results' tabs.

Figure 12-43 Add Bank Identifier Value XYZ

4. For the values of BankIdentifier XYZ and environment Development, the expected endpoint should be VerifyCustomerXYZBankSCAExport. Click the **Run** button. Figure 12-44 shows the simulation results.

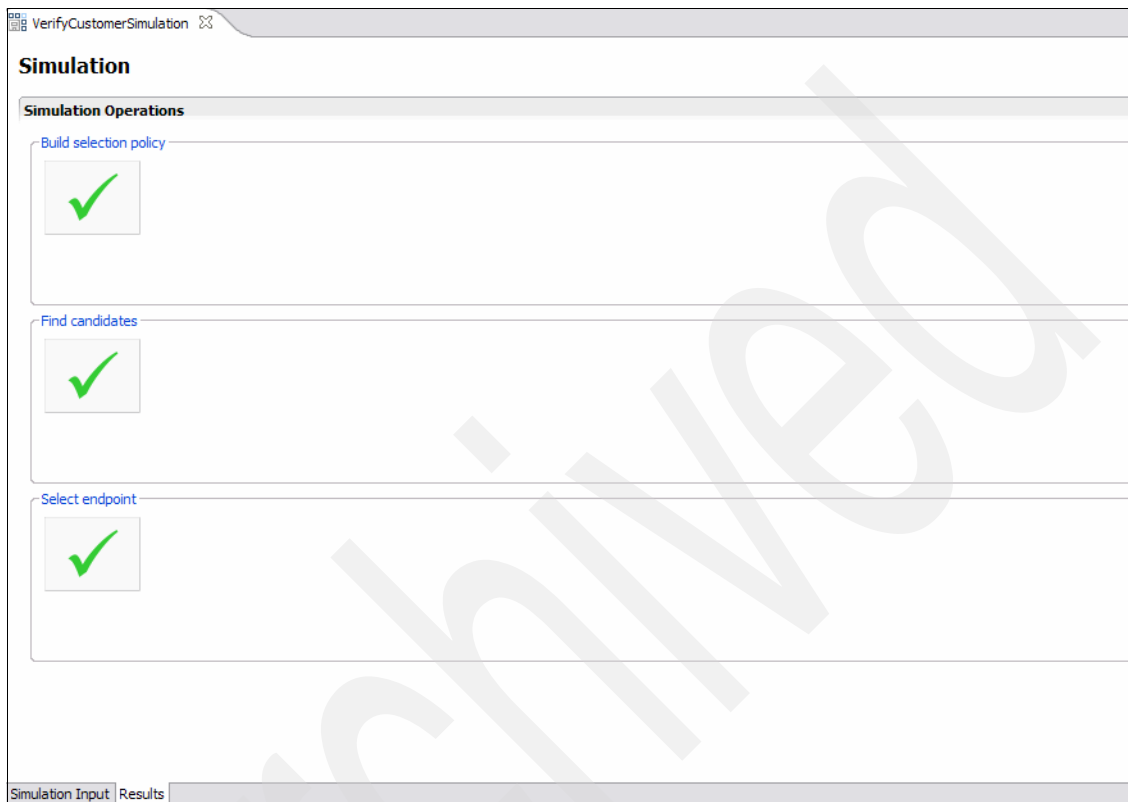


Figure 12-44 Simulation results

5. In the **Simulation Operation** tab, click the **Selected Endpoint** tab if it is not already selected. It shows the selected endpoint VerifyCustomerXYZBankSCAExport as expected, which we show in Figure 12-45.

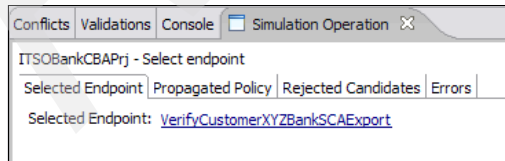


Figure 12-45 Selected Endpoint in Simulation

6. Repeat the same simulation with the BankIdentifier value of **ITSO**, as shown in Figure 12-46.

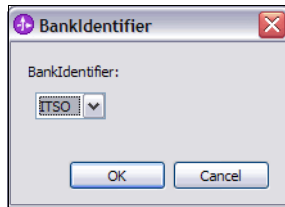


Figure 12-46 New value ITSO

7. For the value of ITSO, the expected endpoint is VerifyCustomerSCAExport. Click the **Run** button, and review the simulation results, as shown in Figure 12-47.

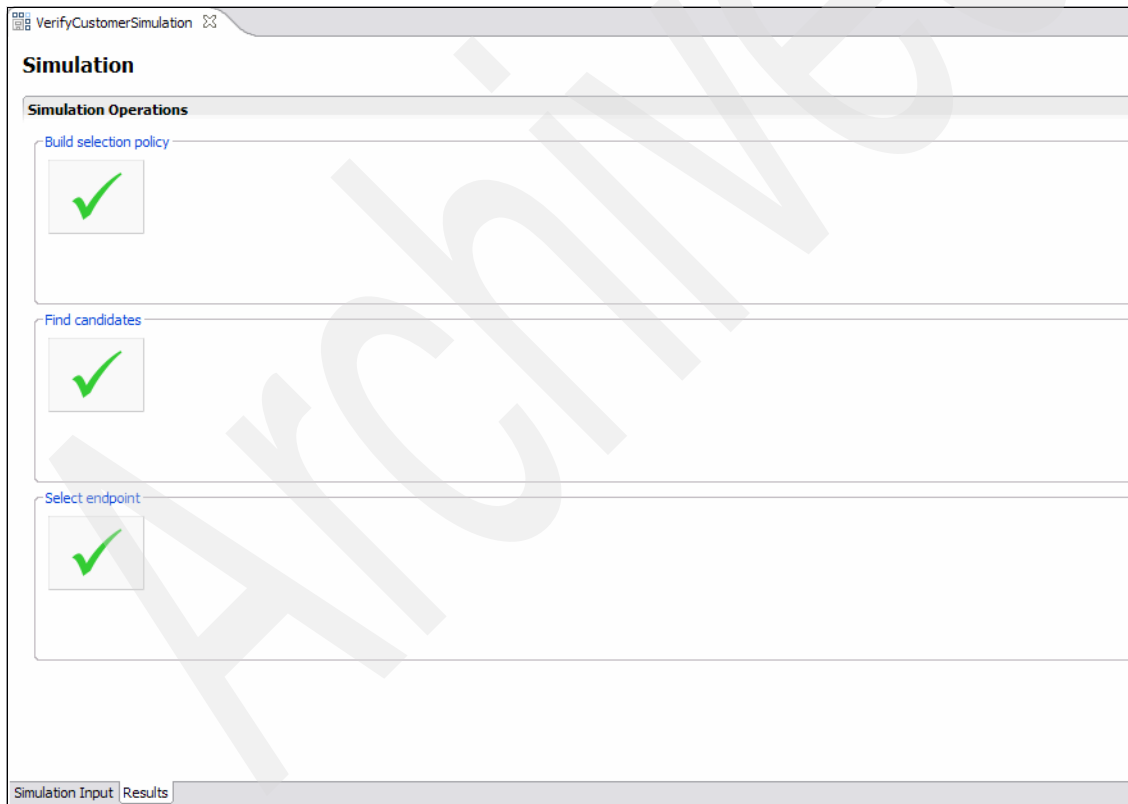


Figure 12-47 Simulation results

Figure 12-48 shows the selected endpoint VerifyCustomerSCAExport, which is the expected result for this test scenario.

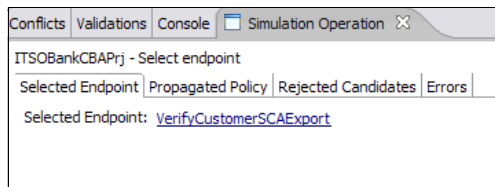


Figure 12-48 Selected Endpoint

## 12.6 Deploying and testing

Finally, we need to deploy the solution. In this case, there is no need to deploy the business process again, and we only publish the business services metadata to the Business Services Repository in Fabric.

### 12.6.1 Deploying metadata

Publish the metadata changes from the Composition Studio. Refer to 10.2, “Publishing the business service metadata” on page 290 for details about how to deploy metadata to the Fabric Business Services Repository. For all of the added metadata, owner and publisher should be ITSOBankOrg to avoid issues during migration.

### 12.6.2 Testing the process

Similar to the original application testing, we use the Fabric test server to test the changed application. Refer to 10.5, “Testing the SCA process” on page 303 for detailed directions about how to test your application within the test server.

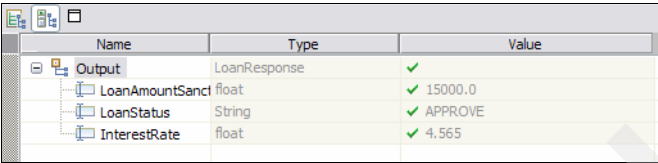
Figure 12-49 shows input data for the test.

Name	Type	Value
Input	LoanApplication	✓
CustomerIdentific	string	✓ 100
CustomerAddress	string	✓
VIN	string	✓ 12345678901234567
LoanAmountRequ	float	✓ 15000
BankId	string	✓ XYZ

Figure 12-49 Input data



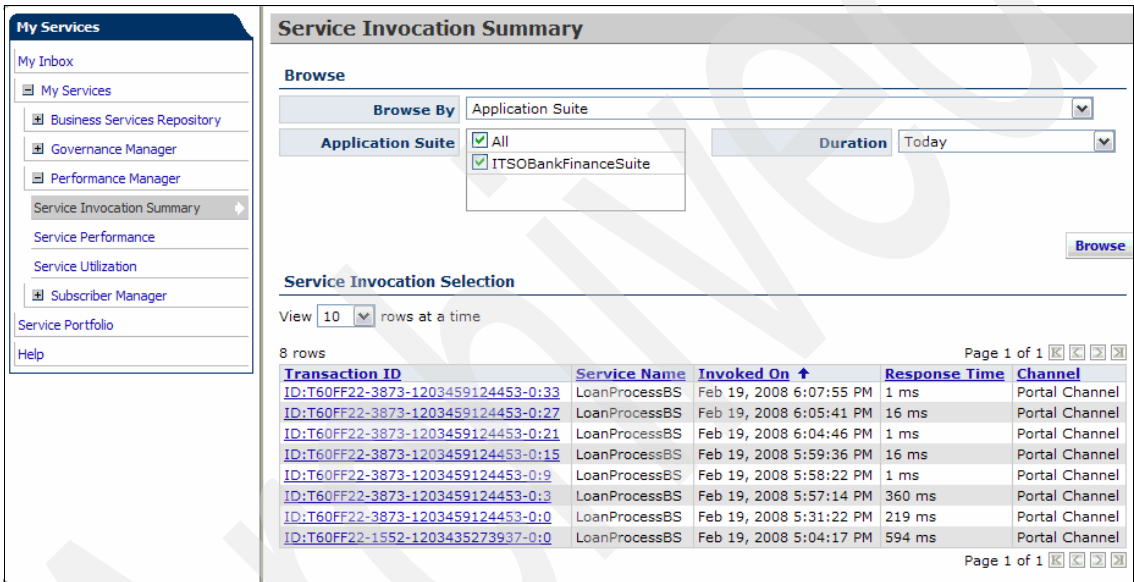
Figure 12-50 shows the results after the test is finished.



Name	Type	Value
Output	LoanResponse	✓
LoanAmountSanct	float	✓ 15000.0
LoanStatus	String	✓ APPROVE
InterestRate	float	✓ 4.565

Figure 12-50 Output data

Figure 12-51 shows the summary for Performance Service Invocation within the Fabric admin console.



**My Services**

- My Inbox
- My Services
  - Business Services Repository
  - Governance Manager
  - Performance Manager
  - Service Invocation Summary
  - Service Performance
  - Service Utilization
  - Subscriber Manager
- Service Portfolio
- Help

**Service Invocation Summary**

**Browse**

Browse By: Application Suite

Application Suite: ☒ All ☒ ITSOBankFinanceSuite

Duration: Today

**Service Invocation Selection**

View 10 rows at a time

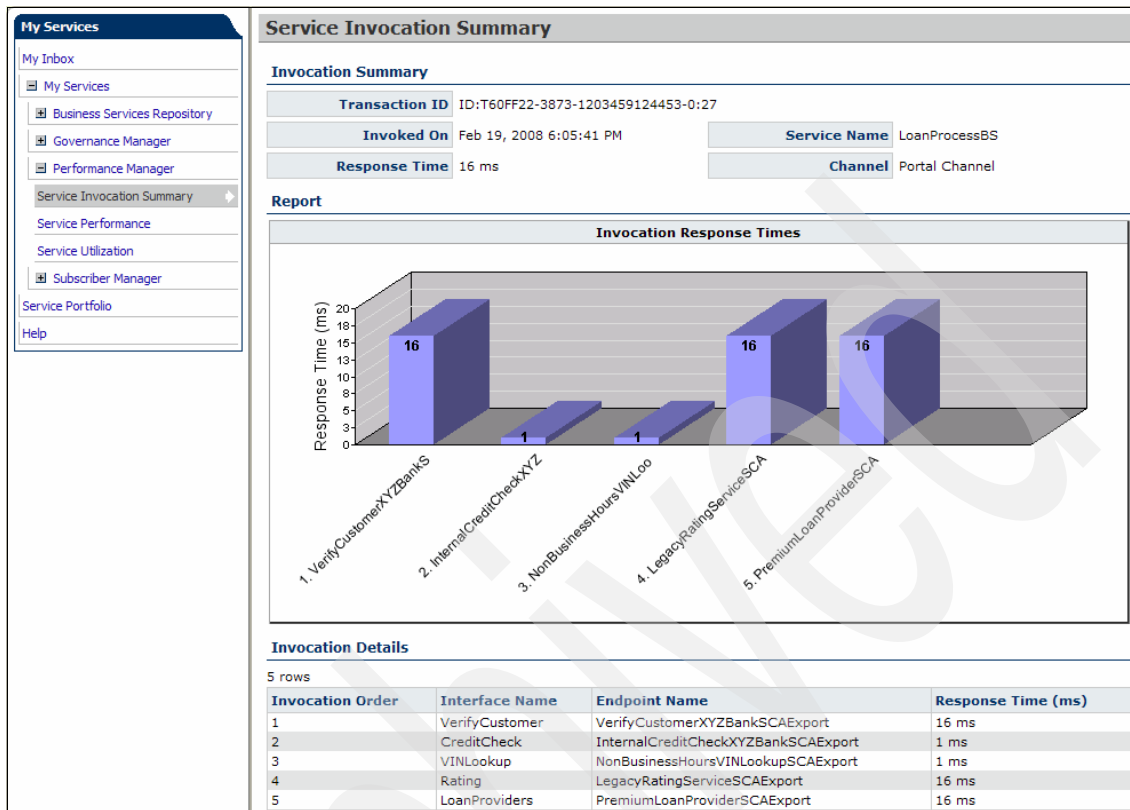
8 rows

Transaction ID	Service Name	Invoked On ↑	Response Time	Channel
ID:T60FF22-3873-1203459124453-0:33	LoanProcessBS	Feb 19, 2008 6:07:55 PM	1 ms	Portal Channel
ID:T60FF22-3873-1203459124453-0:27	LoanProcessBS	Feb 19, 2008 6:05:41 PM	16 ms	Portal Channel
ID:T60FF22-3873-1203459124453-0:21	LoanProcessBS	Feb 19, 2008 6:04:46 PM	1 ms	Portal Channel
ID:T60FF22-3873-1203459124453-0:15	LoanProcessBS	Feb 19, 2008 5:59:36 PM	16 ms	Portal Channel
ID:T60FF22-3873-1203459124453-0:9	LoanProcessBS	Feb 19, 2008 5:58:22 PM	1 ms	Portal Channel
ID:T60FF22-3873-1203459124453-0:3	LoanProcessBS	Feb 19, 2008 5:57:14 PM	360 ms	Portal Channel
ID:T60FF22-3873-1203459124453-0:0	LoanProcessBS	Feb 19, 2008 5:31:22 PM	219 ms	Portal Channel
ID:T60FF22-1552-1203435273937-0:0	LoanProcessBS	Feb 19, 2008 5:04:17 PM	594 ms	Portal Channel

Page 1 of 1

Figure 12-51 Performance Service Invocation summary

In the Fabric admin console, the service invocation summary through the Performance Manager is shown in Figure 12-52 on page 356. It shows the invoked endpoints during the test run.



# Best practices and troubleshooting

In this chapter, we describe best practices while you develop, deploy, manage, and govern a WebSphere Business Service Fabric (Fabric) solution. In this chapter, we also introduce various troubleshooting techniques that are available to you while you use Fabric.

In this chapter, we discuss the following topics:

- ▶ 13.1, “Problem determination in Fabric Composition Studio” on page 358
- ▶ 13.2, “Problem determination in Fabric runtime” on page 359
- ▶ 13.3, “Problem determination in underlying WebSphere Process Server, WebSphere Integration Developer infrastructure” on page 361
- ▶ 13.4, “Common issues” on page 363
- ▶ 13.5, “General best practices and recommendations” on page 364

## 13.1 Problem determination in Fabric Composition Studio

In this section, we discuss problem determination in Fabric Composition Studio.

**Problem:** Fabric Composition Studio does not load correctly after installing the WebSphere Service Registry and Repository (WSRR) plug-in.

**Solution:** The best practice is to install the WSRR plug-in before you install the Fabric Composition Studio plug-in. Doing this in the reverse order may corrupt the Fabric Composition Studio plug-in.

**Problem:** Fabric Unit Test Environment (UTE) does not start inside WebSphere Integration Developer.

**Solution:** In the Server Overview section, ensure that you supplied the correct user ID and password. The default username and password is “admin”.

If it still does not start, the Fabric UTE might be started from the command prompt. Navigate to `<WID_INSTALL_HOME>\runtimes\bi_v61\profiles\WBSFabric\bin`, and issue the `startServer.bat server1` command.

After the server is started, switch back to the WebSphere Integration Developer console, and the server status should now be in “Started” state.

**Problem:** Dynamic Assembly sometimes fails when using Web services imports/exports bindings to link WS-BPEL to back end service providers.

**Solution:** It is a best practice to use SCA Import/Export pairs in place of Web services Import/Export.

User Work Areas do not work when the WS-BPEL engine invokes partner links asynchronously. Fabric Dynamic Assembler uses the Context about (SCA) service usage to perform policy-driven endpoint selection. The extensions to SCA that propagate this context use the WebSphere Programming Model Extension (PME), called User Work Areas.

## 13.2 Problem determination in Fabric runtime

In this section, we discuss problem determination in Fabric runtime.

**Problem:** You performed manual installation of the Foundation Pack; however, the Fabric applications are not starting up, when you tried to start in the admin console.

**Solution:** There are many reasons why the Fabric EARs do not start when deployed on WebSphere Process Server. First, ensure that the JDBC resources are configured properly. When you do a manual installation, sometimes the port number under **JDBC providers → DB2 Universal JDBC Driver Provider (XA) → Data sources → fabric bsr** is wrongly set as “50001” when the actual DB2 port value might be different. Ensure that the DB2 port number is set correctly.

Check the DB2 port number for all of the following data sources under **JDBC providers → DB2 Universal JDBC Driver Provider (XA) → Data sources**:

- ▶ fabric bsr
- ▶ fabric gm
- ▶ fabric pm

Perform a Test Connection on all of the above data sources, and make sure it is successful before you try to start the Fabric applications.

Also ensure that the following jars are present in %wpshome%\lib\ext:

- ▶ fabric-da-api.jar
- ▶ fabric-da-sca.jar
- ▶ fabric-da-scdl.jar

**Problem:** Proper endpoint selection occurs during Policy Simulation in the Fabric Composition Studio; however, it does not work at runtime when the policy models are deployed on the Fabric runtime.

**Solution:** There may be various reasons to this. Ensure the following:

- ▶ Ensure that the runtime catalog is up-to-date. If you are not working against a UTE, verify that your changes were approved and published with the Governance Manager's change management tooling.
- ▶ Flush caches or restart the fabric-engine EAR and try again.
- ▶ Enable Verbose Logging on the Dynamic Assembler component. This is logged to com.ibm.ws.fabric.da. Set the logger to at least an INFO detail level.

**Problem:** Endpoint selection seems to be ignoring certain parts of context.

**Solution:** The following might be the causes:

- ▶ Check if context propagation is working.
- ▶ Ensure that the Work Area service is enabled. If not, enable and restart the application server.
- ▶ Check if the application is expecting context to be passed through a non-SCA intermediary. If so, the intermediary may be dropping SOAP headers.
- ▶ Check if the invocation chain includes a BPEL that invokes an interface export that exposes a Dynamic Assembler component.
- ▶ Try replacing the interaction with an SCA import/export pair.

**Problem:** Cache flushing

**Solution:** Flush the cache on all of the following occasions:

- ▶ Changes to endpoint capabilities.
- ▶ Changes to endpoint status.
- ▶ Changes to assertions schemas.
- ▶ Changes to policies also require cache flushes, if the result caching is enabled on the Dynamic Assembly components of interest.

**Problem:** Cannot install Fabric Foundation Pack on top of WebSphere Application Server V6.1.

**Solution:** The Fabric Foundation Pack must be installed on top of WebSphere Process Server V6.1.

## 13.3 Problem determination in underlying WebSphere Process Server, WebSphere Integration Developer infrastructure

In this section, we discuss problem determination in the underlying WebSphere Process Server, WebSphere Integration Developer infrastructure.

**Problem:** When you test an SCA-based application, which was exported as a SOAP/HTTP Web services binding, you might encounter the following exception.

IWAB0135E An unexpected error has occurred.

IOException

Connection refused: connect

**Solution:** This problem occurs because when an SCA-based application is exported as SOAP/HTTP Web services binding in WebSphere Integration Developer, the default\_host port given in the SOAP address location defaults to 9080; however, the default\_host port of the server on which the SCA-based application is running might be different. Figure 13-1 shows the properties of the SOAP/HTTP Web services binding of the export component in WebSphere Integration Developer.



Export: LoanProcessExport2 (Web Service Binding)	
Address:	http://localhost:9080/ITSOWeb/sca/LoanProcessExport2
Port:	LoanProcessExport2_LoanProcessHttpPort
Service:	LoanProcessExport_LoanProcessHttpService
Namespace:	http://com/ITSOBank/LoanProcess/LoanProcessInterface/Binding

*Figure 13-1 Export Properties*

Therefore, it is always to better to check the default\_host of the server on which the SCA application is running by navigating to **Servers** → **Application Servers** → **server1** → **ports** → **default\_host**. After the default\_host port of the server is noted, locate the WSDL that corresponds to the Web services export component in WebSphere Integration Developer by navigating to **[WIDProject]** → **Web Services Ports**. Open the WSDL in the Text Editor, and edit the following line to reflect the correct default\_host port:

```
<soap:address location="http://localhost:[default_host  
port_of_server]/ITSOWeb/sca/LoanProcessExport2"/>
```

To open the WSDL in the text editor, right-click the port under **Web Service Ports** node, and go to **Open With** → **Text Editor**.

After making this change, build the project, and restart the project in the server.

**Problem:** The back end service is implemented as an SCA module that is exposed as a SOAP/HTTP Web services export binding. When you try to unit test the back end service provider from an external WS-BPEL module through the SOAP/HTTP import binding, it does not work.



**Solution:** This happens on a rare occasion. However when you encounter this problem, the best practice is to delete the SOAP/HTTP Web services export binding and its associated WSDLs, and generate a new SOAP/HTTP Web services export binding for the back end SCA application.

**Problem:** Work Area service is not enabled.

**Solution:** WebSphere Process Server ships with the Work Area service feature enabled; therefore, we advise you not to turn this feature off, even if it is not in use because turning it off affects context propagation.

## 13.4 Common issues

In this section, we discuss the common issues that the Fabric developer faces while assembling the Fabric solution.

**Issue:** Unable to see Fabric projects in the Fabric Composition Studio after replication is successful.

**Solution:** When you start a new WebSphere Integration Developer workspace and create a first Composition Studio project, some configuration is needed, which includes server host, port (usually localhost:9080 for UTE), and a username/password pair.

Make sure that the user that logs in belongs to the project team/organization that owns the Fabric project.

**Issue:** HTTP basic authentication for Web Services

With a static Web Service binding to an endpoint, you can set credentials for HTTP basic authentication in the EJB™ deployment descriptor.

With Fabric, a concrete binding is not known ahead of time and different credentials might be necessary for different possible endpoint candidates. Where and how can you specify these credentials? On the other side. You cannot define credentials at the Fabric side because there might be different clients that invoke an endpoint.

Can you add credentials somehow using the extension interfaces of the DA SCA component?

**Solution:** You could write a "wrapper" SCA component that invokes the actual endpoint with HTTP basic authentication credentials.

The "wrapper" service is endpoint registered within Fabric and invoked by Dynamic Assembler.

**Issue:** Need to restart the Fabric server after adding or modifying Web services.

**Solution:** The Dynamic Assembler caches metadata from the BSR for performance reasons.

Also, if you pass the following entry in the SOAP Header, the Dynamic Assembler does not rely on the cache:

```
<property  
name="http://www.webifysolutions.com/context/debug">true</property>
```

Stop and start the Fabric\_Engine EAR file in the WebSphere Process Server Admin Console.

## 13.5 General best practices and recommendations

In this section, we discuss best practices and recommendations while you develop, deploy, manage, and govern a Fabric solution.

**Best Practice:** Select the correct use case while implementing a proof of concept or performing a technology evaluation of Fabric.

**Explanation:** Selecting a use case with more points of variability makes it easier to demonstrate the value of Fabric. The real essence of Fabric is brought out when you extract the variability from the complex business process and manage and govern those complexities by using meta models that are stored in the Business Service Repository. The idea of using Fabric is to drive changes by altering configuration (meta models) rather than by changing the design.

Selecting a linear business process as a use case does not demonstrate how complexities are reduced using Fabric and how a Fabric solution can readily adapt to changing business needs.

**Best Practice:** Unit test all back end applications and service end points before you route those invocation calls through the Dynamic Assembler component of the Fabric and perform an integration test.

**Explanation:** It is critical to unit test all back end application and service endpoints before invoking them through the Dynamic Assembly layer in Fabric. If the services deployed at those end points do not function correctly, the invocation failure can sometimes be wrongly interpreted as an issue with the Dynamic Assembler component in Fabric. Many times, developers suspect the issue to be with the Policy Engine of Fabric, when the actual issue is with the back end services.

You can test the back end service end points with the Web Services Explorer facility that is available in WebSphere Integration Developer.

**Best Practice:** Perform Policy Simulation before actual testing the Composite Business Application.

**Explanation:** It is a good practice to use the Dynamic Assembly Policy Simulator in Fabric Composition Studio to make sure that all policies and assertions are authored correctly, which makes sure that errors, if any, in endpoint selection are detected early in the development life cycle and any changes are effected immediately followed by another round of policy simulation.

**Best Practice:** Use implicit SOAP headers to pass the context information or use explicit SOAP headers. The context information is required for the Dynamic Assembler to invoke the correct service implementation.

**Explanation:** The context information that the Dynamic Assembler requires can be passed in two ways:

► Explicit SOAP headers

In this method, the context information is passed to the business process as a header element in the SOAP request. However this method involves changing the interface of the business process to explicitly add `ContextTypeExt` as an input in the interface WSDL of the business process. `ContextTypeExt` is a complex object that is defined in `prism-context-1.0.xsd`. It carries the context information as a part of the SOAP request.

To see how this technique works, refer to the following article:

[http://www.ibm.com/developerworks/websphere/library/techarticles/0708\\_balani/0708\\_balani.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0708_balani/0708_balani.html)

► Implicit SOAP headers

In this method, the context information is injected implicitly to the SOAP request message. We implemented the scenarios in this book using this technique. This technique does not involve any changes to the interfaces and is a non-intrusive way to inject the SOAP headers.

**Best Practice:** Do not include a Fabric Project when you export a Project Interchange File.

**Explanation:** You could get an error indicating that the WebSphere Business Services Fabric Project does not exist, if you export all modules in a workspace that includes a Fabric project. If this happens, you are returned to the Export Project Interchange File window, where you click **Finish**, which should make it complete without any more errors.

**Best Practice:** Context specifications - use an option called "Only use specified context at runtime".

**Explanation:** On the Dimensions tab in the Context Specification editor, there is an option called "Only use specified context at runtime".

- ▶ When this option is selected, only those dimensions that were specified as required or optional in the Context Specification are used for decision-making.
- ▶ When this option is not selected, we know that the context-specification tells the full story of what is provided at runtime. When this selection is unchecked, new context properties can affect how policies are composed.

It is a best practice to use context-specifications that are configured to only use the specified context.



# Industry Content Packs

In this chapter, we provide an overview about Industry Content Packs. We also address some key questions, such as:

- ▶ What are Industry Content Packs?
- ▶ Why do we need Industry Content Packs?
- ▶ How to leverage Industry Content Packs in accelerating the composite business application delivery.

## 14.1 Overview of Industry Content Pack

*Industry Content Packs* are pre-built, configurable business accelerators that contain industry-specific SOA assets and govern architecture that speeds up the delivery of composite business applications, which leverage and extend the IBM WebSphere Business Services Fabric platform. The SOA assets are primarily derived from industry-specific standards and best practices.

The benefits of using Industry Content Packs are:

- ▶ Designed for specific industry participants to facilitate more rapid service-oriented business solution assembly.
- ▶ Optimized for industry best practices, which ensures consistency and reuse across geographical locations, product lines, and processes.
- ▶ Helps to simplify interoperability by defining the common language to facilitate interoperability between disparate IT assets. These pre-packaged, industry-specific vocabularies are expressed as extensible UML models, for example, HL7, ISO 20022, and NGOSS SID.
- ▶ Helps speed time-to-market with pre-built, frequently reused, and industry-standards specific service implementations for common transactional functions, such as validation and error identification.
- ▶ Enables standardized connectivity to disparate applications in the IT ecosystem with pre-built service interfaces that are based on industry standards.
- ▶ Helps speed time-to-market using pre-built business metadata that ensures consistency and that maintains the standards compliance during the definition and development of business service policies.
- ▶ Expands the use within an enterprise or enhances the security of publishing to a broader partner ecosystem. Enables clients and partners to plug in their SOA assets into predefined Business Service Templates.

As an offering of WebSphere Business Services Fabric, the four Industry Content Packs include:

- ▶ IBM Insurance Property and Casualty Content Pack
- ▶ IBM Healthcare Payor Content Pack
- ▶ IBM Banking Payments Content Pack
- ▶ IBM Telecom Operations Content Pack

We use the IBM Banking Payments Content Pack in this chapter to explain, in detail, how to use the Industry Content Pack.



**Note:** For more information about each of the Industry Content Packs, refer to the documentation that is packaged within the installers of the respective Industry Content Packs.

## 14.2 Industry Content Pack reference architecture

An Industry Content Pack is aimed at accelerating delivery of industry-specific service-oriented business solutions using the IBM WebSphere Business Services Fabric platform. The framework of components in the Industry Content Packs is a logical expansion of SOA, which promotes loose-coupling of assets and thus enables consistency and reuse across business processes. The inconsistency in the current IT ecosystems of companies limits their ability to use industry and technical standards across the enterprise, support common needs, and encourage consistency in the technical architecture of the standards. The framework is optimized to industry and technical standards, which facilitates interoperability across the ecosystem.

The framework is expressed as the reference architecture, with a goal of enabling the customers to extend it to meet their unique business needs. These components work in harmony to create a reference architecture foundation and to help you build their own specific business logic for competitive advantage. You can also add your own components to this reference architecture and model it based on your business demands.

The reference architecture is based on the following key asset types of Industry Content Packs:

- ▶ Industry Capability and Process Maps
- ▶ Industry Business Service Templates
- ▶ Industry Service Interfaces
- ▶ Industry Business Glossary
- ▶ Industry Common Services
- ▶ Industry Business Object Model
- ▶ Knowledge Assets

We explain each of these asset types in 14.3, “Types of assets” on page 373.

Figure 14-1 on page 372 shows the reference architecture for Industry Content Packs.

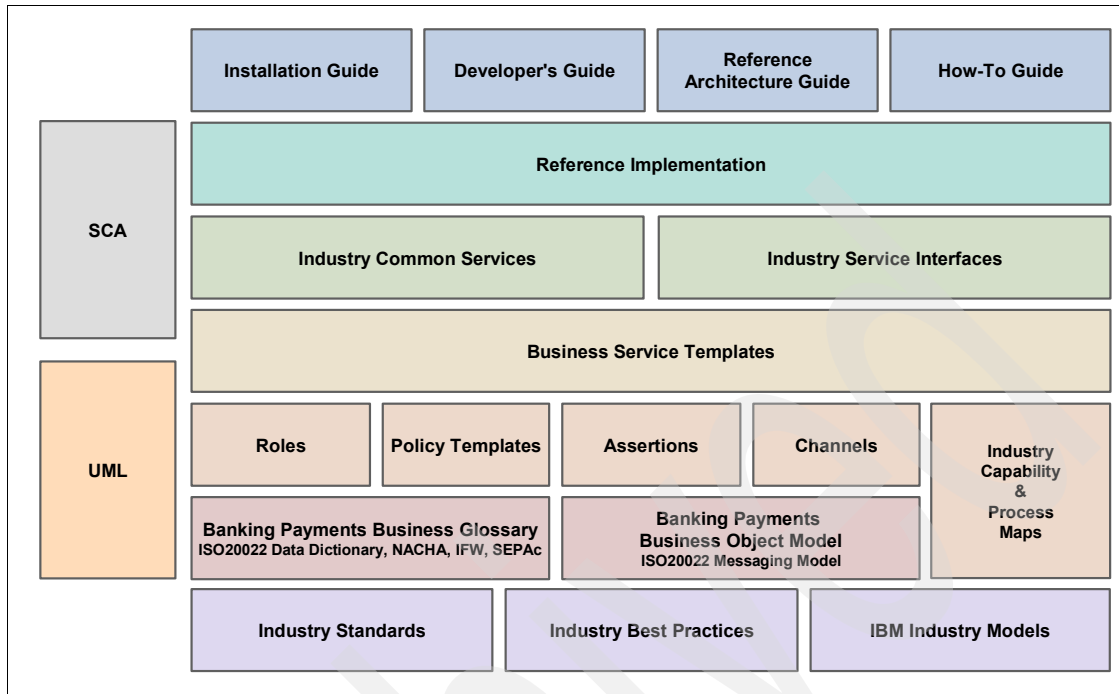


Figure 14-1 Industry Content Pack reference architecture

The reference architecture for Industry Content Packs serves as a generic framework for building industry-specific Content Packs. Using this architecture, Industry Content Packs are developed using industry-specific standards. These standards form the basis for deriving various assets in the Industry Content Packs.

The *Industry Business Service Templates* are based on industry standards and industry best practices, and they also act as a starting point for developing a service-oriented business solution. For Business Service Templates, the associated metadata-like channels, roles, assertions, and sample policies are defined. The *Industry Business Glossary* and the *Industry Business Object Model* are derived from industry standards, such as ACORD, HL7, NGOSS SID, and ISO 20022.

Service Component Architecture (SCA) is a set of specifications that describe a model for building applications and systems on an SOA platform. From a service-oriented business solution perspective, SCA is now being accepted as a standard programming model. The Industry Content Packs support the SCA programming model by packaging the Business Service Templates as SCA modules. The *Industry Common Services* and *Industry Service Interfaces* are

made available as SCA libraries that you can use as part of other SCA components, such as WS-BPEL. The SCA modules that are created as part of each Content Pack provides integration to the WebSphere Business Services Fabric by illustrating the use of IBM Business Services Dynamic Assembler within a reference implementation. Knowledge assets, such as the *Developer's Guide* and the *How-To Guide* provides information about using and extending Industry Content Pack assets.

## 14.3 Types of assets

The following topics details each asset types in the context of the reference architecture.

### 14.3.1 Industry Capability and Process Maps

The *Industry Capability Map* provides a logical view of business competencies that an industry is expected to possess. The *Industry Process Maps* provide a logical view of the business processes that an industry uses:

- ▶ The Industry Capability and Process Maps provide the domain decomposition of the industry.
- ▶ The Capability Map defines the capabilities and the sub-capabilities for the industry.
- ▶ The capabilities can be broken down to a granular level to identify the sub-capabilities. They can also be decomposed to n-levels in the hierarchical representation.
- ▶ The roots in the hierarchy are the capabilities that are represented in the Industry Capability Maps.
- ▶ The Industry Process Maps define a hierarchical representation of the business processes in an enterprise and can be decomposed to n-levels in the hierarchical representation.
- ▶ The Industry Capability Maps are mapped to the Industry Process Maps.

There is a many-to-many relationship between capabilities and processes. Under a specific capability, an n-level grouping of business processes can exist. The various processes and sub processes that are required to support the capabilities are identified.

The capability and process maps can be extended to meet your unique business needs.

The Industry Capability and Process definitions are based on industry standards, such as eTOM, TAM, ISO 20022, ACORD, HL7, or industry best practices.

The Industry Capability and Process Maps enable you to have a top-down visibility into the business by identifying the business capabilities and processes that are consistently reusable by mapping capabilities, processes, and business services; therefore, the Industry Capability and Process Maps demonstrate business-to-IT alignment.

### 14.3.2 Industry Business Services Templates

The Industry Business Service Templates represent the starting point for an industry specific service-oriented business solution.

The Industry Business Service Templates provide the business level building blocks that you can reuse across multiple solutions and processes. Based on business capabilities and business processes, an industry domain is decomposed into business functions that are represented by business services. You can use these business service representations as-is, or you can extend them. These business services adapt their behavior at runtime, based on associated business metadata that includes channels, roles, assertions, and policies.

A Business Service Template consists of the definition of the business service, relevant subscribers for whom the service can be personalized, potential mode of access around which the service can be accessed, assertions to represent its business capabilities, sample policies to define its behavior, and mappings to the associated Web services.

The granularity of Industry Business Service Templates is based on the way business functions are represented in a given industry domain in the context of an SOA solution. Typically, the business services decomposition is based on business capability models and business process models or domain knowledge of subject matter experts (SME) who also understand the principles and methodologies of SOA. After the decomposition, these business services are defined with associated metadata. Thus, there is no as-is usage of any source content (capability and process models) while you define business service templates. In this case, the source content is, at best, used as a reference.

Representing a business process by a set of business services enables shorter change cycles and improves reuse across the enterprise. Business service templates are based on industry best practices and supported by prevalent industry standards. These templates reduce the time and cost to implement business services and enable simpler and more flexible business processes.

They also enable you to plug-in your SOA assets to predefined Business Service Templates.

### 14.3.3 Industry Service Interfaces

The Industry Service Interfaces are prebuilt Web service interfaces that facilitate interoperability across heterogeneous platforms in a given industry.

A key challenge that many enterprises face today is interoperability across heterogeneous systems in their ecosystem, which includes packaged applications, custom applications, established systems, object-oriented applications, and third-party services. All of these disparate assets need operate with each other and the industry service interfaces enable this interoperability. These interfaces are based on industry standards and ensure consistency and reuse.

The Web service interfaces in industry service interfaces are based on industry standards. In most cases, these industry standards provide messaging schemas that define the industry service interfaces. In those cases where the industry standards only define the data types, a custom schema is created to define the industry service interfaces. Similar to the Industry Business Service Templates, the granularity of Industry Service Interfaces is also extremely important, and it is based on the way specific industry domains are decomposed into a set of operations, which is based on business services and the types of interactions that take place across IT ecosystems in an industry. Typically, the industry service interfaces are atomic Web services and fine-grained, which means they can perform a single task. Based on a your unique business needs, multiple industry service interfaces can be orchestrated into a composite service.

Pre-defined Industry Service Interfaces enable interoperability across disparate IT assets using prevalent industry standards, which reduces time and cost and improves consistency.

### 14.3.4 Industry Business Glossary

The Industry Business Glossary is a taxonomy of business terms in an industry. It also provides information about relationships with other industry-specific terms.

WebSphere Business Services Fabric enables interoperability of heterogeneous systems in an enterprise ecosystem through business services metadata that includes roles, channels, assertions, and policies. To enable the consistency and reuse of the associated business services, it is very important to use industry standards to define the metadata. The Industry Business Glossary is a common

vocabulary that represents the taxonomy of business terms from various industry standards.

The Industry Business Glossary is based on an industry standard that represents business objects or data types. You can source the content for the Industry Business Glossary from any of the data standards, such as BOM models, messaging models, and industry dictionaries. An Industry Business Glossary can source content from multiple industry standards, which allows you to choose one of the standards as an enterprise-wide information or canonical model for business services.

The Industry Business Glossary provides predefined and extensible business services metadata (based on industry standards) that serves as a framework to improve business services consistency and reuse.

### 14.3.5 Industry Common Services

The Industry Common Services are industry-specific service implementations that enable transactional functions, such as validation, error identification, and transformation.

In an enterprise ecosystem that consists of heterogeneous applications, there are common industry transactions, such as validation, transformation, and so on, that you need to perform when various business services are instantiated. These transactions are relevant to communication between you and any partners and also within the four walls of your organization. There is a need to provide standards-based implementation of common message and document processing, and Industry Common Services meet this requirement.

The Web service interfaces in Industry Common Services are based on industry-standards. In most cases, these industry standards provide the messaging schemas that define the Industry Common Services. Wherever the industry standards only define the data types, a custom-schema that is based on industry best practices is created to define the Industry Common Services. The granularity of Industry Common Services is based on the kind of common message and document processing transactions that are performed across IT ecosystems in an industry. Typically, the Industry Common Services are atomic Web services and fine-grained, which means that they can perform a single task. Based on your unique business needs, you can orchestrate the Industry Common Services into a composite service.

The Industry Common Services provide pre-built service implementations for industry-specific common functions that you can reuse across multiple composite business applications and thus jump-start delivery of the SOA applications.

### 14.3.6 Industry Business Object Model

*Industry Business Object Model* (Industry BOM) is a logical representation of business entities and attributes that constitute a business domain or a sub-vertical that is within a domain.

The primary purpose of this model is to serve as an input to the logical data model for a business domain. This model is typically sourced from industry standards, such as HL7, ACORD, NGOSS SID, and ISO 20022 model. The BOM contains business concepts, relationships, and enumerations that you can progressively engineer into a tool, such as Rational Data Architect, to construct a logical data model. As part of Industry Content Pack, they are also used as a source for deriving business terms within the Industry Business Glossary.

The source of the Industry BOM could be any industry standard that has a representation of business entities, attributes, and relationships as a BOM, data dictionary, and messaging model.

The Industry BOM provides a de normalized version of business entities attributes along with its relationships, which paves the way towards the creation of an LDM, which you can use to create a physical data model.

### 14.3.7 Knowledge Assets

The *Knowledge Assets* in an Industry Content Pack are intended to make it easy for you to consume and extend the assets that are packaged in the pack to accelerate development of an industry-specific service-oriented business application.

The Knowledge Assets include the following artifacts:

- Installation Guide

Provides the installation procedures for an Industry Content Pack and covers the platform, software, and hardware prerequisites for installation. It also describes both the manual and interactive installation processes for installing the Industry Content Pack.

- Developer's Guide

Provides the technical details of various types of assets that are packaged in an Industry Content Pack. It describes the tooling, architecture, and deployment details of these assets.

- Reference Architecture

Provides the reference architecture for an Industry Content Pack. It describes the framework of components for an Industry Content Pack. It also explains

the alignment of reference architecture of an Industry Content Pack with IBM SOA architecture and methodology.

► How-to Guide

Provides the usage and extensibility of an Industry Content Pack. It describes the application of the Industry Content Pack assets in accelerating the development of a SOA solution. It also covers the details surrounding extensions that you can make to Content Pack assets to meet your unique customers' needs.

► Reference Implementation

Because an Industry Content Pack is an accelerator for development of a service-oriented business application and not the application itself, it is important to take some of the assets and showcase a sample implementation. The Reference Implementation is based on a sample process that is specific to the Industry Content Pack and uses some of the assets (such as one Industry Business Service Template, five Industry Common Services, ten Industry Service Interfaces, and two Roles, three Channels, and three Assertions from the Industry Business Glossary) to demonstrate an end-to-end WebSphere Business Services Fabric-enabled service-oriented business application. The reference implementation is available as an SCA module that is typically a BPEL process, which you can import into WebSphere Integration Developer and execute within WebSphere Process Server. The Reference Implementation leverages all of the other components in the Content Pack with the exception of BOM. Using mock implementations for Industry Service Interface, they demonstrate the use-case and how the Industry Content Pack integrates with the IBM Business Services Fabric Dynamic Assembler component that is available as part of WebSphere Business Services Fabric Foundation Pack.

The Knowledge Assets serve as a guidance mechanism to help set the boundary for deployment of a service-oriented business application using WebSphere Business Services Fabric and the Industry Content Packs.

Table 14-1 summarizes the tooling artifacts and the tools that you can use to create the respective Industry Content Pack assets.

Table 14-1 Tooling artifacts

Type of asset	Design time	Runtime
Industry Capability and Process Maps	Artifact: UML Models Tool: Rational Software Architect	Artifact: Fabric Content Archive (FCA) Tool: WebSphere Business Services Fabric (Fabric)



Type of asset	Design time	Runtime
Industry Business Service Templates	Artifact: SCA module Tool: Fabric Composition Studio, WebSphere Integration Developer	Artifact: FCA Tool: Fabric
Industry Service Interfaces	Artifact: SCA library, WSDL with XSD Tool: WebSphere Integration Developer	
Industry Business Glossary	Artifact: UML Models Tool: Rational Software Architect., Fabric Modeling Tool	Artifact: FCA Tool: Fabric
Industry Common Services	Artifact: SCA Libraries, WSDL with XSD Tool: WebSphere Integration Developer	Artifact: Deployable EAR, FCA Tools: WebSphere Process Server, Fabric
Industry Business Object Model	Artifacts: UML Models Tool: Rational Software Architect.	
Knowledge Assets	Artifacts: Installation guide, Developer's Guide, Reference Architecture and How-to Guide available in PDF and HTML format	Artifact: Deployable EAR, FCA Tools: WebSphere Process Server, Fabric (This is for the reference implementation)

**Note:** For further details about the Industry Content Packs alignment with IBM SOA Reference Architecture and IBM SOA life cycle methodology, refer to the Architecture Guide that is available as part of the installed documents.

## 14.4 Banking Payments Content Pack

*The Banking Payments Content Pack* supports payment capabilities for financial institutions. It consists of Banking Payments' specific Business Services Templates that include the following pre-built assets:

► Banking Payments Capabilities & Process Maps

This asset includes the Banking Payments-specific business capabilities and process maps that are based on ISO 20022 standards and Banking Payments best practices.

► Banking Payments Business Services Templates

This asset includes the Banking Payments-specific business services definitions and the associated metadata that includes roles, channels, assertions, and sample business policies.

► Banking Payments Service Interfaces

This asset includes the Banking Payments specific data types and service interfaces that are based on the subset of the IFW industry model and HR-XML model.

► Banking Payments Business Glossary

This asset includes the banking taxonomy that is derived from banking standards, such as ISO 20022 payments model, ISO 20022 Data Dictionary, NACHA, and IFW model.

► Banking Payments Business Object Model (BOM)

This asset includes the ISO 20022 Data Dictionary Version 5.1.2.1 and the ISO 20022 payments model Version 1.10 based on the Banking Payments BOM.

► Banking Payments Common Services

This asset includes the Banking Payments-specific common Web service implementations using ISO 20022 messaging standards, SEPA standards, SEPA-AOS standards, and NACHA standards as well as the SEPA standards, SEPA-AOS standards, and NACHA standards.

**Note:** Refer to the Banking+Payments+Pack+6.1+Installation+Guide.pdf for installation steps for the IBM Banking Payments Content Pack for WebSphere Business Services Fabric v 6.1.

## Supported standards

The supported standards are:

- IFW: A subset of the BOM concepts that represent the concepts.
- ISO 20022: A subset of terms that represents the concepts. (Deprecated as of Version 6.1)
- ISO 20022-Data Dictionary: The version is as of October 2006, approved by UNIFI Payments SEG. The version is 5.1.2.1.

- ▶ NACHA: A subset of terms that represent the concepts. The version is as of 2007 ACH record format specification.
- ▶ SEPA: A subset of terms that represent the concepts that are based on version 2.2.

## Overview of the IBM Information Framework model

The *IBM Information Framework (IFW) BOM model* is a source to derive the glossary terms for the IBM Banking Payments Content Pack. Additionally, the IFW model has well-defined service interfaces and data types that are part of the Interface Design Model (IDM). A logical subset of Web service interfaces and definitions that are pertinent to the sub-vertical payments were chosen to define Industry Web Service Interfaces. These interfaces and data types are logically grouped into SCA libraries that are used to implement the SCA modules. From a WebSphere Business Services Fabric perspective, a subset of the IFW BOM terms and attributes are marked to derive the Banking Payments Glossary Model. The Banking Payments Glossary Model is then used to create the IFW-based assertions. These assertions are used to model the Business Policies' templates that are made available as part of the Business Service Templates.

## Overview of the ISO-20022 Payments model

Payment initiation messages are a part of the Payments Standards Evaluation Group as UNIFI messages, as of June 2006. This set includes the following messages:

- ▶ Payment messages:
  - Customer Credit Transfer Initiation
  - Customer Direct Debit Initiation
- ▶ Related messages:
  - Customer Payment Reversal
  - Payment Cancellation Request
  - Payment Status Report

In the IBM Banking Payments Content Pack, the messages to define Common Services. Additionally the Banking Payments Business Glossary uses the ISO 20022 terms and concepts.

We suggest that mediation patterns, such as the Enterprise Service Bus (ESB), will use the concepts and taxonomies that are defined in ISO 20022 Payments Initiation standard directly or indirectly to transform data from and to the various target industry models.

## Overview of the NACHA ACH standards

The NACHA ACH standards contain messages that pertain to electronic payments. For the IBM Banking Payments Content Pack, the relevant glossary terms from NACHA are derived with the intention of creating assertions that are based on NACHA. The ACH record format, which is defined as part of the NACHA specifications, is designed to assist ACH participants in formatting and retrieving transaction information. In the IBM Banking Payments Content Pack, the NACHA terms and attributes that are derived out of different ACH record formats define the glossary model. This glossary is used as input to derive the NACHA based assertions. These assertions are then used to model business policies templates that are available as part of the Business Service Templates.

## Extensions to the industry standards used in Banking Payments Content Packs

The models in the following list are defined using different namespaces:

- ▶ IFW-based extensions model contains classes and instances that are not defined in the IFW Glossary model. This model also includes enumerations that are sourced from the IFW-BOM model.
- ▶ ISO 20022-based extensions model contains classes and instances that are not defined in the ISO 20022 Glossary model. This model also includes enumerations that are sourced from Payments schema.
- ▶ NACHA-based extensions model contains classes and instances that are not defined in the NACHA Glossary model. This model also includes enumerations that are sourced from ISO 20022 Payments schema.
- ▶ ISO 20022 Data Dictionary-based extensions model contains classes and instances that are not defined in the ISO 20022 Data Dictionary Glossary model. This model also includes enumerations that are sourced from ISO 20022 Data Dictionary schema.
- ▶ SEPA-based extensions model contains classes and instances that are not defined in the SEPA Glossary model. This model also includes enumerations that are sourced from SEPA Payments schema.

### 14.4.1 Leveraging the Banking Payments Content Pack

In this section, we describe the reference architecture and the key asset types for the Banking Payments Content Pack. Figure 14-2 on page 383 illustrates the instantiation of Reference Architecture for the Banking Payments Content Pack.

Figure 14-2 on page 383 shows the Banking Payments Content Pack.

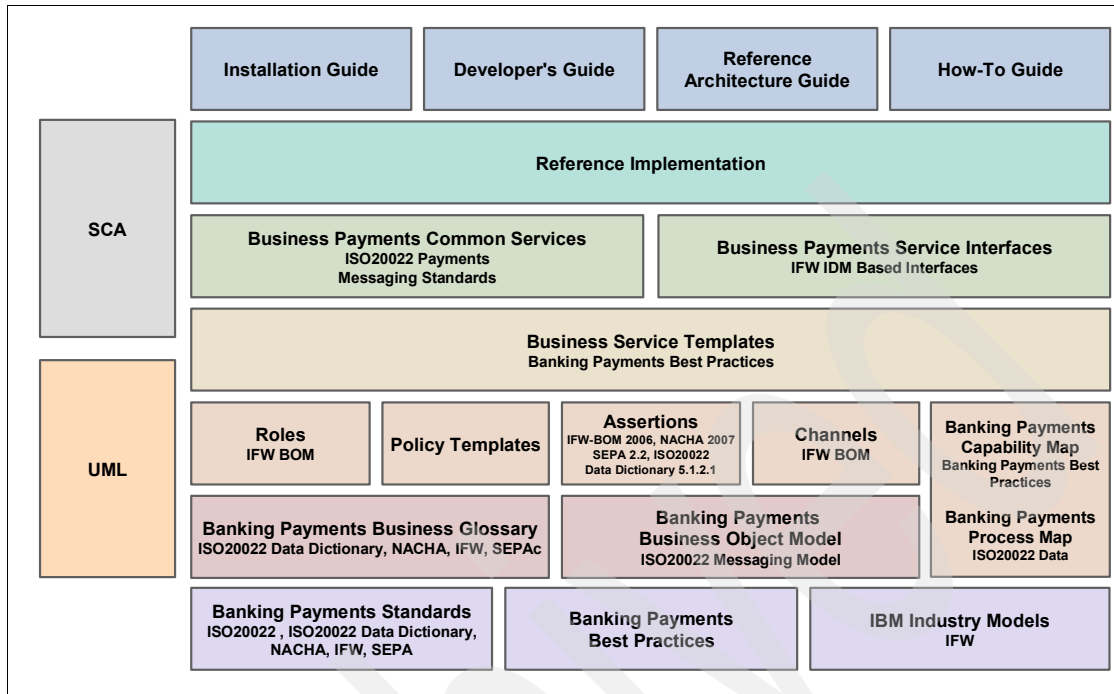


Figure 14-2 Banking Payments Content Pack Reference Architecture

## Assets packaged in the Banking Payments Content Pack

In this section, we provide details about each asset in the Banking Payments Reference Architecture, shown in Figure 14-2.

### ► Banking Payments Capability and Process Maps

The Banking Payments Capability is based on best practices in the banking industry, and Process Maps are based on ISO 20022 Data dictionary's processes.

### ► Banking Payments Business Service Templates

The definition and granularity of the Banking Payments Business Service Templates is based on the Capability and Process Maps, the application ecosystem in the banking industry, and applying SOA principles and methodologies for granularity of services.

### ► Banking Payments Service Interfaces

Banking Payments Service Interfaces are based on IFW and the best practices in the banking industry.

- ▶ **Banking Payments Business Object Model**

Banking Payments BOM is derived using ISO20022. This model provides an information or data reference model and a common information or data vocabulary from a business and a systems perspective. These models use UML to formalize the expression of the needs of a particular view. It provides consistent data and concept reuse across multiple information structures, including messages. It represents a rich set of concepts that must be shared throughout the banking systems.
- ▶ **Banking Payments Business Glossary**

Banking Payments business glossary is derived using ISO 20022 data dictionary, NACHA, IFW, and SEPA:

  - Roles are based on IFW BOM and ISO 20022 Data dictionary.
  - Assertions are derived from IFW, NACHA, SEPA, ISO 20022 Data Dictionary, and Banking Payments Best Practices.
  - Channels are based on the IFW BOM.
- ▶ **Banking Payments Common Services**

Banking Payments Common Services are based on ISO 20022 Payments Message and Banking Payments Best Practices, which also consists of reusable components that are commonly used across the banking industry.

### **Banking Payments Content Pack logical view**

Figure 14-3 on page 385 provides a logical view of the reference architecture for Banking Payments Content Pack. The previous topics addressed the various dependencies between each of the components. These components are visualized using association notation.

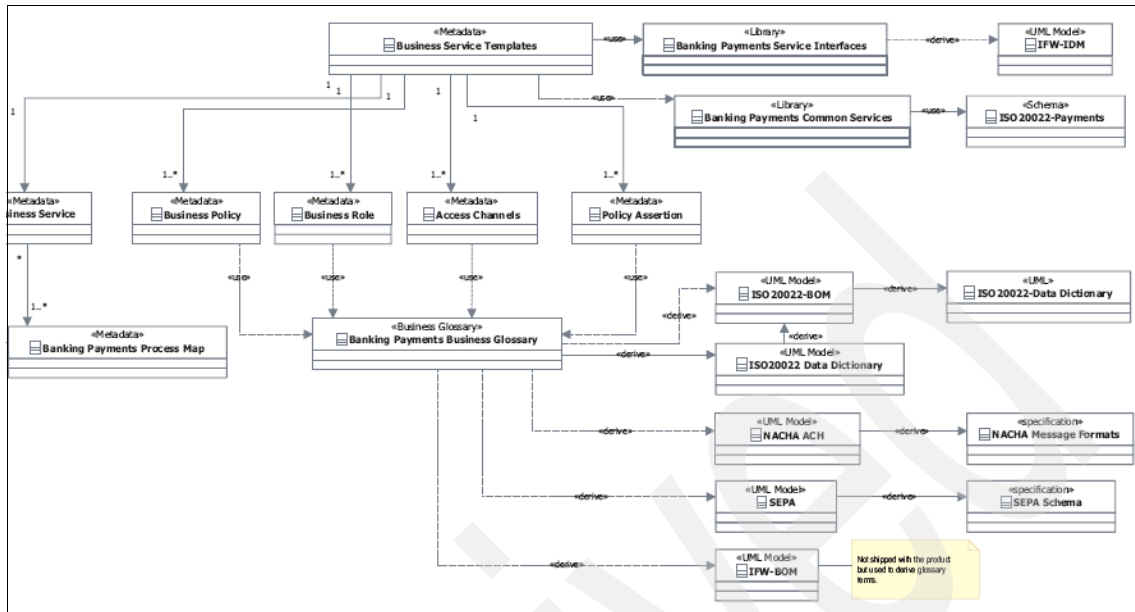


Figure 14-3 Banking Payments Content Pack logical view

## Banking Payments Content Pack component model

In this section, we describe how the Component Model contributes to the reference architecture. Figure 14-4 on page 386 illustrates the component model for the Banking Payments Content Pack. The top level stack demonstrates the runtime components and their dependencies, while the bottom stack shows the design-time components that are made available as part of the Banking Payments Content Pack.

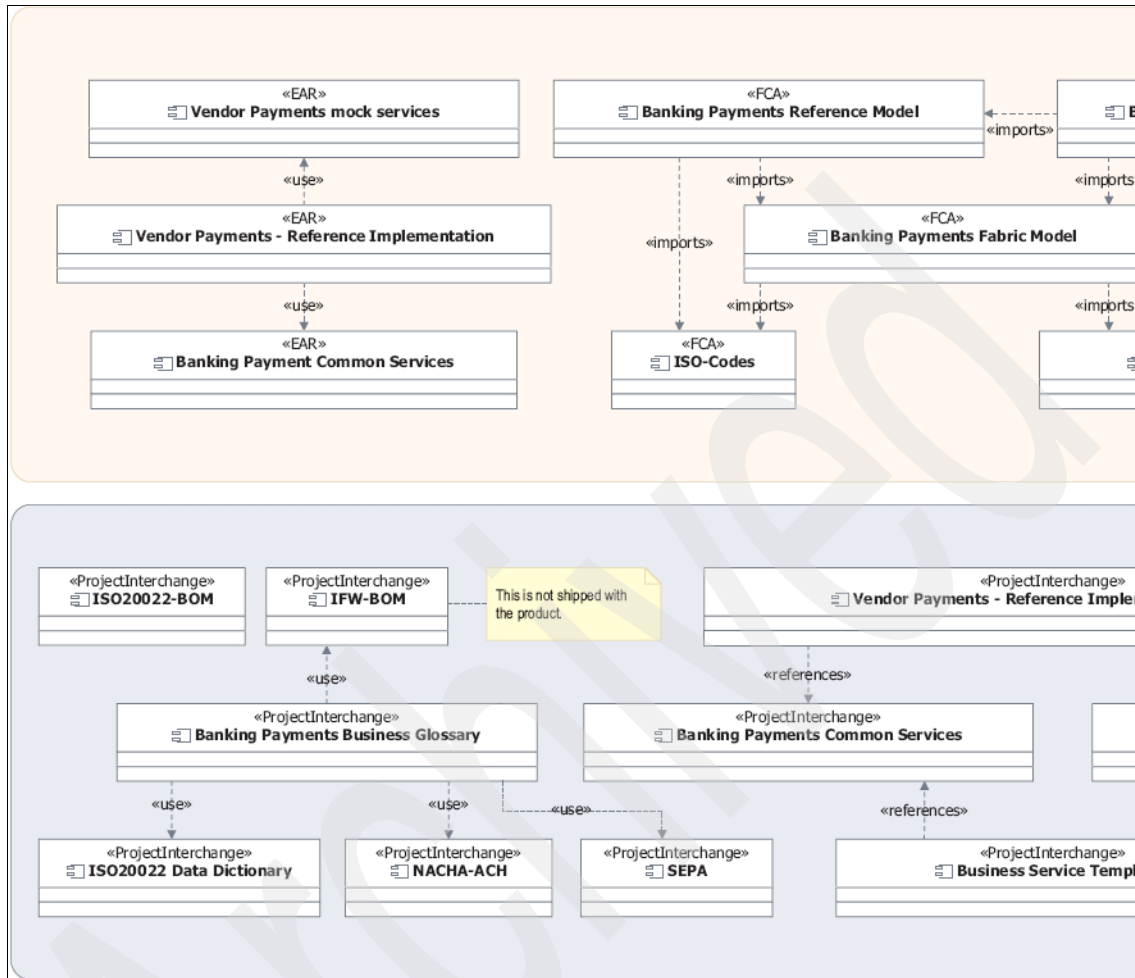


Figure 14-4 Banking Payments Content Pack Component Model

The components that are illustrated in Figure 14-4 include:

- ▶ Banking Payments Business Glossary

The UML model representation of the Banking Payments Business Glossary is packaged as Project Interchange, which you can import into Rational Software Architect. Architects and developers can use these UML models to create and extend industry core concepts.

- ▶ Banking Payments Common Services

The Banking Payments Common Services comprises the standards-based Web service implementation. The interfaces and data types for these



Common Services are made available as an SCA library for design-time, and they are deployed as an executable EAR file into the WebSphere Process Server at runtime.

- ▶ **Banking Payments Service Interfaces**

Banking Payments Service Interfaces in Banking Payments Content Pack are standards-based Web service interfaces that are designed using WebSphere Integration Developer. These Industry Service Interfaces are WS-I compliant, and they conform to the latest WSDL and XSD specifications. While there is no runtime component for these assets, they are packaged as SCA libraries, which are used as dependent projects for SCA modules that are available as part of Business Service Templates.

- ▶ **Banking Payments Reference Model**

Banking Payments Reference Model is comprised of Business Service Templates with its mapping to Process Maps, application suites, applications, roles, assertions, and service interfaces. This is distributed as FCA that you can import into WebSphere Business Services Fabric. Figure 14-5 on page 388 shows the various design-time and runtime components and the recommended tooling for various asset types.

Figure 14-5 on page 388 contains a list of assets with the recommended tooling.

Type of assets	Design time	Runtime	Tools
Banking Payments Capabilities and Process Maps	UML Models	Fabric Content Archive (FCA)	<ul style="list-style-type: none"> <li>• Rational Software Architect – Design-time</li> <li>• WebSphere Business Service Fabric – Runtime</li> </ul>
Banking Payments Business Service Templates	SCA Libraries and Modules, WebSphere Business Services Fabric Projects	FCA	<ul style="list-style-type: none"> <li>• WebSphere Integration Developer – Design time</li> <li>• WebSphere Business Service Fabric – Runtime</li> </ul>
Banking Payments Service Interfaces	SCA Libraries	None	WebSphere Integration Developer – Design-time
Banking Payments Business Glossary	UML Models	FCA	<ul style="list-style-type: none"> <li>• Rational Software Architect – Design time (Recommended Tooling)</li> <li>• WebSphere Business Service Fabric – Runtime</li> </ul>
Banking Payments Business Common Services	SCA Libraries	EAR	<ul style="list-style-type: none"> <li>• WebSphere Integration Developer – Design-time</li> <li>• WebSphere Process Server – Runtime</li> </ul>
Banking Payments Business Object Model	UML Models	None	Rational Software Architect RSA – Design-time (Recommended Tooling)
Reference Implementation	SCA Modules	EAR	<ul style="list-style-type: none"> <li>• WebSphere Integration Developer – Design-time</li> <li>• WebSphere Process Server – Runtime</li> </ul>

Figure 14-5 Asset Table with recommended Tooling

## Banking Payments Content Pack deployment view

Figure 14-6 on page 389 shows the Banking Payments Content Pack deployment view.

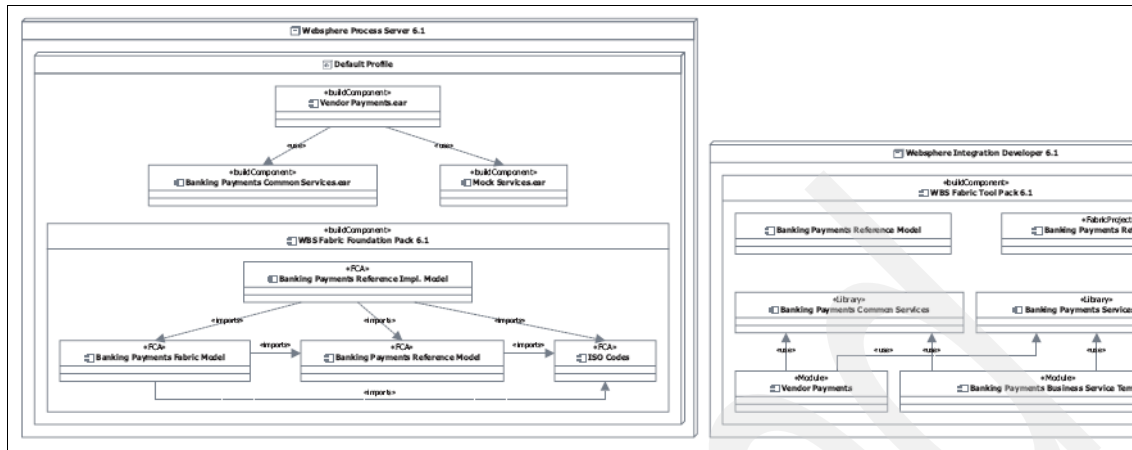


Figure 14-6 Banking Payments Content Pack deployment view

For the design-time, you can import and visualize the Banking Payments Business Service Templates using IBM WebSphere Integration Developer. The SCA modules and libraries are made available as project interchange files that you can import into the Business Integration perspective. You can import, visualize, and extend Banking Payments Business Glossary and Industry BOM using IBM WebSphere Business Services Fabric Modeling Tool within Rational Software Architect.

## 14.4.2 Reference implementation

In this section, we discuss how to use the Banking Payments Content Pack.

### Banking Payments reference implementation

To demonstrate Banking Payments Content Pack capabilities, a real industry use-case is taken into account and a reference implementation is provided by instantiating a Business Service Template from the content pack. The aim is to demonstrate the key aspects of the Banking Payments Content Pack, including Business Service Templates, Channels, Roles, and Assertions.

Ecosystem partners need to interact with other banking systems and networks to carry out a credit or a debit transaction. The partners of an ecosystem can use different modes to access the banking system. The inter-bank transactions take place across different networks, depending on geographies. The customer sends out a credit initiation request to a bank. The bank accepts it, processes it, and sends a payment status report back to the customer. For all of these communications, the ISO-20022 message standards are used. In addition, based on content in the request, Full Customer Details endpoints are defined.

This capability is provided by the use of IBM Business Services Dynamic Assembler and by use of content-based assertions. Similarly, the Payment Network endpoint selections are complete.

The high-level tasks that are involved are:

- ▶ Accept request from various channels: You can use different modes to access the business services. These channels can be Web portal, IVR, JMS, Business-to-Business, fax, mobile, and so on.
- ▶ Validate the request: The request is validated before any further processing.
- ▶ Unbundle the request: A valid request is then unbundled to get individual requests.
- ▶ Accept outpayment instruction: For each unbundled message, the individual transaction instruction request is accepted.
- ▶ Retrieve full customer details: Depending on the status, the customer details are retrieved.
- ▶ Evaluate customer risk: The customer risk is evaluated depending on the transaction history.
- ▶ Prepare out-payment: An out-payment request is prepared for this transaction instruction.
- ▶ Select the payment network: Depending on the creditor's country, the payment network is selected to which the payment is made.

**Note:** The payment status report is generated for all of the instructions that detail the status of the individual transactions, irrespective of whether they were approved or rejected.

## User roles

The following roles can access the reference implementation:

- ▶ Customer: The customer sends the payment request to the bank for processing.
- ▶ Treasury back office operator: The treasury back office operator reviews the amounts to verify that the bill amount does not exceed the customers' limit.

## Channel

The following channels can access the reference implementation:

- ▶ JMS Channel (Branch Channel)
- ▶ Web Channel (Web Portal Channel)

## Business policies

The following business policies are defined in the reference implementation:

The Web Portal Channel is available 24/7, while the branch channel is available Monday to Friday from 10 a.m. to 2 p.m.

Table 14-2 provides the business policies that are associated with the different states.

*Table 14-2 Plan to policy mapping*

State	Policy
TN	rFCDTX
TX	rFCDTN
Rest All	rFCDOther

The endpoint for Payment Network Selection is different for different country codes, as shown in Table 14-3.

*Table 14-3 Plan to policy mapping*

State	Policy
GB	SEPA
US	SWIFT
Rest All	SWIFT

## Common services

The common services that are associated with the reference implementation are:

- ▶ ISO-20022 Customer Credit Transfer Initiation Version 02 Validation Service
- ▶ ISO-20022 Customer Credit Transfer Initiation Version 02 Unbundle Service

## Web services

The Web services that are associated with the reference implementation are:

- ▶ Accept out-payment Instruction
- ▶ Retrieve Full Customer Details
- ▶ Evaluate Customer Risk
- ▶ Prepare Outpayment
- ▶ Payment Network Selection

## Design and process flow

To perform a credit transaction, the customer or a branch submits a bundled ISO 20022 Customer Credit Transfer Initiation Version 02 request document, with the appropriate details, to the banking system. This request undergoes the following processing to send out the status of the transaction in a Payment Status Report document to the customer:

- ▶ **Customer Credit Transfer Initiation Validation Service:** This service validates the ISO 20022 Customer Credit Transfer Initiation Bundled request. The validation is done for its adherence to its contract with the underlying XSD and for certain predefined rule sets.
- ▶ **Customer Credit Transfer Initiation Unbundle Service:** After the validation is performed successfully, that is with no errors, the unbundled service is invoked. This service returns a set of individual transaction initiation requests.
- ▶ **Accept Out-payment Request:** This service accepts the initiation request.
- ▶ **Retrieve Full Customer Details:** This service retrieves the complete details of the customer.
- ▶ **Evaluate Customer Risk:** This service evaluates the customer risk.
- ▶ **Prepare out-payment:** This service prepares an out-payment request for the initiation request.
- ▶ **Payment Network Selection:** Based on the country, this service selects the network where the prepared out-payment request is sent to.

Figure 14-7 on page 393 shows the Reference Implementation Process flow.

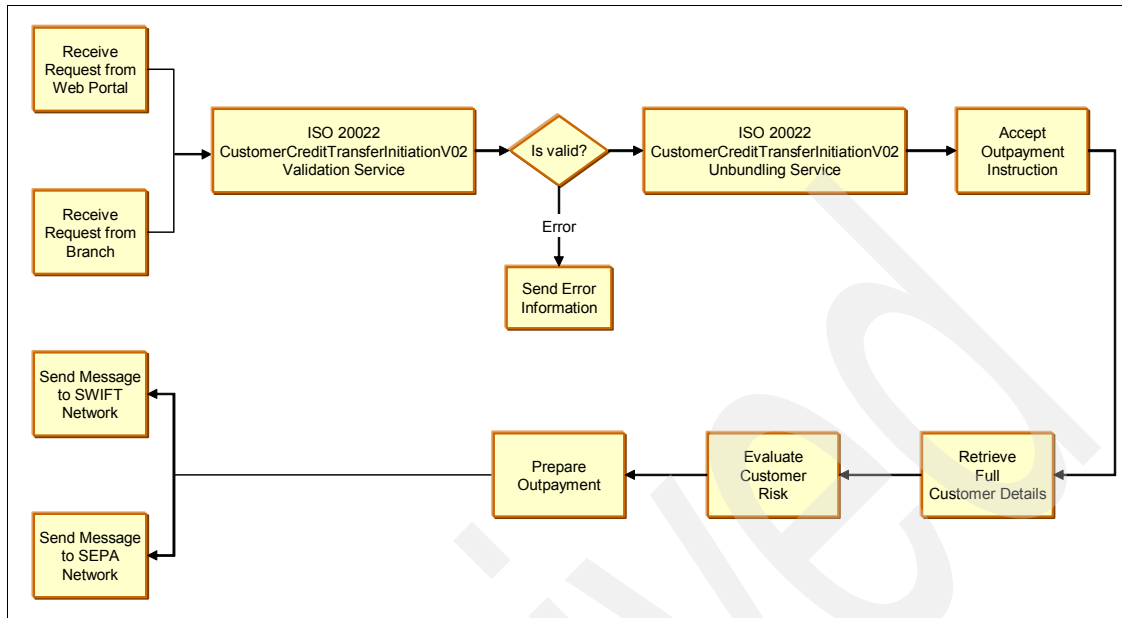


Figure 14-7 Reference Implementation process flow

### WS-BPEL process flow

The WS-BPEL process flow in Figure 14-8 on page 394 illustrates the WS-BPEL process flow for the reference implementation.

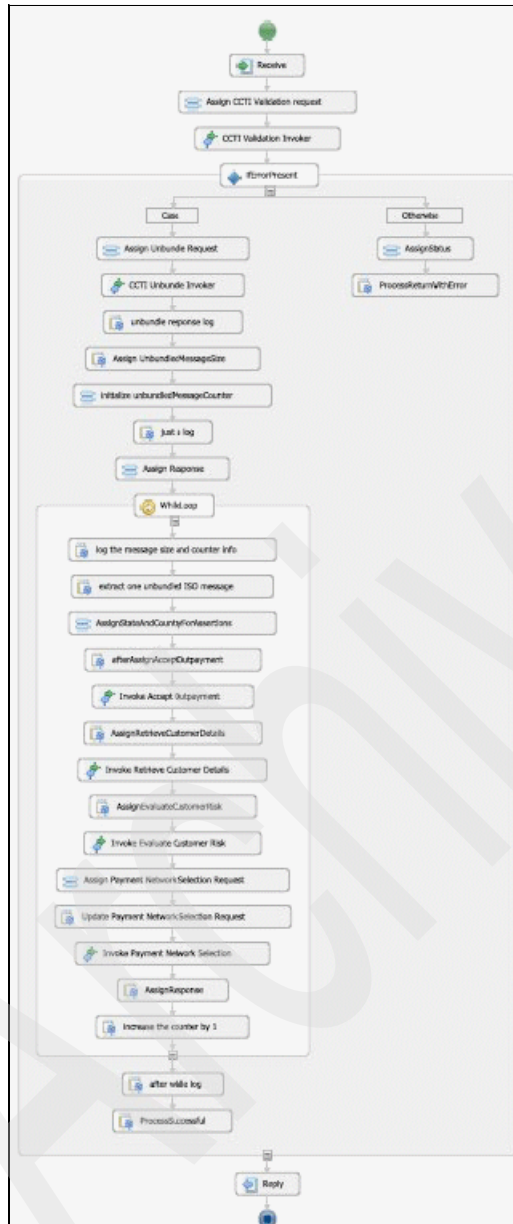


Figure 14-8 The WS-BPEL process flow



## Assembly diagram

Figure 14-9 shows the assembly diagram for the reference implementation.

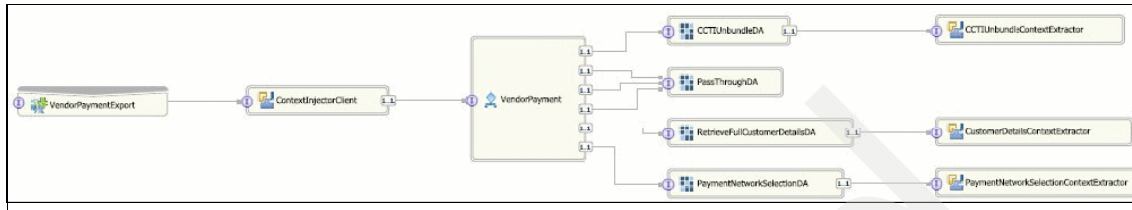


Figure 14-9 Assembly diagram for Reference Implementation

For more information about how the following components are implemented, refer to the *Banking Payments Pack 6.1 Developers Guide.pdf*, which is shipped with the IBM Payments Content Pack.

## Configuring the reference implementation

The configuration of the reference implementation involves setting up the reference implementation project interchanges and running them within WebSphere Process Server.

The following steps outline the procedures for configuring the Vendor Payments Reference Implementation within WebSphere Integration Developer:

1. Create a new workspace for WebSphere Integration Developer.
2. Import the VendorPayment-project-interchange.zip that is available within the installation location for the IBM Banking Payments Content Pack. See the *IBM Banking Payments Content Pack Installation Guide* for the specific location for reference implementation files.

Use the following steps to execute the Vendor Payments Reference Implementation that was already deployed within WebSphere Process Server:

1. As part of installation, the endpoints that are used within the reference implementation, by default, are set to the port 9080. If the reference implementation and Vendor Payments service-oriented business solution EAR are deployed on different ports, we recommend that you use a monitoring utility, such as TCPMon to act as a proxy to the actual endpoint.
2. Enrollments are not provided by default, which means that you must create the new enrollments before you execute the reference implementation, which you can do by creating a new user and assigning an appropriate role.
3. Give this new user a subscription to the Vendor Payments Business Service using IBM Business Services Subscriber Manager.

4. After the subscription in the IBM Business Services Subscriber Manager, copy the new subscription ID and include it as a part of the sample request that is provided within the testdata folder. To view the testdata folder, switch to the Resources perspective within WebSphere Integration Developer after you configure the Vendor Payments reference implementation project.

### **Invoking the reference implementation**

The process of invoking the reference implementation involves the following tasks:

**Note:** Perform the first two steps in WebSphere Business Services Fabric

1. Create an enrollment namespace.
2. Subscribe to the business service.
3. Invoke the service.
4. Verify the invocation in WebSphere Business Services Fabric.

#### ***Creating an enrollment namespace***

Refer to the IBM WebSphere Business Services Fabric for details about creating an enrollment namespace:

1. Logon to the WebSphere Business Services Fabric portal.
2. From the My Services menu, click **Governance Manager** → **Configure Namespaces**:
  - a. Click **Create a Namespace**, and enter a Display Name.
  - b. From the pull-down menu, in the Namespace Type field, select **Enrollment**, and enter the namespace URI (for example, `http://<domain name>/banking/enrollment#`).
3. From the Owner Project pull-down list, select **Vendor Payments Reference Implementation**.
4. Click **Create Namespace**, and the Enrollment namespace is created in WebSphere Business Services Fabric.

#### ***Subscribing to a business service***

In this section, we provide details about subscribing the Vendor Payments Business Service. To subscribe the Vendor Payments Business Service to the System organization:

1. Logon to WebSphere Business Services Fabric portal.
2. From the My Services menu, click **Subscriber Manager** → **Manage Subscribers**.

3. From the subscriber selection table, select the organization.
4. Click **Enrollments**, and select **VendorPaymentApp**, and then save the enrollment by clicking **Save Enrollments**.
5. Navigate to the Users tab, click the **admin** → **Subscriptions** tab.
6. From the Organization pull-down list, select **System**, and scroll down to select **VendorPaymentApp**, and click **Save Subscriptions**.
7. Click **View Subscriptions** to see the Subscription Details.

**Note:** See the *IBM Business Services Subscriber Manager* documentation for the details about subscribing to the business services.

### ***Invoking the service using a Web services client***

To invoke the Web services client such as Web Services Explorer or SoapUI:

1. Launch the SoapUI, and from the left pane, right-click **Project**.
2. Select **New WSDL Project**, enter the **Project Name** that is VendorPayment, click **OK**, and **Save**.
3. Right-click the created **VendorPayment** project, and select **Add WSDL from URL**.
4. Enter the URL  
`http://VendorPaymentWeb/sca/VendorPaymentExport/wsdl/VendorPaymentExport_VendorPaymentHttp_Service.wsdl`, and click **OK**.
5. Click **Yes**, and enter a valid request, a valid URL that is  
`http://<server>:port/fabric-engine/prism`, and submit the request using the IBM Business Services Dynamic Assembler.

**Note:** While submitting the request, proper header information should be provided in the WSDL file.

### ***Verifying the Reference Implementation in WebSphere Business Services Fabric***

To verify the Reference Implementation in IBM Business Services Performance Manager:

1. Logon to WebSphere Business Services Fabric portal.
2. Navigate to the My Services menu, click **Performance Manager** → **Service Invocation Summary**.
3. Click **Transaction Id** to see the graph.

## Testing and debugging the reference implementation

You can test the reference implementation within WebSphere Process Server and within the WebSphere Integration Developer UTE. In this section, we describe how to test the reference implementation from within WebSphere Integration Developer.

To begin, refer to the WebSphere Business Services Fabric for the details about creating an enrollment namespace.

To test and debug the reference implementation:

1. Launch WebSphere Integration Developer V6.1.
2. Switch to the Business Integration perspective.
3. Right-click, and select **Import**.
4. From the Import window, select **Project Interchange**, and click **Next**.
5. Click **Browse** to select the **Reference Implementation Project Interchange**, which is the VendorPayment-project-interchange.zip from the installed location.
6. Click **Select All**, and click **Finish**.
7. From the **Servers** tab, right-click **WebSphere Process Server**, and select **Add and Remove Projects**.
8. Select the available project to add, which is **VendorPaymentApp**, and click **Finish**. The server should start deploying the VendorPaymentApp into the process server.
9. From the menu bar, click **Window** → **Show View**.
10. Select **Other** from the sliding bar, and the Show View window pops-up. Click **Basic** → **Navigator**, and click **OK**.
11. From the Navigator View, click **VendorPayment**, and double-click **VendorPayment\_Test.wbiextrace** to open its' editor.
12. Under Detailed Properties, from the Initial request parameters, click **makePayment**. From the Context, select **context-identifier** to set the value to New, and click **Enter**. Set the right URI, which is **Context ID** and **Subscription ID** under Value to invoke this request through prism. Click **Continue**.

13. Select **WebSphere Process Server v 6.0**, and click **Finish**. It prompts you to enter the credentials.
14. Enter the user ID and password. Click **OK** to verify the implementation logic and variable values (input and output).
15. Verify the invocation in WebSphere Business Services Fabric.

## 14.5 Summary

In this chapter, we introduced concepts about the Industry Content Pack. We discussed the Industry Content Pack Reference Architecture and the type of assets that ship with the Industry Content Packs.

We also showed how you can leverage the Banking Payments Content Pack as a part of the Fabric solution. In this regard, we used a reference implementation that ships with the Banking Payments Content Pack. We understood the architecture behind the reference implementation and configured it to work our Fabric environment.





## Part 3

# Integration

Archived







## Integrating with WebSphere Service Registry and Repository

The ability to reuse services is one of the core benefits of any service-oriented solution. The question arises where and how created artifacts are stored and managed to maximize reuse, as well as how to support an appropriate set of rules, roles, and procedures to implement a governance model.

In this chapter, we show how WebSphere Service Registry and Repository Version 6.1 enables the publication, retrieval, and management of artifacts that are related to the development of services. Moreover, we describe how you can federate WebSphere Service Registry and Repository with the WebSphere Business Services Fabric (Fabric) product.

This chapter includes detailed descriptions of the individual steps that developers follow when using both products, based on the ITSOBank scenario that we use throughout this book.

## 15.1 Product overview

The WebSphere Service Registry and Repository product is a relatively recent addition to the IBM SOA Foundation product portfolio. Nevertheless, it plays a crucial role in any service-oriented solution and comes with a rich set of features, some of which we describe in detail in this chapter.

**Note:** In this chapter, we focus on those aspects that are important and relevant in the context of the Fabric and only provide a high-level overview of other functionality. For in-depth material on WebSphere Service Registry and Repository, refer to the product infocenter and the Redbooks publication *WebSphere Service Registry and Repository Handbook*, SG24-7386.

### 15.1.1 A registry and repository for SOA governance

Many existing resources talk about the difference between a *registry* and a *repository*. Generally, a repository is a place where physical artifacts (files) are stored and retrieved, which typically includes the use of a (relational) database that handles the actual storage of data. When used in conjunction with services as part of a service-oriented solution, these artifacts include, for example, WSDL and XML schema files.

A registry, on the other hand, contains metadata. Again, put in the context of SOA, this is information about services, for example, the relationship between services and schemas or data that associates a service with a particular line-of-business, just to name two examples. The most prominent, but not widely used, standard for such a registry is Universal Discovery Description and Integration (UDDI).

For the creation and management of a total solution, you always need both parts, for example, a registry *and* a repository, and you need to have them tightly integrated because there is a close relationship between the information stored in either place. Therefore, the WebSphere Service Registry and Repository product, as its name suggests, offers both parts in one. In summary, WebSphere Service Registry and Repository offers a place to store physical service artifacts and the metadata about those artifacts, and it also keeps them in sync!

This combination allows WebSphere Service Registry and Repository to support another critical element of any transformation towards service orientation: governance, or, more specifically, SOA governance. An *SOA governance model* describes roles and responsibilities, procedures, and work products across the entire service life cycle. Expected benefits from an SOA can only truly be gained if a complete governance model is in place *before* development of services even

begins. For more information about SOA governance and how to implement it, see the Redbooks publication *Implementing Technology to Support SOA Governance and Management*, SG24-7538.

WebSphere Service Registry and Repository offers many of the functions that you need to implement an SOA governance model. However, simply installing and using the product does not imply that you have a governance model in place to start with. Again, it is critically important that you define things, such as roles and responsibilities, process milestones, quality criteria, and so forth, up front.

### **15.1.2 Stages of service-oriented solution development**

WebSphere Service Registry and Repository contains functionality that separately supports each phase of service-oriented development, for example, the modeling and assembly phase, the deployment phase, and finally the management phases. Moreover, governance applies to the entire life cycle, and WebSphere Service Registry and Repository offers support for that, too, as shown in Figure 15-1 on page 406.

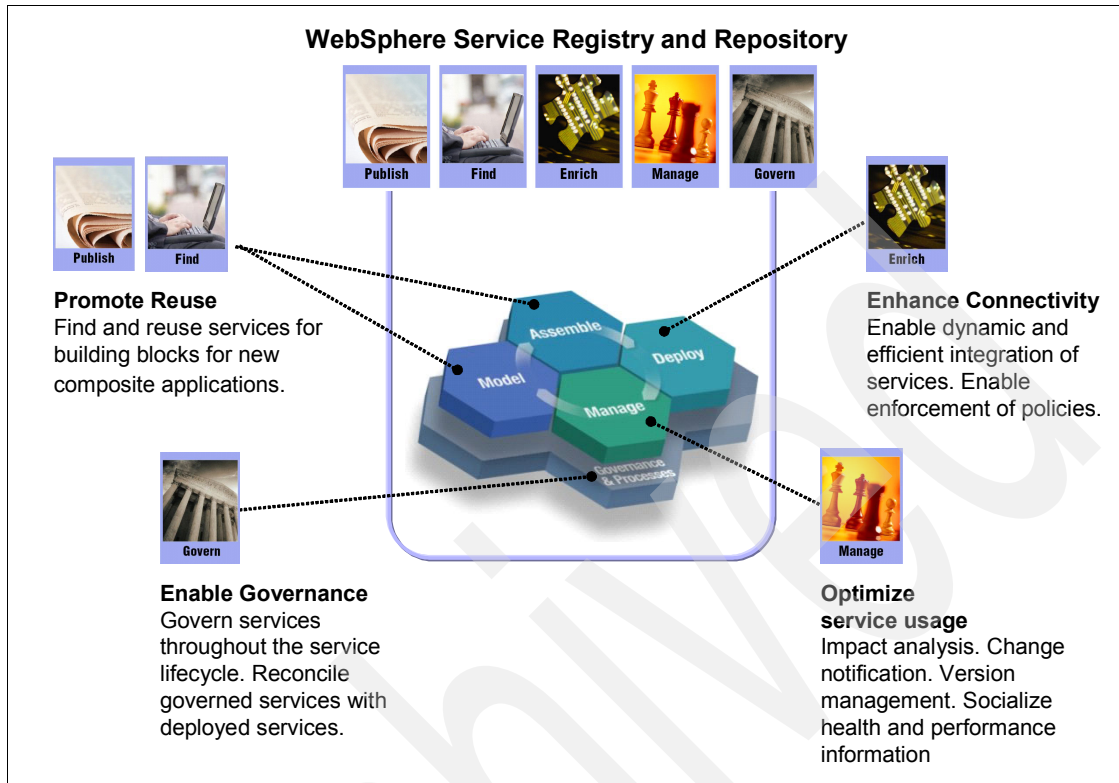


Figure 15-1 WebSphere Service Registry and Repository support for service life cycle

The phases are:

- **Model and assemble**

In the model and assembly phases, services are designed and developed. Many times existing services are composed or aggregated into new services, and by default, each new service is made available to others for reuse; therefore, WebSphere Service Registry and Repository supports publishing information about a service (for example, the WSDL and related XML schema files), and finding existing service using a query interface.

- **Deploy**

In any service-oriented solution, in order to decouple service consumers and service providers, an Enterprise Service Bus (ESB) can be inserted. One of the functions often performed in an ESB is to route requests for a particular service to the correct endpoint. The address of this endpoint can be stored in WebSphere Service Registry and Repository. All of the IBM ESB products

support tight integration with WebSphere Service Registry and Repository in the form of special nodes that execute endpoint queries for a given service.

However, as you learned in this book, the Fabric supports an advanced form of request routing that depends on context, content, and contract. In a Fabric scenario, endpoints are generally managed in the Business Services Repository and not in WebSphere Service Registry and Repository. Therefore, in this book we assume that the Fabric Dynamic Assembler handles request routing, so we do not cover the WebSphere Service Registry and Repository integration with an ESB in any more detail.

- Manage

Management of deployed services includes controlled handling of changes to artifacts, including validation of any change before it gets committed to persistent storage or notification of subscribers when certain events occur. WebSphere Service Registry and Repository also supports the analysis of the potential impact that a change might have.

You can also retrieve real time performance characteristics for services from IBM Tivoli Composite Application Manager for SOA. However, the Fabric delivers similar functionality in its Performance Manager component.

- Enable Governance

WebSphere Service Registry and Repository lets you define a *life cycle* for your services artifacts. A life cycle consists of states that the artifacts can be in and the transitions that take them from one state to another. The definition of roles and permissions for those roles lets you control which type of user has the authority to do what. You can also apply permissions to a life cycle, which lets you define who can transition artifacts to a new state.

Moreover, WebSphere Service Registry and Repository can handle information that IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA) collects. IBM Tivoli Composite Application Manager for SOA monitors invoked services (for example, services that have a request message sent to them) and can collect a list of these services into a file, which lets you compare the list of services that are registered in WebSphere Service Registry and Repository with a list of services that were invoked at runtime; therefore, supporting the identification of 'rogue' services, for example, services that exist and are used but do not exist in WebSphere Service Registry and Repository.

Finally, WebSphere Service Registry and Repository also comes with a service discovery component that can analyze an existing application server instance for installed services. Supported application servers are WebSphere and Microsoft .Net.

### 15.1.3 Technical product details

**Note:** In this section, we focus on those details of WebSphere Service Registry and Repository that are relevant in the context of the Fabric and our ITSOBank scenario and might therefore not include a complete list of all of the functions that WebSphere Service Registry and Repository supports.

#### **WebSphere Service Registry and Repository as a J2EE application**

WebSphere Service Registry and Repository is built as a J2EE application and is installed on top of an existing instance of WebSphere Application Server Network Deployment, which lets it take advantage of the operational features of the application server, such as load balancing and clustering for high availability. WebSphere Service Registry and Repository utilizes a relational database to store its data, meaning that you can use regular backup and restore functionality of the database for repository content.

#### **The WebSphere Service Registry and Repository information model**

Figure 15-2 on page 409 shows an overview of the types of information that you can store in WebSphere Service Registry and Repository.

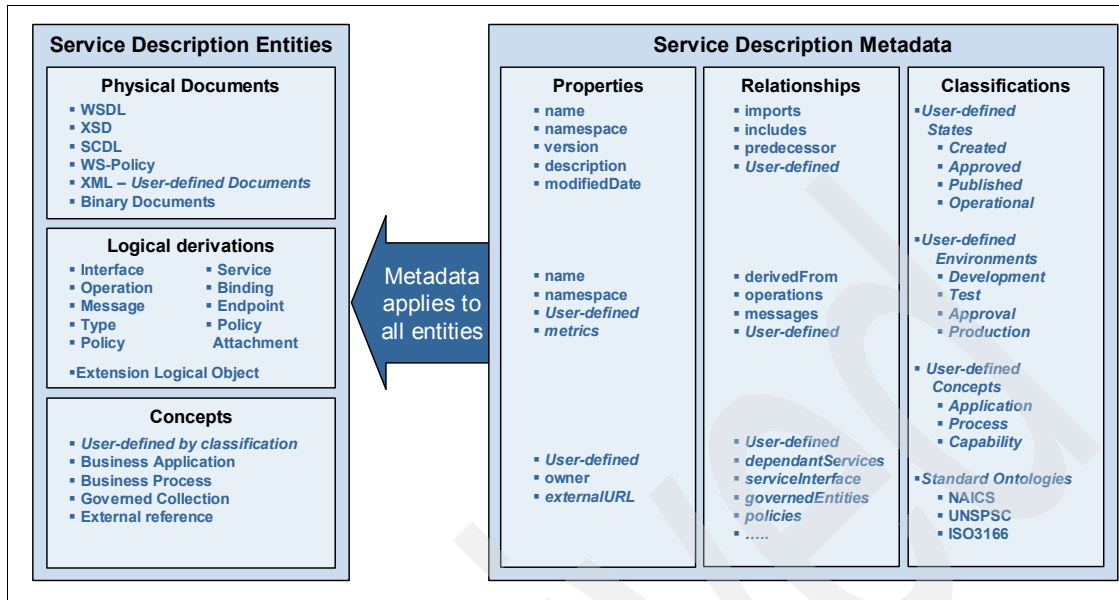


Figure 15-2 WebSphere Service Registry and Repository content

### Physical documents and logical derivations

The main type of physical documents that are stored in WebSphere Service Registry and Repository are files that contain service-related information, such as WSDL, XML Schema, or SCDL files that contain data about an SCA module. Besides loading and storing the actual content, the product parses those well known files into a set of logical objects, for example, a WSDL file is decomposed into its contained elements, such as port type, operations, and message. These derived logical components can then be managed individually.

You can also load other non-XML documents into the registry. However, because WebSphere Service Registry and Repository does not know their type or structure, it stores them as binary documents, and no further parsing or decomposition takes place.

### Logical model

On top of loading physical documents into the registry, WebSphere Service Registry and Repository supports building a logical model that consists of user-defined elements that have no physical representation. This model is defined as an ontology in the *Web Ontology Language* (OWL), which is then imported into the registry. Logical elements are known as *templates*, which are instantiated into *concepts*.

This is similar to the model that Fabric supports out-of-the-box, which has the notion of business service, application suite, and so on. You could theoretically manually build the same model on top of WebSphere Service Registry and Repository. However, because the Fabric comes with a pre-built, ready-to-use logical service model, we recommend that you use that model and do not build your own. We revisit this subject in the next section.

### ***Additional metadata***

You can associate all elements that are stored in the registry, regardless of whether they are based on physical files or logical elements, with additional metadata. One set of this metadata includes properties, such as a name, description, or version field. Another set of metadata defines relationships between artifacts, which effectively ties them together. This relationship between artifacts allows you to define dependencies between artifacts that are honored when content is exported or transitioned to a new state as part of its life cycle transition.

The third type of metadata that you can define for artifacts that are stored in WebSphere Service Registry and Repository is called *classifications*. A classification can be a life cycle state or any user-defined taxonomy that describes an artifact, for example, you might define classifications that describe lines-of-business or technical service layers. You can then use these classifications to structure artifacts into groups that can be queried accordingly, for example, you might want to classify each WSDL file (and thus, the service it represents) according to the line-of-business for which the service was built. Subsequently, if you try to find relevant services for reuse, you can then filter the list of returned services based on a particular line-of-business. You define classifications in OWL, just like the business templates that we mentioned earlier. The WebSphere Service Registry and Repository Web user interface comes with a limited function OWL editor that allows you to define classifications without having to know the details of OWL.

The capability to classify artifacts, both by adding generic classifications and by making them part of a life cycle, is an important concept that we discuss again later in this chapter.

### **Accessing registry content**

There are three ways to access WebSphere Service Registry and Repository to store and retrieve content:

- ▶ **API**

WebSphere Service Registry and Repository supports full access to its content through a set of programmatic interfaces. These interfaces are available through several protocols, namely SOAP, REST, and an EJB Remote interface.



► Web UI

Full control of WebSphere Service Registry and Repository is possible through a Web application that allows you to access it from any browser. User-specific views (*perspectives*) can easily be defined, which gives each user only the subset of functionality that they need and are entitled to.

► Development tool plug-in

You can publish physical files to and import them from WebSphere Service Registry and Repository directly from within one of two supported development environments: Eclipse and Visual Studio® .Net.

Examples of Eclipse-based tools are WebSphere Integration Developer or Rational Application Developer, which enable developers to interact with WebSphere Service Registry and Repository without ever leaving the tool that they use to develop services and their implementations. We show an example of what this looks like in WebSphere Integration Developer in 15.4.2, “Publishing using the Eclipse plug-in” on page 436.

The details of exactly which functionality is available over each channel are beyond the scope of this book, but let it suffice to say that not every function is supported in every channel.

## Dependency resolution

When you load non-binary documents into the registry (for example, WSDL, XSD), WebSphere Service Registry and Repository parses the loaded document to see if it imports any other files, thus creating a dependency between the documents. In case these additional files are not yet stored in the registry, you are challenged to also load those dependent files. In case it does exist, you can choose between the already loaded version of the file or load a new version, which ensures that no file in the registry contains imports that cannot be resolved.

This function is very helpful in resolving dependencies between a potentially large number of files, for example, it is very common for XML Schema files that represent a complex data model to build a large graph of dependencies among each other.

## Querying content

All content in WebSphere Service Registry and Repository is stored in a hierarchical tree. The location within that tree depends on the type of artifact. Queries into that tree, for example, to locate certain artifacts, are expressed by using XPath statements. The root element of the target tree is called /WebSphere Service Registry and Repository. All XPath statements that are used in the Service Registry queries have the same structure:  
`/WSRR/type[meta-data expression]`

The type part of the structure represents the type of artifact for which you are looking, for example, a query using `/WSRR/BaseObject` returns all content that is stored in the registry because `BaseObject` is the parent element for all stored elements. As another example, `/WSRR/WSDLDocument` returns all stored WSDL documents. For details about the types and the tree structure that WebSphere Service Registry and Repository uses, refer to the online infocenter.

You can use the metadata expression part to further constrain the list of returned elements by applying classification and other information, for example, the following query returns a subset of all WSDL documents, depending on their name:

```
/WSRR/WSDLDocument[starts-with(@name, "Claims")]
```

Example 15-1 selects all XSD documents that are classified as a utility service (assuming, of course, that such a classification exists, since it does not come with the product).

*Example 15-1 XSD documents*

---

```
/WSRR/XSDDocument[classifiedByAnyOf(.,  
'http://my_enterprise.com/class#Utility')
```

---

Both the WebSphere Service Registry and Repository Web user interface and the Eclipse and Visual Studio .Net plug-ins make life a little easier because it lets you define queries interactively through a set of dialogs, so that you do not have to articulate the actual query as described above. The Web UI even lets you store queries that you defined, which allows them to be reused over and over without the need to step through the query wizard dialogs again.

## Service life cycle

As previously mentioned, you can associate all of the artifacts in WebSphere Service Registry and Repository with a life cycle. A life cycle consists of a number of states that the artifact can be in and the transitions that take it from one state to another.

A life cycle can be visualized as a state machine, as shown in Figure 15-3.

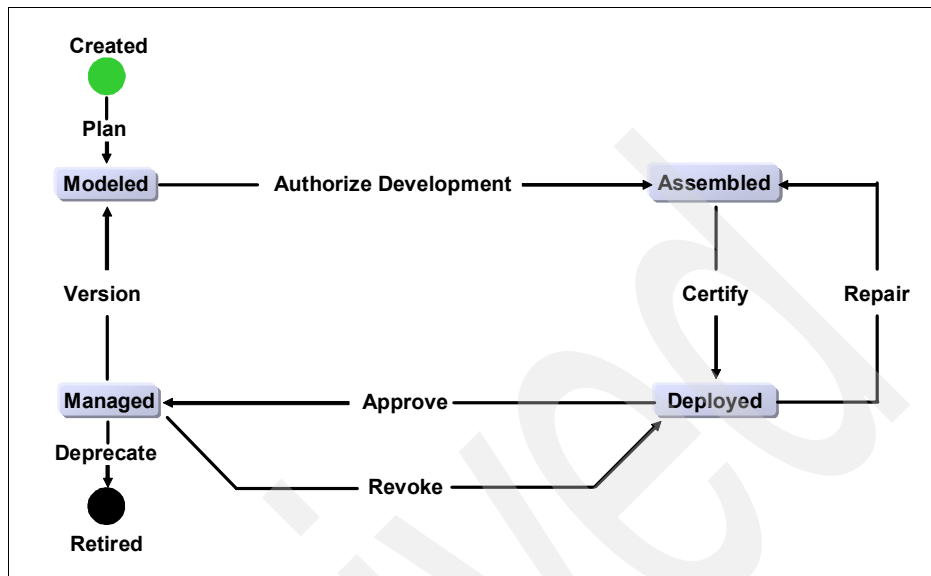


Figure 15-3 The WebSphere Service Registry and Repository default service life cycle

Service life cycles for WebSphere Service Registry and Repository are developed using the State Machine editor in WebSphere Integration Developer. You then export the life cycle to the file system as an SACL file, which is finally imported into WebSphere Service Registry and Repository. You can then use WebSphere Service Registry and Repository to apply this life cycle to any existing artifact. As we will see later, you can define access permissions that restrict access to artifacts based on the state that they are in. This is useful, for example, to ensure that only files that have reached a certain state (for example, “Deployed”) are retrieved and reused by other developers. You can also define who is authorized to transition artifacts between states.

## Security

Given that WebSphere Service Registry and Repository runs on top of WebSphere Application Server, it can take advantage of the J2EE-based security model of the application server. This model lets you restrict access to the registry and its Web UI using regular access control mechanisms.

On top of this coarse-grained access control, you can define very fine-grained permissions that manage who is entitled to which actions on which artifacts in the registry, which you accomplish by defining *permissions*. A permission consists of a target (which is identified by an XPath statement, similar to how you query content) and an action that can be performed on this target. Available actions are

create, delete, retrieve, update, make governable (that is, add the artifact to a life cycle), and transition (that is to move it from one state in the life cycle to another).

Next, a permission is associated with a user role. A role can be associated with more than one permission. The combination of all associated permissions defines the level of access that a certain user has.

Security definitions can be made through the Web UI or in a file using the eXtensible Access Control Markup Language (XACML), which is an OASIS standard for describing access control. You can import XACML files into WebSphere Service Registry and Repository.

## Validation and notification

As an additional form of governance, WebSphere Service Registry and Repository lets you inject user-defined behavior when certain events happen, for example, when a new artifact gets created or when an existing artifact is changed or deleted. Another type of event that can be used is the transitioning of an artifact from one state to another (or when the artifact is added to a life cycle to begin with). This user-defined logic can be injected at two times, namely right before the underlying data store is actually updated and right after the underlying data store is actually updated.

The first type of user-defined logic is called *validation* because it lets you validate updates to the registry before they become permanent. WebSphere Service Registry and Repository has a *validation plug-in* that lets you define for which events the functionality is invoked. This functionality is implemented in the form of validators, which are Java classes that apply whatever logic is needed, for example, you can develop a validator that ensures that a <documentation> element exists in any WSDL file that is promoted from “proposed” to “under development” state. In case a validator returns an error, the intended change is not made permanent and is rolled back instead.

The second type of logic is inserted post commit, in other words, after a change is stored permanently. It is called *notification*, and as the name suggests, you can use it to notify users of things that occurred. Just like in the case of validation, you can configure the notification plug-in to call user defined logic - called notifiers. Additionally, you can use this plug-in to generate emails that are sent to subscribers for the particular type of event that happened, for example, the ‘service registrar’ might want to receive an e-mail whenever a new WSDL file is created in the registry so that he or she can go and review it or add it to the appropriate life cycle.

On top of validation and notification, there is also a modification plug-in. The custom code you can insert here also runs before the underlying transaction is committed, just like in the case of the validator. The idea is that a *modifier* can

make required updates to the affected artifact before a change is made permanent.

### Configuration profiles

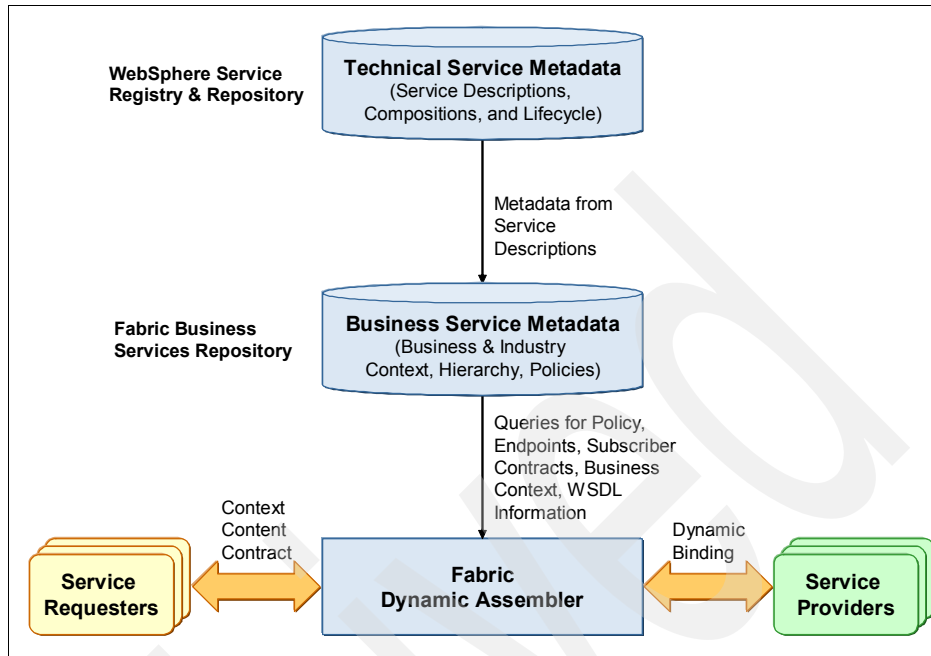
All of the definitions that you make for a certain instance of WebSphere Service Registry and Repository, that is classifications, life cycles, security definitions, and so on, you can export into one compressed file. This compressed file then effectively contains an entire configuration for the registry. This file contains what is called the 'configuration profile'. You can then import this profile into another instance of WebSphere Service Registry and Repository. Each registry instance can contain multiple configuration profiles, but only one of them can be active at any given time.

This capability allows you to back up a configuration, promote it to other instances, and runtime switch between configurations.

## 15.2 WebSphere Service Registry and Repository and the Fabric: three types of scenarios

This book shows you how you can build composite applications using Fabric. Part of that is to register new and existing technical services in the Business Services Repository. This repository contains *business metadata* about your services and your composite application: business policies, subscription policies, industry metadata, and so on. At the same time, we described in this chapter that WebSphere Service Registry and Repository offers a repository for *technical service metadata* plus support for classification and governance of this metadata.

Fabric supports federation of technical service metadata from WebSphere Service Registry and Repository into the Business Services Repository where it can be leveraged for building composite applications (see Figure 15-4 on page 416). This ability to federate data between WebSphere Service Registry and Repository and Fabric creates a level of integration that positions WebSphere Service Registry and Repository to be used in concert with Fabric, each playing a specific role in the overall environment. The Business Services Repository (BSR) handles business relevant information that Dynamic Assembler uses to make dynamic runtime decisions about which services to invoke, based on context, content, and contract. WebSphere Service Registry and Repository delivers the technical information that complements and completes the data that is stored in the BSR.



*Figure 15-4 Data federation between Fabric and WebSphere Service Registry and Repository*

In this section, we introduce three types of use case scenarios that apply when you use Fabric and WebSphere Service Registry and Repository together. The details of this are simply driven by the sequence in which the products are rolled out:

- ▶ In the first scenario, we assume that you start out working with Fabric on building composite applications and later add WebSphere Service Registry and Repository to manage technical metadata.
- ▶ In the second scenario, we assume that you are already using WebSphere Service Registry and Repository as your registry and repository for service-related data and then evolve towards building composite applications with Fabric.
- ▶ In the third scenario, we cover when both products are introduced at the same time.

Later sections in this chapter contain detailed step-by-step descriptions of how to flow relevant artifacts through the various tools.

### 15.2.1 Scenario 1: Deploy Fabric first and WebSphere Service Registry and Repository second

The Business Services Repository feature of Fabric comes with a complete, predefined meta-model for services, which not only includes the notion of application suites, applications, business services, and so on but also includes policies and assertions that are applied to individual services. Among other things, the Dynamic Assembler component uses this additional metadata at runtime to make routing decisions.

WebSphere Service Registry and Repository also offers the ability to create a meta-model for services. Both products leverage ontologies described in the OWL language, but in their current releases, they do not offer a way to share these models. Therefore, assuming you use the Fabric's meta-model to describe your services, business services and composite applications, there is no need to do the same in WebSphere Service Registry and Repository.

At the same time, WebSphere Service Registry and Repository act as the back end repository for purely technical artifacts, such as WSDL and XML Schema, which cannot be stored in the Business Services Repository. These artifacts are governed in WebSphere Service Registry and Repository and then federated into Fabric for further service composition. In fact, Fabric does not handle XML Schema at all. Its service model breaks down to the level of interfaces (port types) and operations but does not cover the messages that are part of an operation. The message structure, described in XML Schema, can only be handled in WebSphere Service Registry and Repository. WSDL documents can be imported into Fabric, which does not physically store the actual document; instead, it creates the interface and endpoint definitions from them.

This approach of integrating both products leads to the guidelines in the next section for configuring WebSphere Service Registry and Repository.

#### **Do not create business templates**

Fabric comes with a predefined logical model, which is comparable with a logical model you would build using business templates in WebSphere Service Registry and Repository. You can also extend the model in Fabric as needed for specific policies for your services. Therefore, building the same or additional logical constructs in WebSphere Service Registry and Repository is redundant and needlessly distributes your overall service metadata across multiple products.

However, you might want to use WebSphere Service Registry and Repository as a repository for additional documents that are part of your service design. This repository could include UML model files that you create in a tool, such as Rational Software Architect or a Word document that describes non-functional

requirements for a service and other non-technical information. Link these documents to the technical artifacts (these are WSDL and XML Schema files) for a given service by creating a light weight logical model that defines the necessary links and relationships between these artifacts.

### **Modelling services in Composition Studio**

Instead of creating business templates in WebSphere Service Registry and Repository, you can ensure that services and the relationships between them are always completely represented in Fabric. Your development process should include the necessary steps to make this happen as soon as new services are developed.

The federation capability between the two products, which we review in detail in 15.3, “Federation between WebSphere Service Registry and Repository and Fabric” on page 424, helps with this because WSDL documents in WebSphere Service Registry and Repository are automatically imported into Fabric as interfaces.

### **Classifying artifacts stored in WebSphere Service Registry and Repository**

WebSphere Service Registry and Repository supports adding user-defined classifications to any artifact that is stored in its repository, which helps to logically and semantically group artifacts. The grouping is also useful when searching the repository for content. Examples for this are classifications around lines-of-business or service layer, for example utility versus process-centric versus data-centric services.

As part of establishing your governance model, develop a common taxonomy for services, possibly based on existing industry standards. This taxonomy (which is basically a common way of naming and structuring your services) is then deployed into WebSphere Service Registry and Repository as a classification. Each new WSDL document or XML Schema that is published to the registry should subsequently be classified appropriately.

The federation between WebSphere Service Registry and Repository and the Fabric does not support taking advantage of classifications. In other words, you cannot select a subset of service artifacts for federation based on how they are classified. But the classification is still useful when you look for services for reuse, composition, and aggregation—it simply requires the developer to use the WebSphere Service Registry and Repository Web UI to find the right services and then model their relationships in Fabric accordingly.



## Defining a service life cycle

The technical artifacts that describe a service always go through a life cycle that takes them from initial proposal and development to deployment and eventually retirement. The exact details of the life cycle, that is which states and transitions are needed, depends on your particular environment, your development process, and the roles that exist in your organization. WebSphere Service Registry and Repository comes with a simple default life cycle out-of-the-box and also offers a set of more complex life cycle as part of the *governance profile* that comes with the product. You can either use these unchanged or use them as a starting point for your own custom life cycle.

After you define and configure your life cycle in WebSphere Service Registry and Repository, associate each WSDL and XML Schema file that is published to the registry with this life cycle. In WebSphere Service Registry and Repository terms, this means making these artifacts *governable*. You can define who has the authority to add artifacts to a life cycle and then transition them through its individual states.

Associating your technical service definitions with a life cycle also allows you to enforce rules about what these definitions need to look like and their classifications, for example, assuming that only the *service architect* role has the authority to transition a WSDL from under development to approved state, this architect can verify that it is classified and that it conforms to any other inhouse convention and standard that might exist. And later on, you will see how we use the life cycle to only federate those artifacts into Fabric that went through the appropriate review process.

## Using the version support in WebSphere Service Registry and Repository with caution

By default, all artifacts that are stored in WebSphere Service Registry and Repository have a version property. This property is defined as a string field and can contain any value. You can store files with the same name, but with different versions, to account for changes that are made to, say, a service. Consequently, you can query for artifacts with a specific version number. You can also leverage the version property in any validators or notifiers that you might configure, for example, you can enforce a certain versioning scheme by checking in a validator that the version property always follows a particular structure (a common scheme is to use major.minor version numbers, for example, 1.1, 1.2, and so on).

WSDL files or XML schemas with the same name but different versions can be completely different from each other. There is no automatic checking for backward compatibility or any other type of logic that is executed automatically when a version of an existing document is published.

The versioning support in Fabric is different. You can have multiple versions of a service, but they all still have to support the same technical interface. The Dynamic Assembler allows you to pick the appropriate instance of that service based on the context and content of the incoming request message.

Moreover, the current federation between WebSphere Service Registry and Repository and the Business Services Repository will only copy the first returned version of a WSDL-defined service. Additional versions of that service are ignored, which means that you have to be very careful when you use the versioning support in WebSphere Service Registry and Repository because it might not properly reflect in the Fabric. Given the fact that the notion of versions in Fabric is different, you can only manually synchronize between the two if you make sure that you only create backward compatible versions of a service in WebSphere Service Registry and Repository. Non backward compatible changes must go into a new service altogether, including a new WSDL document.

### **Sharing user registries and synchronizing role definitions**

Fabric runtime sits on top of WebSphere Process Server, which, in turn, runs on top of WebSphere Application Server Network Deployment. WebSphere Service Registry and Repository also runs on top of WebSphere Application Server Network Deployment. Both products take advantage of the underlying security infrastructure that the application server offers, including the ability to share user registries.

Both products support the ability to define specific roles and map them to existing users and groups. Fabric uses *organizations* and *users* to control subscriptions to services and access to the Business Services Repository. WebSphere Service Registry and Repository uses *roles* to define fine-grained access control to the registry and repository.

When you use both products, always make sure that both sets of definitions are in sync, which is a manual process. In most cases, the role definitions are not identical because Fabric requires more business relevant roles, whereas WebSphere Service Registry and Repository requires more technically-oriented roles. But there are many roles that are shared between the two, for example, roles, such as Developer, Architect, or User.

## **15.2.2 Scenario 2: Deploying WebSphere Service Registry and Repository first and Fabric second**

As part of their transformation towards SOA, IT organizations often start out with building an appropriate level of infrastructure to host and connect services, which

might include an Enterprise Service Bus or the deployment of application servers to host newly developed services, just to name two examples.

Regardless of what the exact approach is, establishing a governance model and an environment with tools and products that helps to implement this model is always a critical success factor. WebSphere Service Registry and Repository directly addresses this need and is typically used even by companies that build an initial set of services without the immediate intent of building composite applications to implement business processes.

Taking full advantage of WebSphere Service Registry and Repository includes building a logical model that describes the overall service model on top of any technical artifacts that are stored in the repository. WebSphere Service Registry and Repository comes with a predefined sample model as part of its governance profile that can be used as a starting point. Of course, all of the other features of WebSphere Service Registry and Repository that we outlined earlier apply to this situation, too: defining life cycles, access control, notification, validation, and so on.

After an organization is ready to start building composite applications and wants to take advantage of Fabric for the initiative, the question arises of how to properly integrate both products.

### **Logical model**

An existing logical model for things, such as (technical and business) services, applications, processes, and so forth, which were defined in WebSphere Service Registry and Repository, cannot automatically be migrated to the existing service meta-model that Fabric supports; therefore, you must manually translate any existing definitions into the Fabric's Business Services Repository to the degree possible. Typically, we do not recommend that you preserve the WebSphere Service Registry and Repository logical model unless it describes things that are not covered in Fabric. We previously mentioned the example of service design documentation that is associated with technical artifacts, such as WSDL and XML schema.

### **Versioning**

In the previous section, we explained how the approach towards versioning is different between WebSphere Service Registry and Repository and Fabric. If you have an existing environment that includes services with the same name, but different versions, in WebSphere Service Registry and Repository, you cannot directly apply these versions to the Business Services Repository for service composition. If the newer versions of a service are backward compatible with earlier instances, you can manually enter them into the Fabric repository. If they are not backward compatible, you must manually convert them to new services

altogether, possibly indicating their relationship by choosing proper naming conventions.

## Routing

This scenario often applies in cases where an Enterprise Service Bus is deployed before Fabric is considered for building composite applications. WebSphere Service Registry and Repository offers integration with all three of the IBM Enterprise Service Bus products (WebSphere Message Broker, WebSphere Enterprise Service Bus, and WebSphere DataPower®). This integration is normally used to look up endpoint addresses for service providers from within the ESB to further decouple service consumers and providers by routing requests through the ESB.

The Dynamic Assembler component of Fabric provides the ability to help choose the appropriate endpoint for a given service request, based on the context and content of the request message. It uses the policies that are defined in the Business Services Repository to select this endpoint. Using both mechanisms for routing, namely the ESB-to-WebSphere Service Registry and Repository integration and the Dynamic Assembler, leads to the requirement to manage endpoints in both. Consider migrating endpoint selection from your ESB into Fabric if possible. However, keeping both in place might have merits, depending on your particular requirements. You just need to ensure that endpoint management is synchronized properly between registries.

## Policies

Fabric allows the use of policies to dynamically route requests to the right service provider instance. These policies are stored in the Business Services Repository and are checked against existing endpoint assertions for each message that flows through the Dynamic Assembler.

WebSphere Service Registry and Repository supports the storage and retrieval of policies based on the WS-Policy standard. The overall goal of this standard is to increase the consumability and completeness of service definitions by allowing the addition of behavioral information in a declarative manner. Several advanced standards and specifications take advantage of the WS-Policy standard, by applying it to a particular aspect, for example, the WS-SecurityPolicy standard describes how to define policies for secure interactions between service consumers and providers.

Despite the fact that both products use the same term, they apply policies at different levels. Fabric uses policies to define endpoint selection decisions mostly based on business level information, for example, a Fabric policy can describe that preferred customers are routed to a different service instance than

non-preferred customers, and there are many more examples throughout this book.

WebSphere Service Registry and Repository, on the other hand, supports the handling of technical policies that the underlying infrastructure can use to handle service interactions properly. One example is the definition of security policies that define how messages are encrypted or how to exchange and promote credentials. These policy definitions can be used at runtime, for example, by retrieving them from within an ESB that then acts as an enforcement point for these policies, or simply to document and manage them in a standardized form and in the same place where the associated service definitions are handled.

Thus, you will most likely end up with policy definitions in both places, namely in the Business Services Repository and WebSphere Service Registry and Repository. Based on the description above, both types of policies are serving different roles and, consequently, are owned by different roles of users. Business policies contained in Fabric are typically handled by users who understand business requirements, whereas the technical policies in WebSphere Service Registry and Repository are managed by infrastructure experts.

### **Everything else is smooth sailing**

Beyond the considerations listed in this section, everything we mentioned in the previous section still applies: you will always want to leverage classifications and life cycles and synchronize role definitions between the two products.

## **15.2.3 Scenario 3: Deploying WebSphere Service Registry and Repository and Fabric simultaneously**

In this scenario, we assume that both Fabric and WebSphere Service Registry and Repository are rolled out at the same time. Because either Fabric or WebSphere Service Registry and Repository often exist before the other within a project, this is less visible in practical examples. But if an opportunity is available, it is a preferred option to start with both products. The reason for this is simply that it avoids getting too deeply entrenched in the details of one product without being able to take full advantage of the other, for example, it avoids having to manually migrate content based on a logical service model that is defined in WebSphere Service Registry and Repository to the service meta-model that Fabric promotes.

It can also be anticipated that in the future, the integration between both products is further improved, making some of the manual steps that we describe in this chapter obsolete. While the details of that are unknown, of course, leveraging both products in parallel makes the transition to future enhancements easier.

## 15.3 Federation between WebSphere Service Registry and Repository and Fabric

Now that we described the various functions of WebSphere Service Registry and Repository and how to integrate them with Fabric, we look at how this is actually done and take you through the necessary configuration steps.

Figure 15-5 shows the activities that take place when you use Fabric and WebSphere Service Registry and Repository together and the respective roles that execute each step.

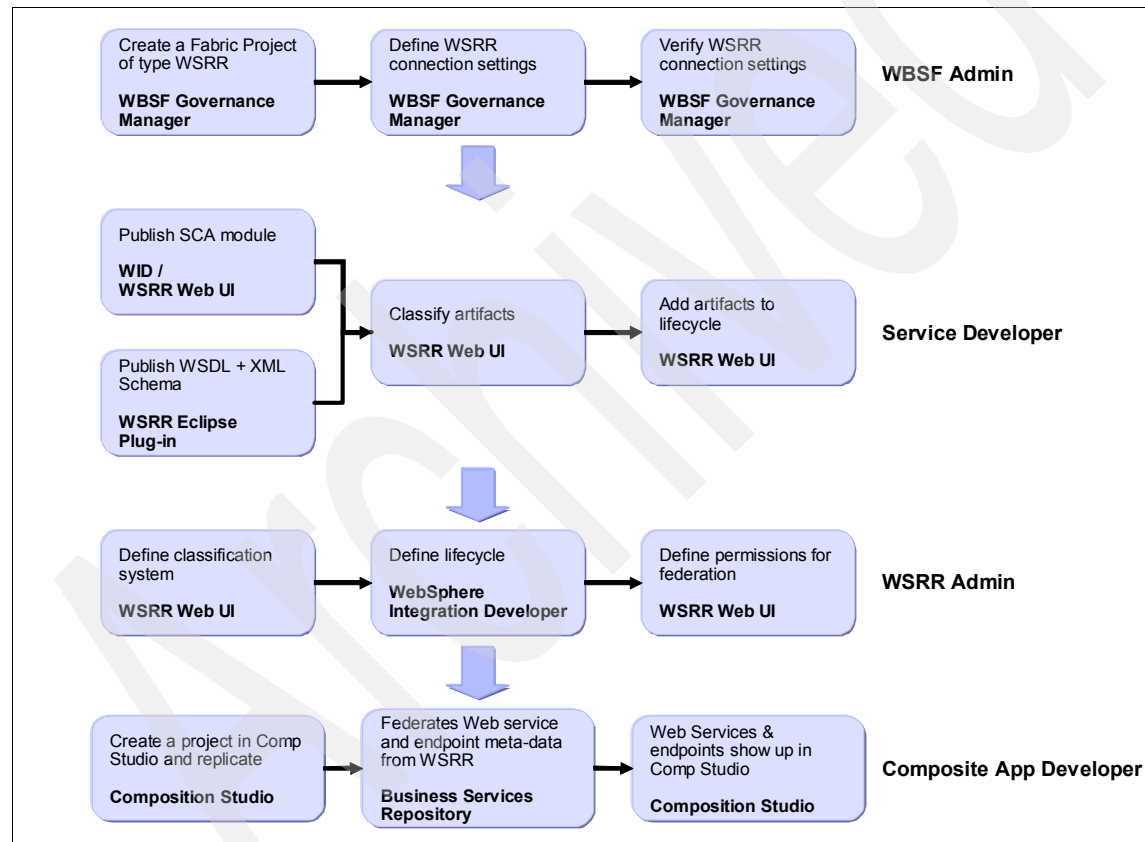


Figure 15-5 Usage scenarios

In the remainder of this chapter, we take you through a concrete example for each of the steps in Figure 15-5.

### 15.3.1 Configuring a federation project

At the core of the integration between both products is the ability to federate content from WebSphere Service Registry and Repository into the Fabric Business Services Repository. To enable this federation of data between the BSR and WebSphere Service Registry and Repository, you define a specific type of project in Fabric that automatically connects to an existing WebSphere Service Registry and Repository instance and copies the relevant content into its own registry, where you can use it for service composition and so forth.

The required configuration, at least for now, happens solely through the Fabric admin console; therefore, for the remainder of this section, we assume that you have an instance of WebSphere Service Registry and Repository installed and running, either locally or on a machine to which you have network connectivity.

To configure a federation project:

1. Open the Fabric console application in your browser, and log in.
2. Navigate to **My Services** → **Governance Manager** → **Configure projects**.
3. Click **Create a Project**.
4. In the following dialog, in the Project Name field, enter **WSRR Federation**. Optionally add a description.
5. In the Project Type field, click **External WSRR**, which causes a couple of additional entry fields to appear.
6. In the External Details section, in the Source Name field, enter **Corporate\_IT\_Registry**.
7. In the Sponsored NS URI field, enter  
`http://www.itsobank.com/wsrr/sponsored/corp#`  
This namespace URI can contain pretty much any value as long as it is a valid URI. The # at the end is mandatory.
8. In the Project Team section, for the Team Organization, click **System**. Your window should look like Figure 15-6 on page 426.

Create a Project

Fields marked with an asterisk (\*) are required fields.

Project Details

\* Project Name

WSRR Federation

Description

This project handles the federation of data with WSRR.

Project Specification

\* Project Type

External WSRR

External Details

\* Source Name

Corporate\_IT\_Registry

\* Sponsored NS URI

http://www.itsobank.com/wsrr/sp

Project Team

\* Team Organization

System

Create Project

Cancel

Figure 15-6 Creating a WebSphere Service Registry and Repository federation project

9. Click **Create Project**, which creates the new project.
10. In the resulting dialog, click **Federation Settings**. Even though the project exists already, we have not yet defined how it can access the WebSphere Service Registry and Repository instance for actual federation of data.
11. Under Hostname, enter the name or IP address of the machine where your instance of WebSphere Service Registry and Repository is running. We assume here that it runs locally, so enter localhost.
12. For the Port, enter the appropriate number for your installation. Because we assume that your registry runs in secure mode, this is the default port number for secure access. In our example, the port number is 9448.



**Tip:** If you are not sure which port number your WebSphere Service Registry and Repository instance is using, you can use the application server console application to obtain it:

1. Open the admin console for the application server instance on which you installed WebSphere Service Registry and Repository.
2. Click **Servers** → **Application servers**.
3. Click your application server (usually called server1). In the resulting window, click **Communications** → **Ports**, which opens a table with all of the port numbers that your server uses. The number listed next to **WC\_defaulthost\_secure** is the number you are looking for.
4. Under Security Settings, set Requires Security to True.
5. We are using a specific user ID to access WebSphere Service Registry and Repository (you will see later why we are doing this). In our example, the user ID that we use is WSRRFabricAdmin. This must be a valid user ID to access your WebSphere Service Registry and Repository instance. Enter this user ID into the Username field, and then enter the appropriate password into the Password field.
6. The Expiration Settings define how often content is retrieved from WebSphere Service Registry and Repository. We will use the Time To Live parameter, and set it to 30 min. This means that the Fabric will start a thread every 30 minutes that retrieves new content from WebSphere Service Registry and Repository. Note that setting this to Do Not Cache indicates that no content is to be retrieved at all, effectively turning the federation off.
7. Make sure you save your changes by clicking the **Save** button. Figure 15-7 on page 428 shows what your window now looks like.

Configure Projects

Project

WSRR Federation

Back to Projects

Project Details

Project Team

Namespaces

Federation Settings

Fields marked with an asterisk (\*) are required fields.

Connection Details

\* Hostname

localhost

\* Port

9448

Security Settings

\* Requires Security

True

\* Username

WSRRFabricAdmin

\* Password

\*\*\*\*\*

Expiration Settings

\* Data Expiration

Time to Live

\* Time to Live

5 min

External Settings

Source Name

Corporate\_IT\_Registry

Sponsored Namespace

WSRR Federation Sponsored

Save

Test Connection

Figure 15-7 Setting federation parameters for the WebSphere Service Registry and Repository federation project

- To validate that you can now access your registry, click the **Test Connection** button, which should result as shown in Figure 15-8.

Project Details

Project Team

Namespaces

Federation Settings

Fields marked with an asterisk (\*) are required fields.

Info

- The connection test was successful.

Connection Details

\* Hostname

localhost

Figure 15-8 Testing the connection with WebSphere Service Registry and Repository

### 15.3.2 Verifying content federation

Now that we configured the connection between Fabric and WebSphere Service Registry and Repository, we can verify that data is indeed read from within the registry into the Business Services Repository.

The easiest way to verify that data is read from within the registry to the BSR is to take a simple example WSDL file, load it into WebSphere Service Registry and Repository, and then verify that its interface is also available in Fabric. For this simple test, you can either load a sample WSDL file from the Internet or create a sample file using the WSDL editor in WebSphere Integration Developer. For sample WSDL files, go to:

<http://xmethods.net>

To verify content federation by loading a sample WSDL file directly from the Internet:

1. Open the WebSphere Service Registry and Repository admin console (depending on the host name and port number you use, the URL looks like this: <http://localhost:9085/ServiceRegistry>). Assuming that you run your registry in secure mode, it automatically redirects you to a secure connection and asks you to log in using your user ID and password.
2. Make sure that you are in the Administrator perspective.
3. Click **Service Documents** → **Load Documents**.
4. In the Load Documents panel, click **Remote file location**, and enter the following URL:  
<http://webservices.daelab.net/temperatureconversions/TemperatureConversions.wsdl>

This is a simple temperature converter service that we are using for our test. Optionally, enter a description and version number for the service. Make sure that the Document Type field is set to WSDL. Figure 15-9 on page 430 shows what this looks like.

**Path to the Document**

☐ Local file system

Specify path

☒ Remote file location

Specify URL

Document type  
WSDL

Enter document description:

Enter document version:

Figure 15-9 Loading an external WSDL document into WebSphere Service Registry and Repository

5. Click **OK**. In the window that is displayed, click **Finish**.
6. After the WSDL documents are loaded into WebSphere Service Registry and Repository, Figure 15-10 is displayed.

Load Documents			
<b>Documents Loaded Successfully</b>			
The following documents have been loaded into the repository:			
Name	Description	Namespace	Version
ITempConverter	Temperature converter service	http://www.borland.com/soapServices/	1.0

Figure 15-10 The WSDL document is loaded into WebSphere Service Registry and Repository

The next part of our test is to verify that this service is available through the Business Services Repository. We check that by searching the repository from within the Fabric admin console. You might have to wait for a few minutes to ensure that the synchronization between WebSphere Service Registry and Repository and Fabric has actually taken place.

To verify that the service is available through the Business Services Repository:

1. Navigate to **My Services** → **Business Services Repository** → **Search Repository**.
2. Navigate to **Search** → **Search For**, clear the All option. Select the **Interfaces** check box.

3. Click **Search**. The search result looks similar to Figure 15-11.

**Search**

Resources Subscribers Policies

**Search**

Search For: ☐ All ☐ Application Suites ☐ Applications ☐ Business Services ☒ Interfaces ☐ Composite Services ☐ Endpoints

Name:

Search

**Resource Selection**

View 10 rows at a time

2 rows

Name ↓	Type	Publisher
<a href="#">Event Consumer</a>	WSDL Interface	System
<a href="#">TemperatureConversionsSoapType</a>	WSDL Interface	WsrrProvider

Page 1 of 1

Figure 15-11 Searching for federated content in the Business Services Repository

Depending on your setup, you may have more interfaces defined. The interface called `TemperatureConversionsSoapType` is the one that we retrieved through WebSphere Service Registry and Repository.

We do not need this imported service for any other testing, so you can delete it in WebSphere Service Registry and Repository; however, deleting it in WebSphere Service Registry and Repository does not remove it from the Business Services Repository. To remove it in the BSR, you must create an associated Fabric project in Composition Studio and manually delete the interface there.

4. Click the `TemperatureConversionsSoapType` interface. In the resulting Details view, in Figure 15-12 on page 432, you can see that the owning namespace of this interface is called `WSRR Federation Sponsored NS 1`. This namespace owns all interfaces that are retrieved from WebSphere Service Registry and Repository. As a consequence, you must import the namespace into every Fabric project that intends to take advantage of services that are registered in WebSphere Service Registry and Repository.

Details

General Information

Name	TemperatureConversionsSoapType	Type	WSDL Interface
Project	WSRR Federation	Created	Feb 12, 2008 1:20:19 PM
Namespace	WSRR Federation Sponsored NS 1	Modified	Feb 12, 2008 1:20:19 PM
Description			
author	WsrProvider		
interface document	urn:wsrr:12b6c712-aa72-4291.8b32.126b461232a6		
interface name	http://webservices.daelab.net/temperature#TemperatureConversionsSoapType		
publisher	WsrProvider		
visibility	Intranet		
WSDL Version URI	http://schemas.xmlsoap.org/wsdl/		

service Operation

Name	Description
FahrenheitToCelcius	
WindChillInCelcius	
CelciusToFahrenheit	
WindChillInFahrenheit	

Figure 15-12 Details for an interface federated from WebSphere Service Registry and Repository

For now, we showed that a sample WSDL document (which we pulled off the Internet) was successfully copied into the Business Services Repository. In the next two sections, we describe when and how to publish the appropriate content to WebSphere Service Registry and Repository, how to manage it in WebSphere Service Registry and Repository, and then when and how to make it reusable for service composition in Composition Studio.

## 15.4 Using WebSphere Service Registry and Repository as part of a Fabric project - publishing content

Earlier in this chapter, we mentioned that in our scenario, WebSphere Service Registry and Repository stores and manages technical service artifacts. In concrete terms, these are WSDL and XML schemas as well as SCA integration modules. An SCA integration module can contain a number of WSDL and XML schema files, plus specific files that describe how all of the services that are defined and referenced in the module act together. If you followed the creation of

the ITSOBank example throughout this book, you used these concepts already, which is why we do not explain them in detail here again.

### 15.4.1 Publishing an SCA integration module

Let us assume that you developed part of a solution using the SCA programming model and thus created an SCA module in WebSphere Integration Developer. The first step to allow reusing this new module is to publish it to WebSphere Service Registry and Repository. In the following steps, we assume that we reuse the ITSOBank example, and, specifically the ITSO module.

**Note:** Before starting to work on the example, shut down your Fabric server. The reason is that we do not want to federate any content from WebSphere Service Registry and Repository into the Business Services Repository until we configure some further settings, which we do in 15.5, “Using WebSphere Service Registry and Repository as part of a Fabric project - filtering federated content” on page 442.

To publish an SCA integration module:

1. Open WebSphere Integration Developer, and make sure you are in the Business Integration perspective. One of the modules you can see in the navigator pane on the left is called ITSO. Right-click the ITSO module, and click **Export**.
2. Select **Business Integration** → **Integration module**, and click **Next**.
3. In the resulting dialog, select the ITSO module. Select the **EAR files for server deployment** option. Click **Next**.
4. In the next window, define a directory for the export, for example, c:\temp. Click **Finish**.
5. Open the WebSphere Service Registry and Repository admin console in a browser window. Click **Service Documents** → **Load Documents**.
6. Select **Path to the Document** → **Local file system**, and enter the name and location of the module file that you just exported. In our example, that location and name is c:\temp\ITSO.ear.
7. Select SCA integration module as the Document type, and optionally enter a description and version, as shown in Figure 15-13 on page 434. Click **OK**.

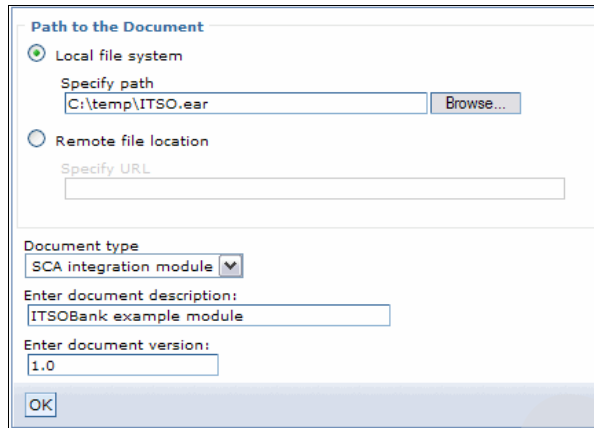


Figure 15-13 Loading an SCA integration module into WebSphere Service Registry and Repository

8. After successfully loading the file, you can go to **Service Documents** → **SCA Integration Modules** → **SCA Module Documents**, where you can see the new ITSO module that you just loaded.
9. Click the symbol in the Graph column for the ITSO module, which creates a visual representation of the module and all of its dependencies, similar to Figure 15-14 on page 435. This view allows you to analyze dependencies and relationships between the various artifacts that are used in the module.



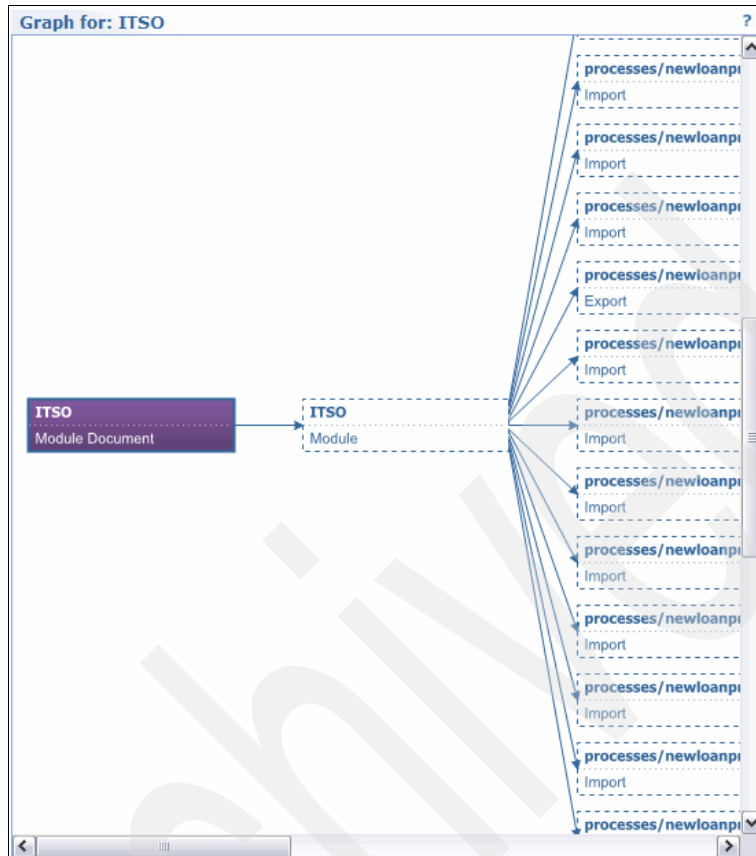


Figure 15-14 Graphical representation of the ITSO module

10. Go to **Service Documents** → **WSDL Documents** to see that a number of WSDL documents were imported as part of loading the SCA module, as well. A couple of XML schema files are also included.

**Note:** SCA libraries commonly contain WSDL files that have no <binding> element; instead, they only contain the port type, that is, the interface of a service. Bindings are then added later using those interfaces with exports and imports.

Next, we show you how WSDL files are federated from WebSphere Service Registry and Repository into the Business Services Fabric. WSDL files without bindings are not federated and no Fabric *Interface* definition is automatically created for them.

## 15.4.2 Publishing using the Eclipse plug-in

If you are developing services that do not use SCA, your technical service artifacts exist somewhere in your WebSphere Integration Developer workspace. You can publish these directly into WebSphere Service Registry and Repository without the need to export them to the file system first. For this, you must have the WebSphere Service Registry and Repository Eclipse plug-in installed. This plug-in is functional in any Eclipse-based tool, not just WebSphere Integration Developer.

Let us assume that you created an `AddressCleansing` service, which consolidates address information that comes from a variety of channels into a common, normalized form. We also assume that this service is developed in Java with the intent to deploy it on WebSphere Application Server.

To make this service reusable for composition and orchestration, its WSDL (we will call it `AddressService.wsdl`) and XML schema file (named `Address.xsd`) must be published to WebSphere Service Registry and Repository.

To publish using the Eclipse plug-in:

1. In WebSphere Integration Developer, go to the J2EE perspective, and find the WSDL/XML schema that you want to publish to WebSphere Service Registry and Repository. Select all of the files that you want to publish, right-click, and select **Service Registry** → **Publish Document(s)**.
2. Fill in description and versioning information for each file that is being published (Figure 15-15 on page 437).

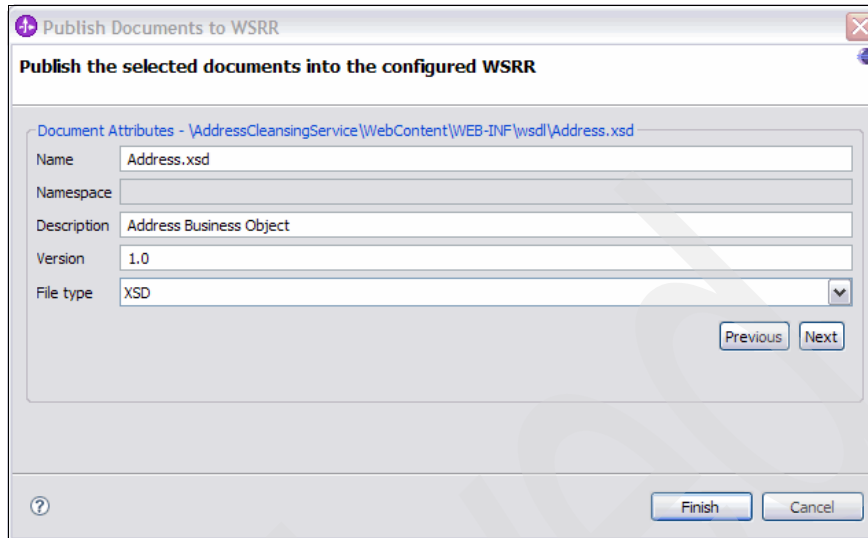


Figure 15-15 Adding information to published files

3. Click **Finish**. The files are published to WebSphere Service Registry and Repository.

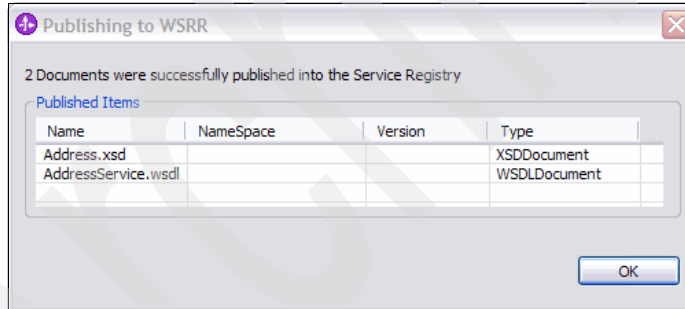


Figure 15-16 Completed publication to WebSphere Service Registry and Repository

4. Optionally, use the WebSphere Service Registry and Repository admin console to verify that the files were loaded successfully.

### 15.4.3 Adding classifications

The next step is to classify the newly published artifacts. Of course, the details of what the appropriate classifications are dependent on your business and technical environment. Industry standards exist that might serve as a good

starting point. For our example, we use a very simple classifications system, and we briefly take you through the steps needed to create this system.

To add classifications:

1. Open the WebSphere Service Registry and Repository admin console. Go to the Configuration perspective.
2. Go to **Active Configuration Profile** → **Classification Systems**.
3. Click **New**.
4. In the URI field, enter `http://www.itsobank.com/classification`.
5. In the Name field, enter `ITSOBank Services Classification`, as shown in Figure 15-17.

Classification Systems > New Classification System

Details of the New Classification System

Classification system Classes

Classification System Properties

URI  
`http://www.itsobank.com/classification`

Name  
`ITSOBank Services Classification`

☒ Name Language Variants

Comment

☒ Comment Language Variants

Save and Commit Cancel

Figure 15-17 Creating a new classification system

6. Click the **Classes** tab.
7. Click **Add Root Class**.
8. Change the Class ID to `http://www.itsobank.com/classification/lob#`.
9. In the Name field, enter `L0B`.
10. Click **OK**.
11. In the resulting window, select the `L0B` class option, and click **Add Child**.
12. Change the Class ID to `http://www.itsobank.com/classification/lob/common#`
13. In the Name field, enter `Common`.
14. Click **OK**.

15. Create additional child classes to LOB, just like we described in steps 11 - 14:

- Personal Banking
- Investment Banking
- Loans

16. Create another root class called ServiceLayer, and add child classes to it, just like in the previous step:

- Utility
- Task
- Entity

Your window should look like Figure 15-18.

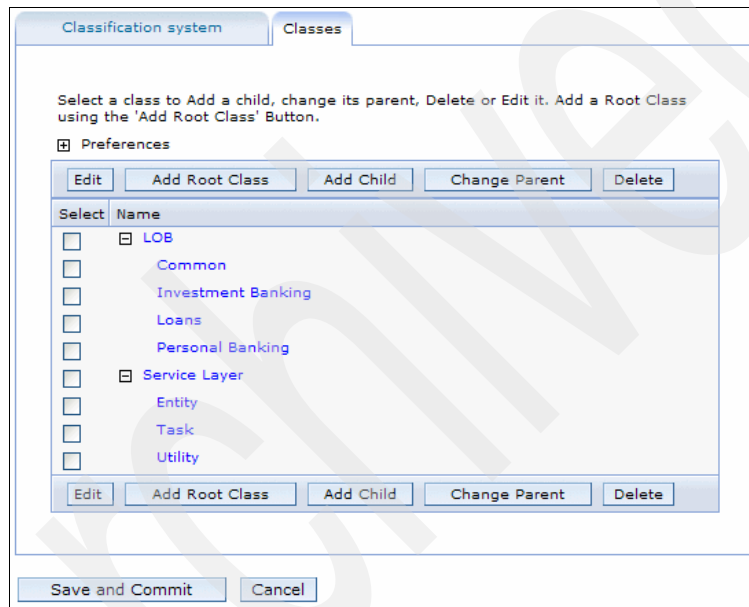


Figure 15-18 Adding classes to the classification system

17. Click **Save and Commit**, which creates the classification system that we can now use to classify our artifacts.

18. Switch to the Administrator perspective, go to **Service Documents** → **WSDL Documents**, and click **AddressService.wsdl** (or whatever the name of your published WSDL is).

19. In the following view, click **Relationships** → **Classifications**.

20. Expand **ITSO Services Classification**. The Address Cleansing service is not specific to any line-of-business; therefore, we classify it as a Common service.

Technically, it is fair to assume that the service is a `Utility` service. Select these two classifications, and click **Add**, as shown in Figure 15-19.

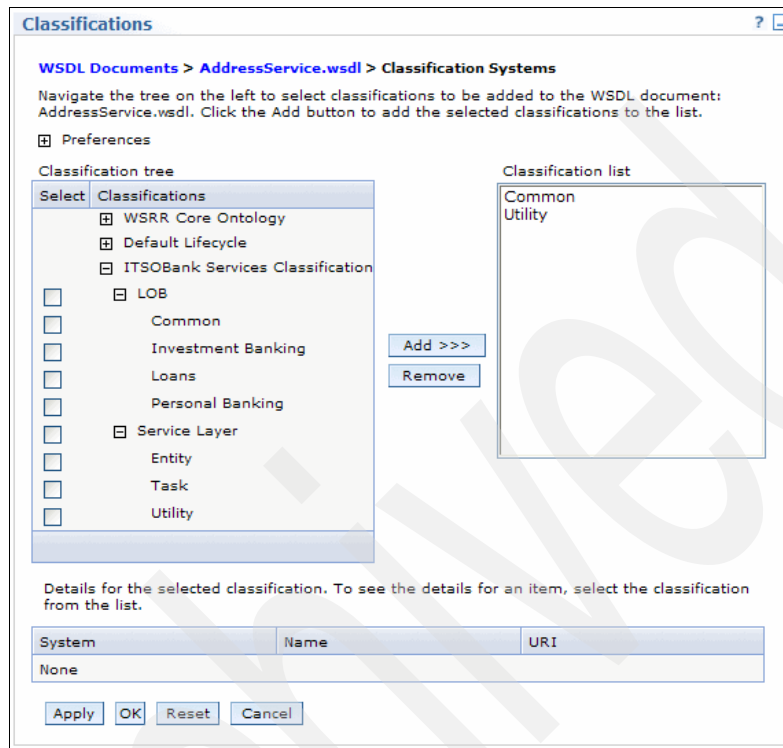


Figure 15-19 Adding classifications to the WSDL file

21. Click **OK**.

22. Repeat these steps for `Address.xsd`. This file is located under **Service Documents** → **XSD Documents**.

#### 15.4.4 Adding artifacts to a life cycle

The next step is to associate the new artifacts with a life cycle, which enables us to apply proper governance before any new artifact is reused elsewhere, for example, a WSDL document in `Modelled` state is reviewed by the responsible person to make sure that it follows in-house standards and conventions before it is promoted to `Deployed` state. Access control settings can ensure that only users with appropriate rights can transition artifacts from one state to the next.

The details of a good life cycle definition depend on your environment, your existing processes, and your defined roles and responsibilities. As before, we use

a simplified version (namely, the default life cycle that comes with WebSphere Service Registry and Repository) for our example.

To add artifacts to a life cycle:

1. The first step is to make the AddressService.wsdl file *governable*:
  - a. In the details view for the file, click the **Governance** tab. Assuming you have not configured a custom life cycle, the only initial transition that is offered is called Default transition.
  - b. Click the **Govern** button (Figure 15-20).

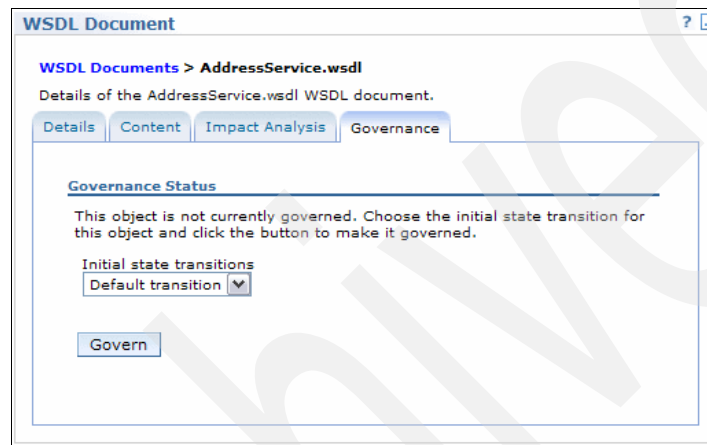


Figure 15-20 Making the WSDL file governable

2. The WSDL file is now part of the default life cycle and can be transitioned to the next possible state, as shown in Figure 15-21 on page 442.



Figure 15-21 The WSDL file is now part of the default life cycle

3. Click **Additional Properties** → **Governed Objects**. Note how the Address.xsd file was automatically added to the same life cycle because the WSDL file imports it. Hence, you do not have to make this artifact governable separately because it is treated as a set from now on when transitioning between states.
4. Return to the AddressService.wsdl governance view, and click **Transition** to move the file to Model state.
5. Transition the file to Assemble state by clicking **Transition** again.

For now, we will leave the AddressService.wsdl and Address.xsd files in their current state. We transition them to the next state later. For your reference, we list the states that we transition these artifacts through in Figure 15-3 on page 413.

## 15.5 Using WebSphere Service Registry and Repository as part of a Fabric project - filtering federated content

Before we start looking at how we can use services that are published to WebSphere Service Registry and Repository to build composite applications in the Fabric, we put some controls over which content is actually federated into the Business Services Repository. In the current release of the product, the federation project that we configured in the Fabric copies all applicable content



from WebSphere Service Registry and Repository into the BSR. No specific queries or filters can be associated with this federation. Most likely, this leads to the federation of unwanted content. One mechanism that we have available at this time to add a filter to this process is to use WebSphere Service Registry and Repository access control.

### 15.5.1 Defining WebSphere Service Registry and Repository permissions for Fabric federation

Simply put, we will restrict the access rights of the user ID that the federation project uses to access WebSphere Service Registry and Repository, so that we can control which content is copied across. WebSphere Service Registry and Repository supports the definition of very fine-grained permissions that we can leverage here. In our example, we define these permissions in a way that only artifacts that are associated with a certain life cycle state are federated.

In WebSphere Service Registry and Repository, permissions are linked with roles. These roles are mapped to *principals*, that is, the user IDs that a client uses for authentication with the server. We recommend that you create a specific user ID (and subsequently, an associated WebSphere Service Registry and Repository role) solely for the federation between WebSphere Service Registry and Repository and Fabric. In our example, we named this user ID `WSRRFabricAdmin`.

Assuming that this user ID already exists in your environment, the next step is to define an associated role in WebSphere Service Registry and Repository and map the two.

To define an associated role in WebSphere Service Registry and Repository to map to a principal:

1. Open the WebSphere Service Registry and Repository admin console. Do not log in as `WSRRFabricAdmin`; instead, use your administrator user ID. Switch to the Configuration perspective.
2. Click **Active Configuration Profile** → **Access Control** → **Roles**.
3. Click **New**.
4. In the Role Name field, enter `WSRRFabricAdmin` as the new role name. Click **OK**.
5. In the resulting view (Figure 15-22 on page 444), note that the new role has no users, groups, or permissions that are associated with it. This is no surprise, of course, because we just created this role.

New Delete				
<input type="checkbox"/> <input type="checkbox"/>				
Select	Role ▾	Users ▾	Groups ▾	Permissions ▾
<input type="checkbox"/>	Developer	1	0	2
<input type="checkbox"/>	Administrator	0	1	6
<input type="checkbox"/>	WSRRFabricAdmin	0	0	0
<input type="checkbox"/>	User	0	1	3
Total: 4				

Figure 15-22 Creating a new role in WebSphere Service Registry and Repository

6. in the **Users** column for WSRRFabricAdmin, click the 0.
7. In the resulting view, you can run a search to find the correct user ID to map to this role. In the Search field, enter WSRR\*, and click Search, which populates the Users/Groups area with existing user IDs that satisfy your search pattern. One of the listed user IDs should be WSRRFabricAdmin (or the user ID that you created before). Select this user ID, and click **Add**, as shown in Figure 15-23.

Details

Search

WSRR\*

Search

Include

Users

Users/Groups

AllAuthenticatedUsers

Add >>>

Remove

Selected Users

ATOST2\WSRRFabricAdmin

Selected Groups

Apply

OK

Reset

Cancel

Figure 15-23 Mapping a user ID to a WebSphere Service Registry and Repository role

8. Click **OK**. In the resulting view, you now have one user mapped to the WebSphere Service Registry and RepositoryFabricAdmin role. However, we have not yet defined any permissions for this role, so that is what we will do next.
9. In the **Permissions** column for user WSRRFabricAdmin, click the 0.
10. As expected, there are no permissions listed. Click **Create a New Permission**.

Permissions in WebSphere Service Registry and Repository are defined in pairs that consist of a *permission action* and a *permission target*. Simply put, the action describes the type of activity that can be performed on the target. Six different actions are supported: the four basic *CRUD* actions (create, retrieve, update, delete) and two actions that are related to governance, namely the step of making an artifact governable and its transitioning to a new state.

For the federation between Fabric and WebSphere Service Registry and Repository, we only need the retrieve action. We will get to defining the target in a minute.

11. Name the new permission WSRRFabricPermission (or pick any other name of your liking).
12. From the Permission Action pull-down list, select the **Retrieve** action.
13. The target for the permission is defined in the form of an XPath expression that selects the correct artifact(s) based on their position in the logical tree that WebSphere Service Registry and Repository uses to manage artifacts. The details about this logical tree and its elements are in the product infocenter. For our Fabric federation permission, we want to make sure that all WSDL-defined ports and interfaces, which are in the Deployed or Managed state, can be retrieved. The appropriate XPath statement that you should enter into the Permission Target (XPATH) field is:  

```
/WSRR/LogicalObject[classifiedByAnyOf('http://www.ibm.com/xmlns/prod/serviceregistry/6/0/governance/DefaultLifecycle#State3','http://www.ibm.com/xmlns/prod/serviceregistry/6/0/governance/DefaultLifecycle#State4')]
```

Click **OK**, as shown in Figure 15-24 on page 446.

Figure 15-24 Creating a new permission in WebSphere Service Registry and Repository

14. The new permission is created and immediately associated with the WSRRFabricAdmin role. Back in the **Manage Roles** view, you should now see that one permission exists for this role, as shown in Figure 15-25.

<input type="checkbox"/>	WSRRFabricAdmin	1	0	1
--------------------------	-----------------	---	---	---

Figure 15-25 The new permission is added to the role

15. Restart your WebSphere Service Registry and Repository instance to put the new settings in effect.

**Note:** By default, the Administrator and User roles are mapped to the principal named AllAuthenticatedUsers, which means that any authenticated user is given Administrator privileges. To make sure that only the appropriate users get those privileges, remove this mapping.

## 15.5.2 Selective federation

Given the new permission setting, the Fabric federation project is only allowed to retrieve artifacts that have the appropriate state. Because we have not yet transitioned any of the artifacts that we published to WebSphere Service Registry and Repository into either the Deploy or Manage state, none of them will be federated.

To federate:

1. Start your Fabric server instance now. After it is up and running, open the Fabric admin console.
2. Click **My Services** → **Business Services Repository** → **Search Repository**.
3. Clear the **All** option, and select the **Interfaces** option.
4. Click **Search**. As we mentioned, no WSDL documents were transitioned to an appropriate state to be federated, so none of them show up in the list, as shown in Figure 15-26.

**Search**

Resources Subscribers Policies

**Search**

Search For: ☐ All ☐ Application Suites ☐ Applications ☐ Business Services ☒ Interfaces ☐ Composite Services ☐ Endpoints

Name:

**Search**

**Resource Selection**

View  rows at a time

2 rows

Name ↓	Type	Publisher
<a href="#">Event Consumer</a>	WSDL Interface	System
<a href="#">TemperatureConversionsSoapType</a>	WSDL Interface	WsrrProvider

Page 1 of 1

Figure 15-26 No artifacts were federated from WebSphere Service Registry and Repository

5. Open the WebSphere Service Registry and Repository admin console, and switch to the **Administrator** perspective.
6. Click **Service Documents** → **WSDL Documents**.
7. In the resulting list view, click `AddressService.wsdl`, which is the service that we published to the registry earlier.
8. Click the **Governance** tab for this WSDL document. Note that we left this artifact in the Assembled state.

9. Click the **Transition** button to move the artifact into the next state, called Deploy, as shown in Figure 15-27.

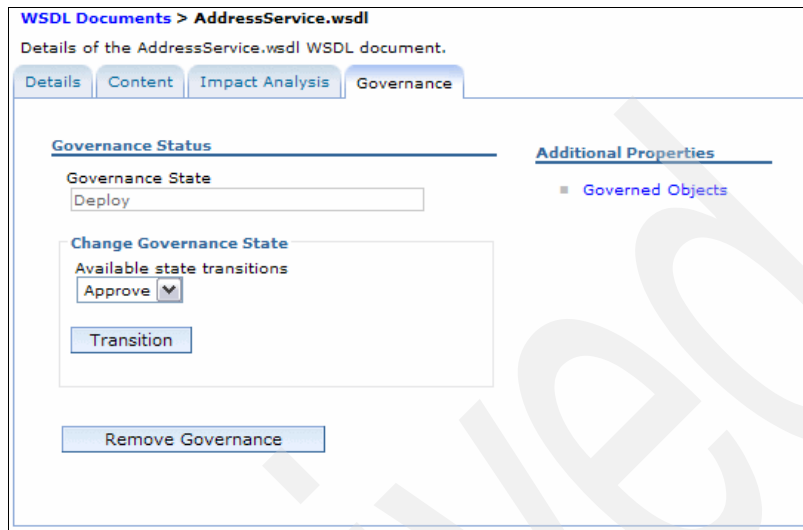


Figure 15-27 The AddressService.wsdl is in Deploy state

10. This state transition effectively results in the WSRRFabricAdmin role obtaining access to the AddressService.wsdl file and, implicitly, its imported XML Schema, Address.xsd. This also means that the WSDL file is now federated into the Fabric Business Services Repository, since we are using the WSRRFabricAdmin user ID (which is mapped to the WSRRFabricAdmin role) for the federation. We can easily verify this by opening the Fabric admin console and doing a search on all interfaces.

In the Fabric admin console, click **My Services** → **Business Services Repository** → **Search Repository**.

11. Clear the **All** option, and select the **Interfaces** option. Click **Search**. The AddressService is listed as one of the interfaces, as shown in Figure 15-28 on page 449.

Resources
Subscribers
Policies

Search

Search For

☐ All  
☐ Application Suites  
☐ Applications  
☐ Business Services  
☒ Interfaces  
☐ Composite Services  
☐ Endpoints

Name

Search

Resource Selection

View 10 rows at a time

3 rows

Name ↓	Type	Publisher
<a href="#">AddressService</a>	WSDL Interface	WsrriProvider
<a href="#">Event Consumer</a>	WSDL Interface	System
<a href="#">TemperatureConversionsSoapType</a>	WSDL Interface	WsrriProvider

Page 1 of 1

Figure 15-28 The AddressService interface is federated from WebSphere Service Registry and Repository

You might have to wait a while before the new interface is listed here. The federation between the Fabric and WebSphere Service Registry and Repository only happens at certain time intervals, depending on the Time To Live setting in the federation project.

## 15.6 Using WebSphere Service Registry and Repository as part of a Fabric project - finding content

Up to this point, we showed you how technical service artifacts are published to WebSphere Service Registry and Repository and how they are classified and added to a life cycle there. Moreover, we explained how you can use WebSphere Service Registry and Repository access control definitions to limit the artifacts that get federated into the Business Services Repository. In this section, we describe how you can use these published and federated services to create a composite application.

Two complimentary methods exist to find existing services and to reuse them in a Fabric project. You can use the query mechanisms that WebSphere Service Registry and Repository offers to find services that are relevant for your purpose, or you can search the Fabric Business Services Repository for the appropriate

interfaces and endpoints. For the remainder of this chapter, we look at both approaches in more detail.

## 15.6.1 Finding artifacts in WebSphere Service Registry and Repository

WebSphere Service Registry and Repository offers several ways to query its stored artifacts: a Web UI, plug-ins for Eclipse, Visual Studio .Net, and an API that you can use in any custom application. Now we briefly show you how to use the Web UI and the Eclipse plug-in to find the AddressService artifacts that we published earlier.

### Using the WebSphere Service Registry and Repository Web UI application

To use the WebSphere Service Registry and Repository Web UI application:

1. Open the WebSphere Service Registry and Repository admin console.
2. In the **Administrator** perspective, click **Queries** → **Query Wizard**.
3. In the **Prepare to Run a Query** → **Select the Type of Entity to Query** pull-down list, click **WSDL Documents**. Click **Next**.
4. In the resulting dialog, we can enter more specific query details. In our example, we only use classifications as additional query criteria. Depending on the number of services that you published in WebSphere Service Registry and Repository, you might want to further filter the list of returned results.  
Click the **Add** button next to the **Classifications** area.
5. In the Classification tree pane, expand the **ITSO Services Classification** node, then expand both the **LOB** and **Service Layer** nodes.
6. Select the **Common** and the **Utility** classifications, and click **Add**, as shown in Figure 15-29 on page 451.



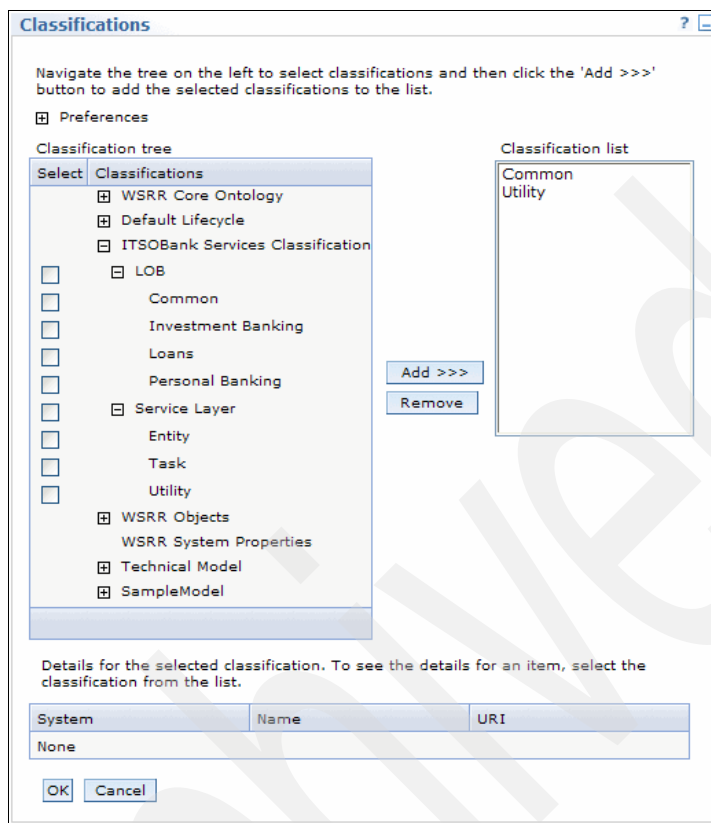


Figure 15-29 Selecting classifications for a WebSphere Service Registry and Repository query

7. Click **OK**.
8. Click **Next**.
9. Click **Finish**.

The query you just defined should return only one document, namely the AddressService.wsdl document that we classified accordingly, as shown in Figure 15-30 on page 452.

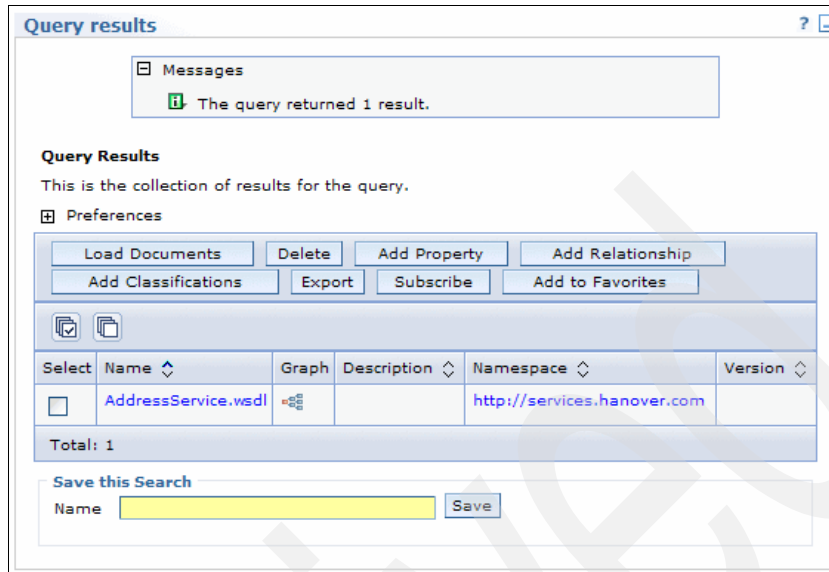


Figure 15-30 WebSphere Service Registry and Repository query results window

10. Let us assume that you want to run this type of query more often. The Web UI lets you save your query so that you can run it again at a later time.  
Enter Common Utility WSDL documents in **Save this Search** → **Name**. Click **Save**.
11. In the main menu, click **My Service Registry** → **My Saved Searches**.
12. Select the search named **Common Utility WSDL documents**, and click **Run Search**, which results in the same WSDL document being returned.

## Using the Eclipse plug-in

If you want to actually use the artifacts that you found in WebSphere Service Registry and Repository using the Web UI, you have to export and then import them into your development tool, for example, WebSphere Integration Developer. Leveraging the WebSphere Service Registry and Repository Eclipse plug-in lets you do this from within the tool.

To use the Eclipse plug-in:

1. In WebSphere Integration Developer, go to the **Business Integration** perspective (you can use any other perspective, too, but we use this one for our example).
2. In the main menu, click **Window** → **Show view** → **Other**.

3. Click **Service Registry** → **Service Registry**. You only see this view being offered if you installed the WebSphere Service Registry and Repository Eclipse plug-in.
4. Anywhere in the new view, right-click, and select **Retrieve**.
5. In the resulting pop-up window, click the **Choose** button that is located next to the Classifications area. This retrieves all existing classifications from WebSphere Service Registry and Repository.
6. In the next window, expand the **ITSO Services Classifications** node, followed by the **LOB** and **ServiceLayer** nodes in it. Select **Common**, and click the >> button, as shown in Figure 15-31. Do the same for **Utility**.

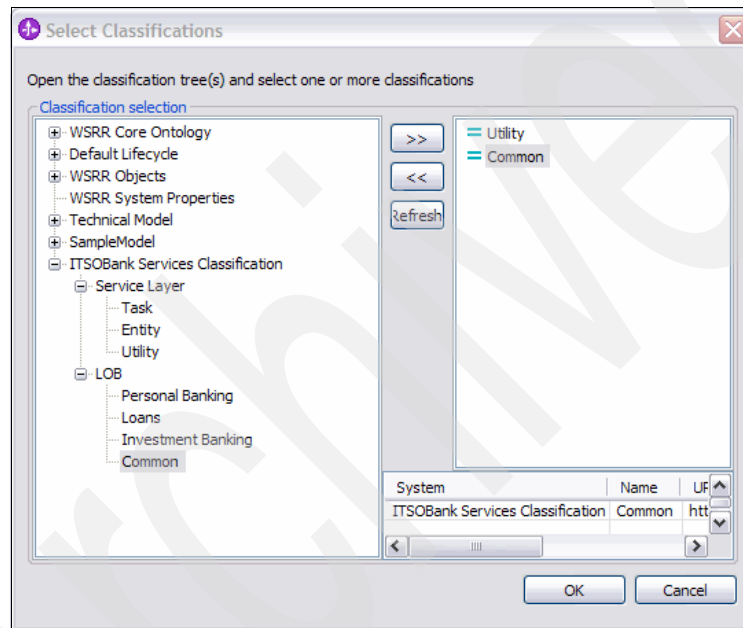


Figure 15-31 Choosing classifications in the WebSphere Service Registry and Repository Eclipse plug-in

7. Click **OK**.
8. Back in the Search window, select the **Search for** → **WSDL docs** option.
9. Click **Search**. Running this query returns the AddressService.wsdl, as shown in Figure 15-32 on page 454.

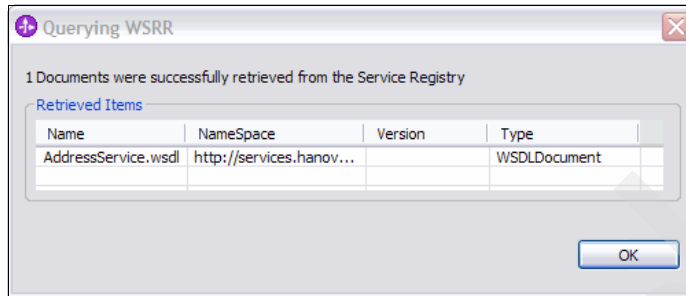


Figure 15-32 WebSphere Service Registry and Repository query results

10. Click **OK**. The returned WSDL file is now listed in the Service Registry view, under WSDL Documents. From there, you can easily import it into your WebSphere Integration Developer workspace by right-clicking and selecting the **Import Document** option.

After you import the document, you can use it in your business integration module, for example, call the underlying service from within a business process, or, more interestingly in the context of this book, route requests to this service through the Dynamic Assembler.

## 15.6.2 Finding artifacts in the Business Services Repository

We already described how interface definitions that exist in WebSphere Service Registry and Repository are automatically federated into the Business Services Repository, which means that we can find them there using the Fabric admin console. In addition, we can import them into Composition Studio. In the next sections, we show you both ways.

### Using the Fabric admin console

We already described how to use the Fabric admin console to find artifacts in 15.5.2, “Selective federation” on page 446. The WSDL document that was federated from WebSphere Service Registry and Repository was turned into an Interface in the Business Services Repository. What we did not use was the Name field in the Search Repository panel to further restrict the results list; however, it was not necessary because we have not yet created many artifacts. But as your service portfolio grows, it might be a good idea to limit the number of entities that you search for.

### Using Composition Studio

We perform the actual composition of services and building out of the associated model that is then used at runtime in Composition Studio. Part of this is to find

the right services for composition to begin with. Content that is federated from WebSphere Service Registry and Repository into the Business Services Repository is automatically replicated into Composition Studio, which makes that content available for developing composite applications.

Remember that when we created the WebSphere Service Registry and Repository federation project, it got assigned a namespace, called WSRR Federation Sponsored NS 1. All content that is federated from WebSphere Service Registry and Repository is owned by that namespace and by the federation project.

Any new Fabric project that you might create as part of developing a composite application has to import the WebSphere Service Registry and Repository namespace so that it can reuse the interfaces that are federated from WebSphere Service Registry and Repository. In Composition Studio, you have the option of associating a Fabric project with any existing Business Service project or associating it with the WebSphere Service Registry and Repository federation project itself. In the first case, all content that is federated from WebSphere Service Registry and Repository is read-only because it came through an import of the owning namespace. In the second case, you have full write access to the federated content.

To use Composition Studio:

1. Open the Fabric admin console. Click **My Services** → **Configure Projects**.
2. Click **Create a Project**.
3. In the Project Name field, enter New Composite Application.
4. Leave the Project Type as Business Service. In the Team Organization field, select IBM.
5. Click **Create Project**.
6. In the resulting window, click the **Namespaces** tab.
7. Under normal circumstances, we would now create a new namespace for this project (unless we had already done so). Here, we will skip this step and simply import the WebSphere Service Registry and Repository federation namespace. Click **Import Namespaces**.
8. Click **Project Selection** → **Project**, and click **WSRR Federation**.
9. Click **Import Namespaces**.
10. In the resulting window, select the **WSRR Federation Sponsored NS 1** namespace, and click **Import Namespaces**, as shown in Figure 15-33 on page 456.

Project Details Project Team **Namespaces**

**Import Namespaces**

Fields marked with an asterisk (\*) are required fields.

**Info**

- Select a **Project** and **Namespaces** to import.

**Project Selection**

\* **Project** WSRR Federation

**Namespace Selection**

1 rows

<input type="checkbox"/>	Namespace	Type	Description
<input checked="" type="checkbox"/>	WSRR Federation Sponsored NS 1	Sponsored	

**Import Namespaces** **Cancel**

Figure 15-33 Importing the WebSphere Service Registry and Repository federation namespace

This action makes the content that is federated from WebSphere Service Registry and Repository available to the new project.

11. Click the **Project Details** tab, and click **Save**. We are now ready to move into Composition Studio.
12. In WebSphere Integration Developer, go to the **Business Service** perspective.
13. In the Business Service Explorer pane, right-click, and select **New** → **Fabric Project**.
14. In the resulting dialog, In the Project Name field, enter New Composite Application. Click **Next**.
15. Click **Update Project**.
16. After the replication with the Business Services Repository is completed, click **Next**.
17. From the **Fabric Project** pull-down list, select **New Composite Application**.
18. Click **Finish**.

After the new project is created, you can expand it in the Business Service Explorer. Note how the AddressService was imported. The original WSDL document was split into an interface and an endpoint, as shown in Figure 15-34 on page 457.

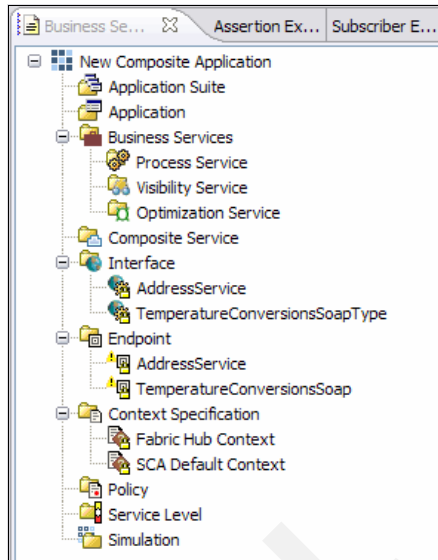


Figure 15-34 A new Fabric project in Composition Studio with WebSphere Service Registry and Repository content imported

However, the interface and endpoint were imported as read-only. After you open them in the Advanced Editor, you cannot make any changes to it because they were included in this project through importing the owning namespace.

To actually change the federated entries, we must create another Fabric project in Composition Studio:

1. In the Business Service Explorer pane, right-click, and select **New** → **Fabric Project**.
2. In the Project name field, enter WSRR Federation, and click **Next**.
3. Click **Update Project**.
4. Click **Next**.
5. From the Fabric Project pull-down list, select WSRR Federation.
6. Click **Finish**.

The same interface and endpoint are imported, but this time, you can change them.

**Note:** While creating a Fabric project in Composition Studio that is associated with the WebSphere Service Registry and Repository federation project allows you to change federated content, these changes are not replicated back into WebSphere Service Registry and Repository. Therefore, you should approach this with extreme caution because it will put both repositories out of sync.

### 15.6.3 Using both approaches together

The WebSphere Service Registry and Repository query mechanism allows you to search for relevant artifacts based on classifications and other criteria. After you find the content that is relevant to you, you can easily find the same content in the Business Service Repository (to which it was automatically federated), using either the Fabric admin console or Composition Studio. Especially, if you have a large set of services defined, you might want to use a WebSphere Service Registry and Repository query to apply finer-grained criteria to your search, and then use that to quickly find the correct artifacts in the Business Services Repository to build a composite application.

Keep in mind, however, that only the WebSphere Service Registry and Repository Eclipse plug-in allows you to directly import the appropriate WSDL and XML schema files directly into your workspace, where you can use them for the actual implementation of the composite application.





## **Integrating with IBM Tivoli Composite Application Manager for SOA**

A Fabric solution typically spans across multiple channels and disparate systems to provide integrated services. It is an important requirement for any enterprise to manage and monitor several systems that are involved in building composite applications.

In this chapter, we introduce some of the basics of service-oriented architecture (SOA) management. Further we look at how to integrate a Service Component Architecture (SCA)-based WebSphere Business Services Fabric solution with IBM Tivoli Composite Application Manager for SOA.

## 16.1 SOA management for composite business applications

A typical composite business application spans across multiple architectural layers, such as business processes, service components, service consumers, and operational IT systems. Managing the SOA infrastructure upon which composite business applications are built needs careful attention to address various challenges. Because enterprise SOA infrastructures matured over a period of time, the need for management solutions has emerged to address:

- ▶ Understand the end-to-end flow of business processes.
- ▶ Meet the required Non Functional Requirements (NFRs) and Service Level Agreements (SLAs).
- ▶ Ensure that operational systems that provide integrated services are available as per the SLA requirements.
- ▶ Monitor any potential security threats for the SOA infrastructure.
- ▶ Understand the relationship between various participating services in the composite business application.
- ▶ Analyze performance loop holes.

Refer to the following Redbooks publications for a detailed discussion about SOA management and its challenges:

- ▶ *Best Practices for SOA Management*, REDP-4233
- ▶ *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240

For composite business applications, SOA management requirements span across multiple architectural layers. These layers are best explained by the IBM SOA Foundation Reference Architecture, as shown in Figure 16-1 on page 461.

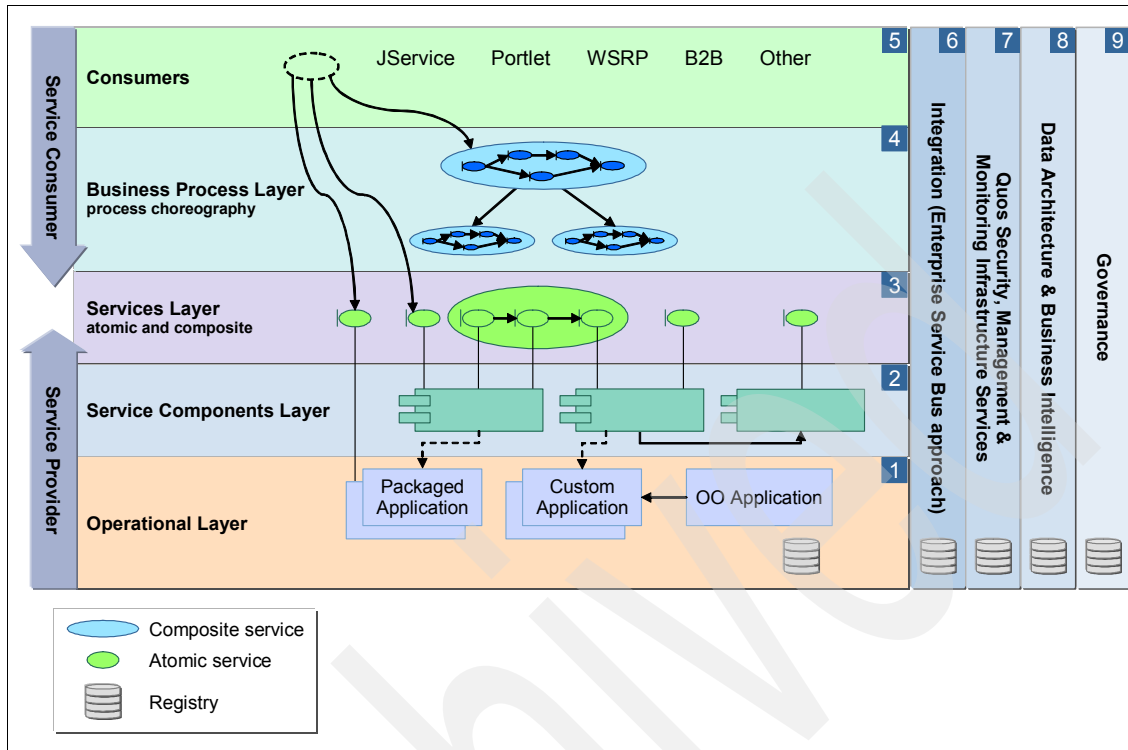


Figure 16-1 IBM SOA Foundation Reference Architecture - Solution Stack overview

The most important layer from the perspective of composite business applications is the services layer, which integrates various operational systems, service components, business policies, and service consumers through provisioned channels. You can leverage an SOA management solution that is implemented for business services across the enterprise to:

- ▶ Understand how services relate to each other for providing a business functionality.
- ▶ Assess the dependency of business services on IT Infrastructure and the business process layer.
- ▶ Define and refine business related goals.
- ▶ Control the message flow for services infrastructure through management mediations, such as filter, log, transform, and route, which is specifically helpful because the composite business application spans across multiple architectural layers.
- ▶ Understand the performance of a specific service with transactional metrics of each request.

- ▶ Provide relationships between service requests and the implementation artifacts (for example, a JDBC request, Java Beans, SCA Plain Old Java Object (POJO) component, and so on).
- ▶ Monitor the health of the operational systems that support the services implementation (for example, an Enterprise Resource Planning (ERP) system, a custom billing system, and so on).
- ▶ Provide inputs to solve operational problems (for example, low end-to-end response times, exhausted thread pool, and so on).

It is evident that there is a cost that is associated with implementing and maintaining such SOA management solutions across the enterprise. A centralized SOA management policy set across the enterprise by the SOA governance body is a critical success factor for such solutions.

## 16.2 Overview of IBM Tivoli Composite Application Manager for SOA

IBM Tivoli Composite Application Manager (ITCAM) for SOA monitors, manages, and controls business services that run on supported application servers. ITCAM for SOA provides some of the following key features:

- ▶ Provides a comprehensive and configurable dashboard for viewing service monitoring data through Tivoli Enterprise Portal Server (TEPS):
  - Service Management Agent Environment shows:
    - Performance Summary
    - Message Summary
    - Fault Summary
  - Service Management Agent shows monitoring agent configuration summary, data collectors, monitoring profiles, and filters.
  - Mediation configuration shows mediation primitive entries for mediation on Service Component Architecture (SCA) components in WebSphere Process Server or WebSphere Enterprise Service Bus.
  - Message arrival view shows the message arrival rate and events based on the message arrival critical situation, which you can use for looking at the throughput rates.
- ▶ Heterogeneous platform support:
  - Supports Microsoft.NET, BEA WebLogic, IBM WebSphere Application Server (WAS) JAX-RPC and SCA, IBM DataPower appliance, WAS Community Edition, and JBoss.

- CICS® Transaction Server environment
- SAP® NetWeaver environment
- ▶ Leverages Tivoli Enterprise Portal situations to check thresholds. ITCAM for SOA provides some predefined situations that you can customize. Some of the pre defined situations are:
  - Number of messages received by a service within a time window
  - Size of the messages
  - Faults
- ▶ Provides a list of services and operations that are monitored in the environment.
- ▶ Leverages Tivoli Enterprise Portal workflow and policy editor for threshold-triggered action sequences.
- ▶ Provides basic mediation support with the ability to filter or reject Web services call messages from a particular client or service. It can log request and response messages for analysis.

The building blocks of ITCAM for SOA consists of the following logical components:

- ▶ **Data Collector**  
Collects the services data that is appropriate to the environment in a non-intrusive fashion.
- ▶ **Enterprise Monitoring Agent**  
Works as a data consolidator. Responsible for collecting data from data collectors and forwarding them to Tivoli Monitoring Server.
- ▶ **Web Services Navigator**  
An Eclipse-based navigator that can process the collected log files and provide visual representations of various characteristics of the monitored data.
- ▶ **Mediation Primitives**  
Allows control of mediation primitives within the WebSphere Enterprise Bus and WebSphere Process Server.

To know more about the ITCAM product family, refer to the following resources:

- ▶ **ITCAM for SOA Infocenter**  
<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamsoa.doc/welcome.htm>
- ▶ *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151.

## 16.3 Integration between ITCAM for SOA and WebSphere Business Services Fabric

We now have some ideas about what ITCAM for SOA is and how it fits into the enterprise-hosting infrastructure for providing management and monitoring functionalities. Let us now look at how the integration of WebSphere Business Services Fabric with ITCAM for SOA works.

Figure 16-2 shows the components of how a composite business application environment interacts with the Tivoli Enterprise Monitoring Server infrastructure.

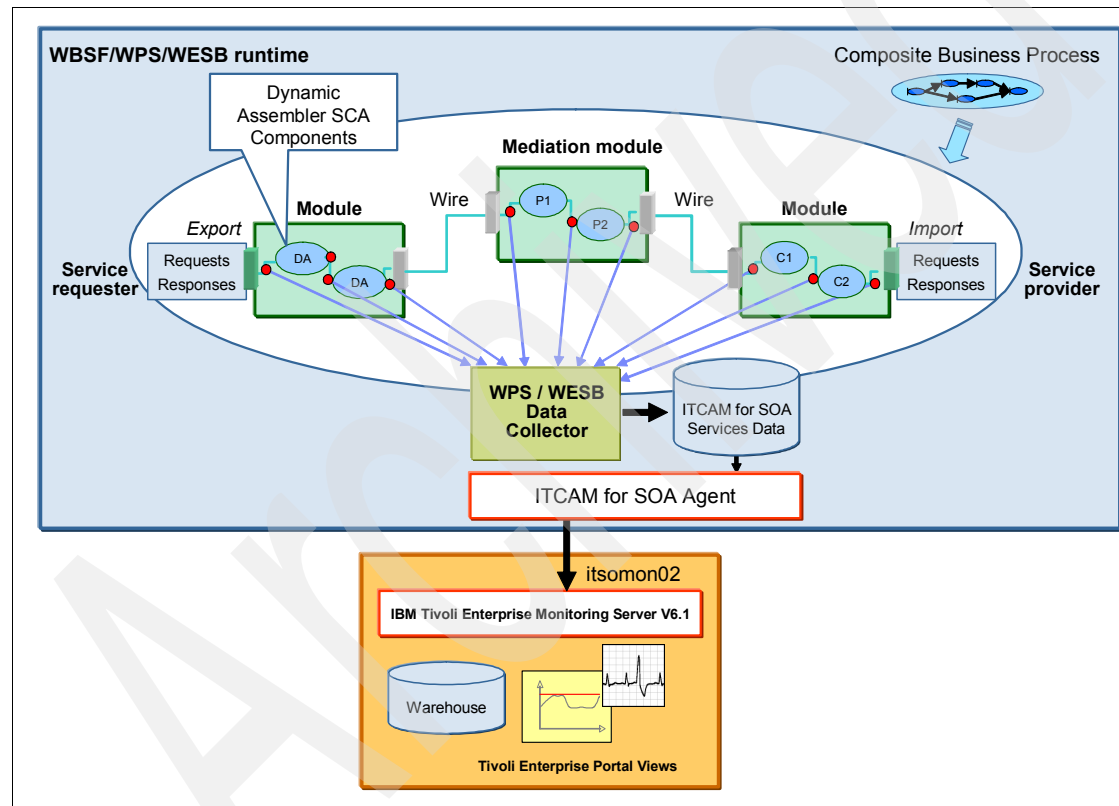


Figure 16-2 Composite business application environment with Tivoli Enterprise Monitoring server

Integration between WebSphere Business Services Fabric and ITCAM for SOA is achieved through support for underlying SCA components. Dynamic Assembler is an SCA component and the data collector implemented for WebSphere Process Server SCA components captures the messages that flow through these special types of SCA components. The Tivoli Enterprise

Monitoring Server captures, collates, and transforms into different views this data and other types of monitoring data (Web services data for instance).

As the Dynamic Assembler (DA) represents the service endpoints that are associated with business policies, monitoring DA components is an effective and efficient way to obtain an end-to-end view of the business process flow; therefore, it is not sufficient to just capture monitoring information for Dynamic Assembler invocations for a business process. We recommend that you make use of different types of data capturing techniques, such as Web services, transactions, database, operating systems, and so on, to get a holistic picture of the operational environment.

In the following sections, we explain how to configure a simple IBM Tivoli Enterprise Monitoring Server infrastructure for ITSOBank to capture and view monitoring information of the loan process, as we explained in previous chapters.

## **16.4 Installing and configuring ITCAM for SOA in ITSOBank**

In this section, we give a high-level overview of the Tivoli Enterprise Monitoring infrastructure that we are considering for ITSOBank. We will further look at how you can configure composite business applications that are delivered on WebSphere Business Services Fabric to capture services-related data.

### **16.4.1 Infrastructure setup**

ITSOBank's SOA management infrastructure is configured by having a central monitoring server. Monitoring agents are installed on different application nodes (where supported). Table 16-1 on page 466 gives the list of products that are installed on different nodes.

Table 16-1 Installation components on specific nodes

Installation node	Installation components
Monitoring Server	<ul style="list-style-type: none"> <li>▶ Microsoft Windows Server® 2003 Enterprise Edition</li> <li>▶ IBM DB2 UDB Enterprise Edition v8.2 with fix pack 16</li> <li>▶ Tivoli Enterprise Monitoring Server v6.1: <ul style="list-style-type: none"> <li>– Tivoli Enterprise Monitoring Server (TEMS)</li> <li>– Tivoli Enterprise Portal (TEP) Server</li> <li>– TEP Desktop Client</li> <li>– Tivoli Enterprise Monitoring Agents <ul style="list-style-type: none"> <li>• Tivoli Enterprise Monitoring Agent Framework</li> <li>• Warehouse Proxy</li> <li>• Monitoring Agent for OS</li> <li>• Universal Agent</li> <li>• Summarization and pruning agent</li> <li>• *other agent support as needed</li> </ul> </li> </ul> </li> <li>▶ Tivoli Monitoring Application Support for ITCAM for SOA. - Server Component</li> </ul>
Service Consumers and Service Providers  WebSphere Business Services Fabric Server, WebSphere Application Servers	IBM Tivoli Composite Application Manager for SOA v6.1 - Agent component
Developer Node	(optional) - ITCAM for SOA Tools - Eclipse-based Web Services navigator.

A lot is written about how to install and configure ITCAM for SOA infrastructure. Refer to *IBM Tivoli Composite Application Manager for SOA Installation and User's Guide*, GC32-9492 for a detailed discussion.

The ITSOBank composite business applications are deployed using various types of servers. Figure 16-3 on page 467 shows a basic view of different types of service providers, service consumers, and mediation modules that feed data into a central Tivoli Enterprise Monitoring Server ITSOMON02.



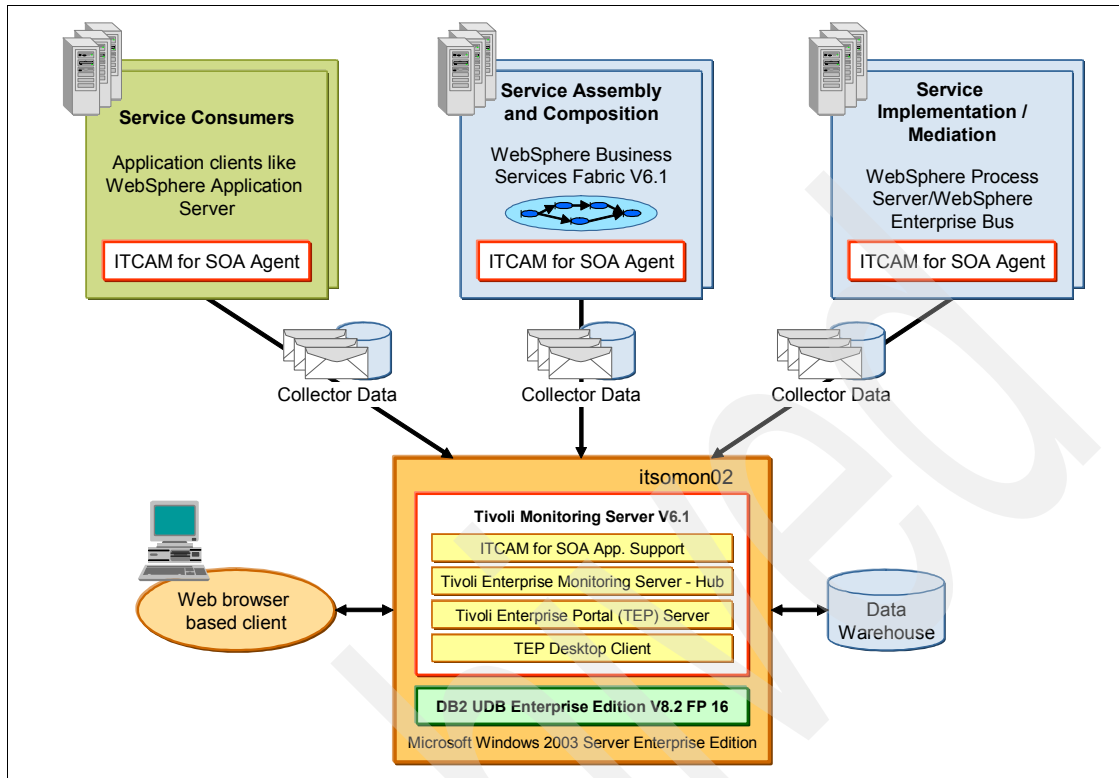


Figure 16-3 ITSOBank Tivoli Monitoring infrastructure

If everything goes fine, you should have the Tivoli Enterprise Monitoring Server configured and running, as shown in Figure 16-4.

Manage Tivoli Enterprise Monitoring Services - TEMS Mode - [Local Computer]										
Actions Options View Windows Help										
Service/Application	Task/SubSystem	Configured	Status	Startup	Account	Desktop	HotStdby	Version	Host	Port
Eclipse Help Server	HELPSVR	Yes	Stopped	Auto	LocalSystem	No	No	3.0.1		
Tivoli Enterprise Portal Browser	Browser	Yes		N/A	N/A	N/A	N/A	06.10.05.01	localhost	
Tivoli Enterprise Portal Desktop	Desktop	Yes		N/A	N/A	N/A	N/A	06.10.05.01	ITSOMON02	
Tivoli Enterprise Portal Server	KFWSRV	Yes (TEMS)	Started	Auto	LocalSystem	No	No	06.10.05.01		
Universal Agent	Primary	Yes (TEMS)	Started	Auto	LocalSystem	No	No	06.10.05.01		
Warehouse Summarization and Pruning Agent	Primary	Yes (TEMS)	Started	Auto	LocalSystem	No	No	06.10.05.01		
Monitoring Agent for Windows OS	Primary	Yes (TEMS)	Started	Auto	LocalSystem	Yes	No	06.10.05.01		
Warehouse Proxy	Primary	Yes (TEMS)	Started	Auto	LocalSystem	No	No	06.10.05.01		
ITCAM for SOA	Primary	Yes (TEMS)	Started	Auto	LocalSystem	No	No	06.10.00.00		
Tivoli Enterprise Monitoring Server	TEMS1	Yes	Started	Auto	LocalSystem	No	No	06.10.05.01		

Figure 16-4 Tivoli Enterprise Monitoring Services on ITSOMON02 server

On each of the service consumer (Web-service clients) and service provider server nodes, the ITCAM for SOA agent is configured and started, as shown in Figure 16-5.

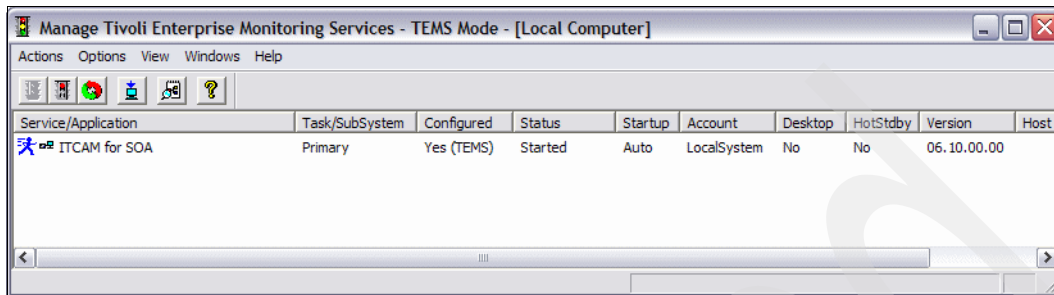


Figure 16-5 ITCAM for SOA Agent running on WebSphere Business Services Fabric server

ITCAM for SOA agents are configured to send the captured service-related data to the central server ITSOMON02. Tivoli Enterprise Monitoring Agent on ITSOMON02 for the Tivoli Enterprise Monitoring Server collates the data that is collected from various servers. Figure 16-6 shows the configuration details for the ITCAM for SOA agent on WebSphere Business Services Fabric node.

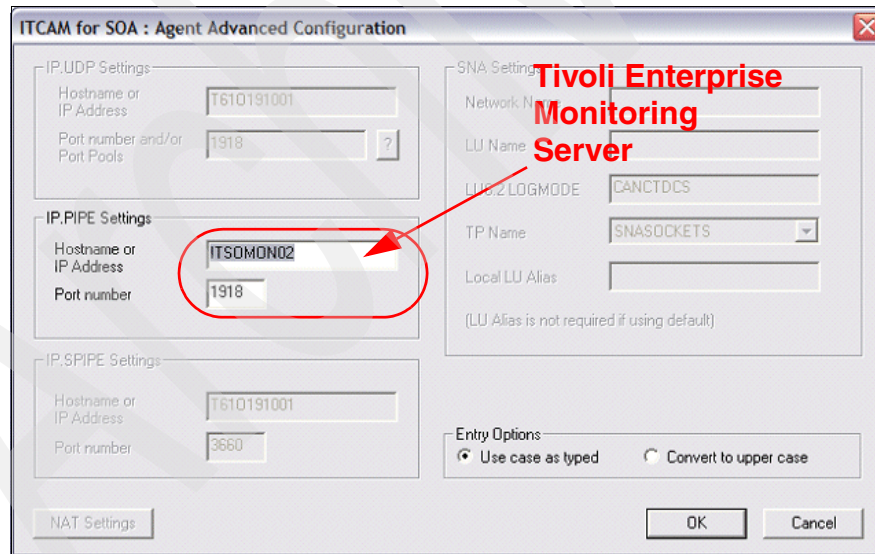


Figure 16-6 ITCAM for SOA Agent configuration on Fabric server

To make sure that the agents on the Fabric server can communicate with the Tivoli Enterprise Monitoring Server:

1. Open the Manage Enterprise Monitoring Services console.
2. Make sure that ITCAM for SOA agent is started. Right-click the agent to open the context menu.
3. Click **Advanced** → **View Trace Log**.
4. Select the latest log (based on the time stamp of the log), and click **OK**.
5. Make sure that ITCAM for SOA can connect to the Tivoli Enterprise Monitoring Server by observing the log details, as shown in Figure 16-7.

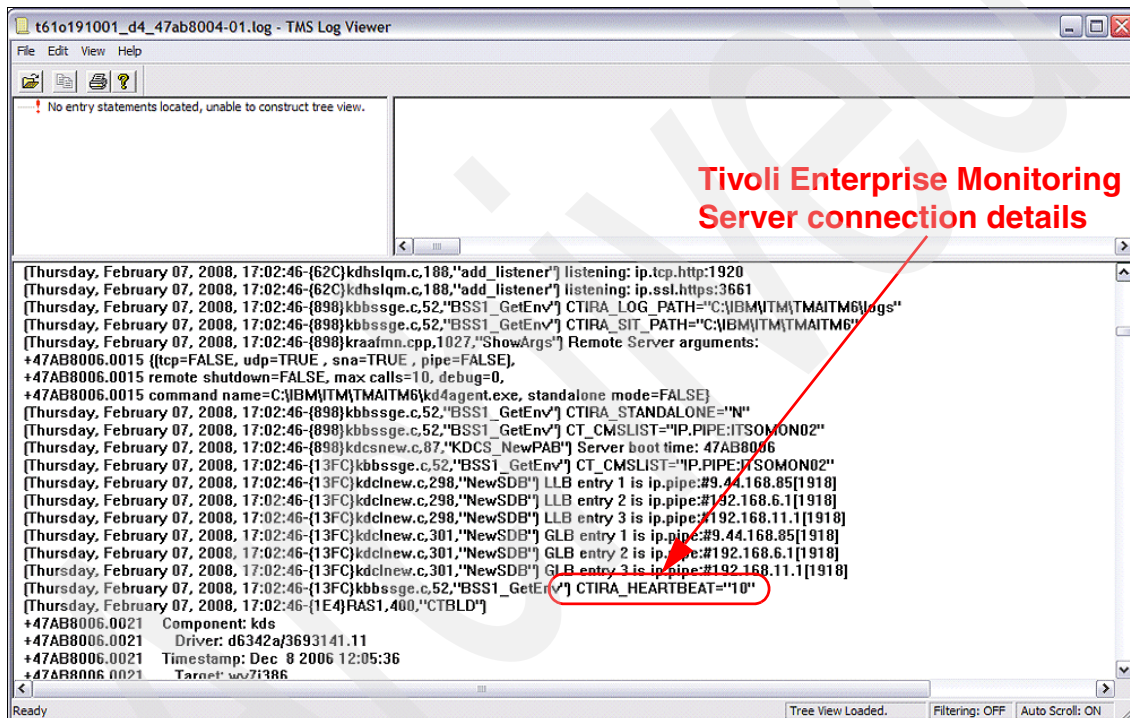


Figure 16-7 ITCAM for SOA Agent logs on Fabric server

**Note:** We recommend that you look at the problem determination section of the infocenter at the following Web address to get more details about the messages in the logs to resolve installation and configuration issues.

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamsoa.doc/kd4ugmst217.htm>

## 16.4.2 Enabling data collection from WebSphere Business Services Fabric

Before we enable the data collector on Fabric nodes, an important aspect to consider is the usage of the data collector in an environment where both SCA components and Web services exports that are on WebSphere Process Server are monitored. This case is especially true when SCA components communicate with services on other application servers. As of now, ITCAM for SOA supports the monitoring of the flows that are implemented using Web service bindings. Therefore we recommend that you enable applications for both the SCA and IBM WebSphere Application Server environment types.

To enable the data collection for SCA components on Fabric server node:

Execute the **KD4ConfigDC.bat -enable -env <x> <WPS\_HOME>** command, as shown in Figure 16-8.



```
Command Prompt
C:\IBM\ITM\ITMAITM6\KD4\bin>KD4configDC.bat -enable -env 9 c:\IBM\WPS\
Configuration command = "C:\IBM\ITM\ITMAITM6\KD4\bin\configSCADC.BAT -enab
le c:\IBM\WPS\
1 file(s) copied.
Return code from configSCADC.bat = 0
C:\IBM\ITM\ITMAITM6\KD4\bin>KD4configDC.bat -enable -env 1 c:\IBM\WPS\
Configuration command = "C:\IBM\ITM\ITMAITM6\KD4\bin\configWASDC.BAT -enab
le c:\IBM\WPS\
1 file(s) copied.
Return code from configWASDC.bat = 0
C:\IBM\ITM\ITMAITM6\KD4\bin>
```

Figure 16-8 Enabling data collection for SCA and Web Services

Note that executing this command copies relevant jar files into the <WPS\_HOME>\lib\ext folder. Similarly, disabling the data collection deletes the respective files. In both cases, you must restart WebSphere Process Server for the data collector to start and stop monitoring. For the purposes of this chapter, we enable the above monitoring for the WebSphere Process Server test environment that is installed along with WebSphere Integration Developer v6.1. We can use the WebSphere Business Services Fabric Unit Test Environment (UTE) to run various tests to capture sample data for Tivoli Enterprise Monitoring Server.

For further information about how to enable and disable the data collector, refer to the following Web site:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamsoa.doc/kd4ugmst49.htm>

It is important to take a note of the following limitations in the data collector that ships with v6.1 for SCA components:

- ▶ The data collector in the SCA environment only supports monitoring, not filtering. You can define monitoring controls to limit which SCA services or operations are monitored, but any filter controls that you define to reject messages are ignored.
- ▶ Monitor control for the message content parameter (none, headers, body, and full) only supports the *none* value. All other values are ignored and treated as *none*.
- ▶ The message length for SCA interactions is always reported as *zero*.
- ▶ Information collected in the SCA environment might not render properly in the IBM Web Services Navigator topology, sequence diagram, and patterns views.
- ▶ Some asynchronous interactions between SCA components might result in requests or responses being reported multiple times or not at all, which is caused by *thread switching* in the SCA application server runtime environment and the limitations of the data collector to track flows across these thread switches. Monitoring asynchronous interactions is provided to give a more complete picture of the services that exist in the environment; however, do not expect the metrics that are collected for asynchronous interactions to be precise.

**Note:** The limitations are current at the time we wrote this book. We recommend that you keep track of the latest additions to the product, and monitor the limitations for SCA support from the infocenter at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamsoa.doc/kd4ugmst95.htm>

We configured the ITSOBank enterprise monitoring infrastructure based on IBM Tivoli Enterprise Monitoring Server. As the data is captured, you can use Tivoli Enterprise Portal to analyze various views of the services invocation by service consumers.

Enterprises have to design sophisticated test strategies to understand the performance of the hosting environment and set proper filters to gather the necessary data for management and monitoring purposes. A key consideration here is to enable data capturing in a non intrusive way so that the operational environment performance is not negatively effected.

## 16.5 Data collection and monitoring for the ITSOBank Loan Process composite business application

Data collection and monitoring takes memory, storage, and CPU cycles; therefore, it is important to plan the time and duration during which data is collected. The intention behind which the data is being collected plays a vital role in planning when and how the data needs to be collected.

Data collection can happen in the following cases, but not limited to:

- ▶ Finding the root cause of the problem after receiving a trouble ticket.
- ▶ Understanding the performance of specific services or composition of services.
- ▶ Collecting historical data for services inventory and metrics.
- ▶ Monitoring the message flow during a specific time of the business day.

Because we enabled the data collection for the production servers that run the ITSOBank Loan Process composite business application, when the tests are executed, the data is automatically collected and transmitted to the Tivoli Monitoring Enterprise Server configured on ITSOMON02.

To check that the communication between the hosting server and the Tivoli Enterprise Monitoring Server works, we can use the Unit Test Environment in the development environment, and run sample test runs, as shown in Figure 16-9 on page 473.

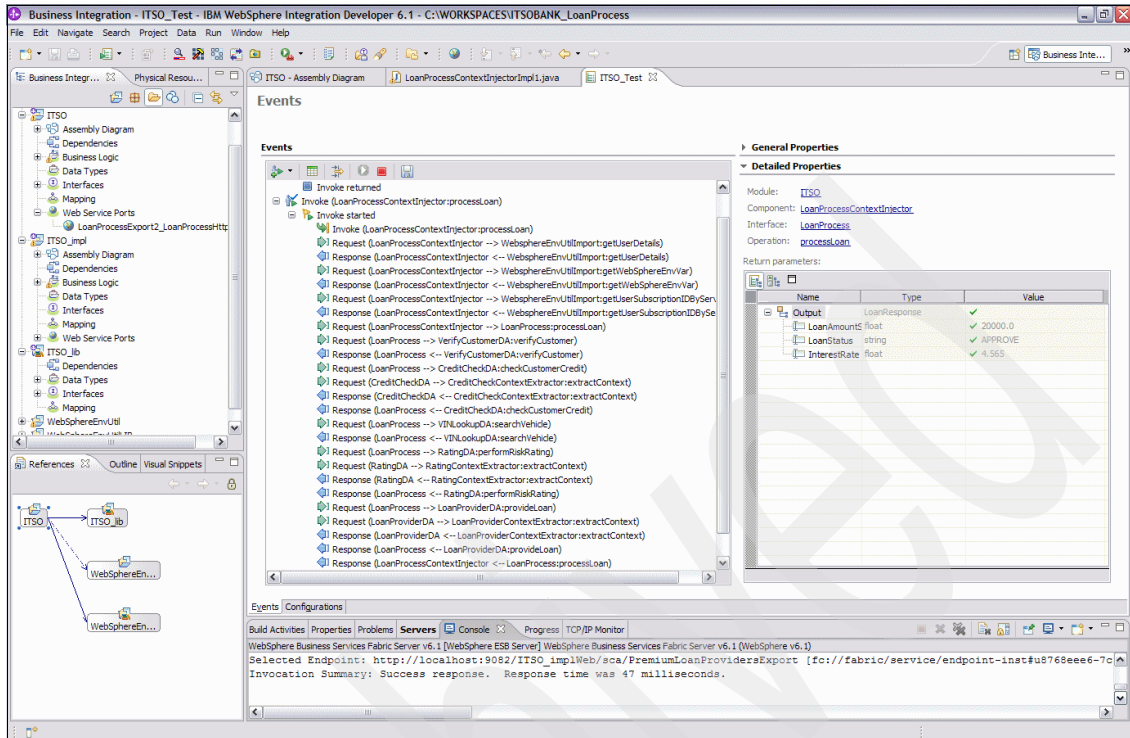


Figure 16-9 Running composite business application unit tests for sample monitoring data

After you successfully run this unit test, the ITCAM for SOA agent that runs on this server creates a metrics file, as shown in Figure 16-10. This log is named in the form: `KD4.<app_srv>.<cluster>.<cell>.<node>.<server>.metric.log`.

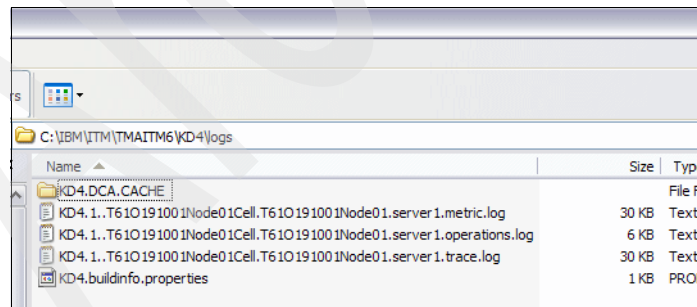


Figure 16-10 Metrics, operations, and trace log created by ITCAM for SOA Agent

Note that if the ITCAM for SOA agent successfully established contact with the Tivoli Enterprise Monitoring Server that is running on ITSOMON02, this file is immediately flushed out to the server.



Tivoli Enterprise Monitoring Server on ITSOMON02 collates the test data that is collected from the hosting server unit test environment. You can view the test data through various workspaces that are configured on Tivoli Enterprise Portal.

You can view Tivoli Enterprise Portal using a browser or through a desktop client by starting the respective application, as shown in Figure 16-11.

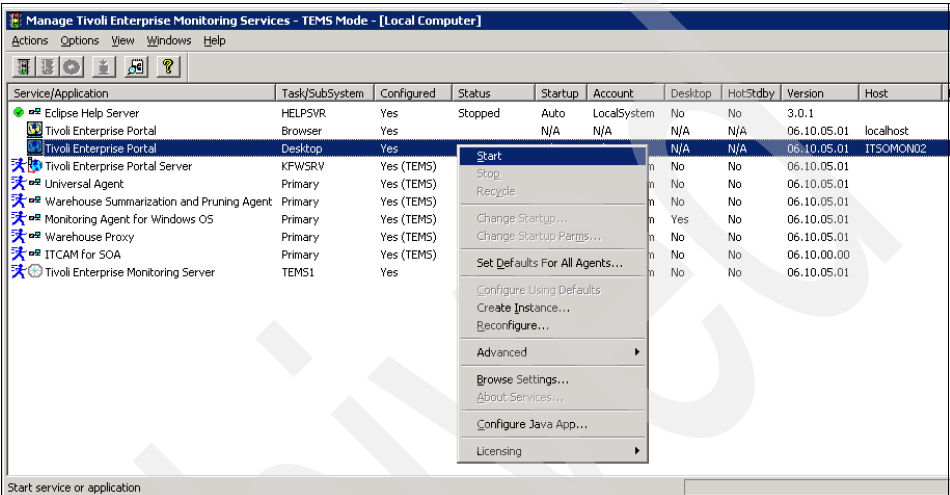


Figure 16-11 Starting the Tivoli Enterprise Portal desktop client

You can now verify that the test runs successfully collected the services invocation details by clicking **Services Management Agent** → **Services Management Agent Environment** → **<Node\_Name\_Server\_Name>** → **Performance Summary**, as shown in Figure 16-12 on page 475.



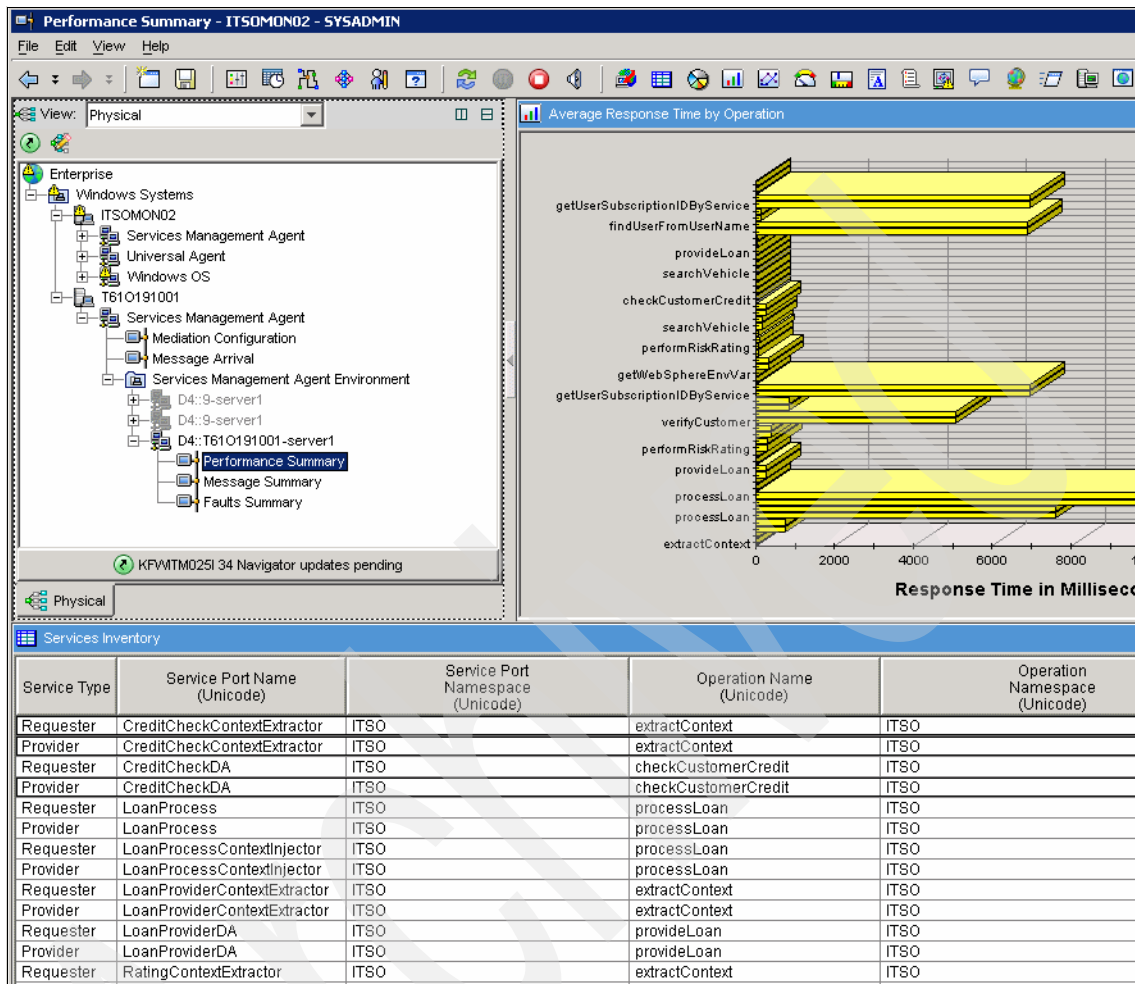


Figure 16-12 Tivoli Enterprise Portal work space for Performance Summary of Web services

You can observe that various operations are listed along with their average message round trip time.

There are multiple views that you can configure in ITCAM for SOA for viewing different types of data. Refer to the following Web site for more information about workspaces and configuring navigators:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamsoa.doc/kd4ugmst118.htm>

In a similar way, you can collect and monitor data for either performance testing the composite application or monitoring the production deployment during a specific time frame.

For the purposes of illustration in this chapter, we used Apache Bench to simulate simultaneous requests to the loan process Web service. To learn more about Apache Bench, refer to:

<http://httpd.apache.org/docs/2.2/programs/ab.html>

Apache Bench is a free tool that is shipped with Apache and IBM HTTP Servers. In the example shown in Figure 16-13 on page 477, Apache Bench simulates 10 concurrent users, putting 30 requests for the loan processing composite application.

```

C:\ITSOBank\LoanProcess_Performance>ab.exe -t 500 -n 30 -c 10 -p ITSOLoanProcess
_Test_Request.xml -k -H "SOAPAction: urn:processLoan" -I "text/xml; charset=UTF-
8" http://fabricserver:9082/ITSOWeb/sca/LoanProcessExport2
This is ApacheBench, Version 2.0.41-dev <$Revision: 1.121.2.12 $> apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright (c) 1998-2002 The Apache Software Foundation, http://www.apache.org/

Benchmarking fabricserver (be patient).....done

Server Software:      WebSphere
Server Hostname:      fabricserver
Server Port:          9082

Document Path:        /ITSOWeb/sca/LoanProcessExport2
Document Length:      531 bytes

Concurrency Level:    10
Time taken for tests:  19.93750 seconds
Complete requests:    30
Failed requests:      0
Write errors:         0
Non-2xx responses:    30
Keep-Alive requests:  0
Total transferred:    21990 bytes
Total POSTed:         40380
HTML transferred:     15930 bytes
Requests per second:  1.57 [#/sec] (mean)
Time per request:     6364.583 [ms] (mean)
Time per request:     636.458 [ms] (mean, across all concurrent requests)
Transfer rate:        1.10 [Kbytes/sec] received
                      2.07 kb/s sent
                      3.19 kb/s total

Connection Times (ms)
      min      mean[+/-sdl] median      max
Connect:    0      0      2.8      0      15
Processing: 1468  5941  2995.0    6203   19093
Waiting:    1453  5939  2996.5    6203   19093
Total:      1468  5941  2995.2    6203   19093

Percentage of the requests served within a certain time (ms)
 50%    6203
 66%    6906
 75%    7171
 80%    7187
 90%    7250
 95%    7250
 98%   19093
 99%   19093
100%   19093 <longest request>

C:\ITSOBank\LoanProcess_Performance>_

```

Figure 16-13 Apache Bench for load testing the ITSOBank Loan Process

Refer to the ITSOLoanProcess\_Test\_Request.xml that we provide in the additional materials to know more about how the SOAP request is constructed for this test. For details about how to obtain the additional materials that are supplied with this book, refer to Appendix A, “Additional material” on page 503.

As a result of this load testing, you can observe that Tivoli Enterprise Monitoring Server flagged a critical event, which points out that the hosting environment received more requests than expected at the same time. This can possibly also mean a security threat. Figure 16-14 on page 478 shows the Tivoli Enterprise Portal with the critical message flow alert.

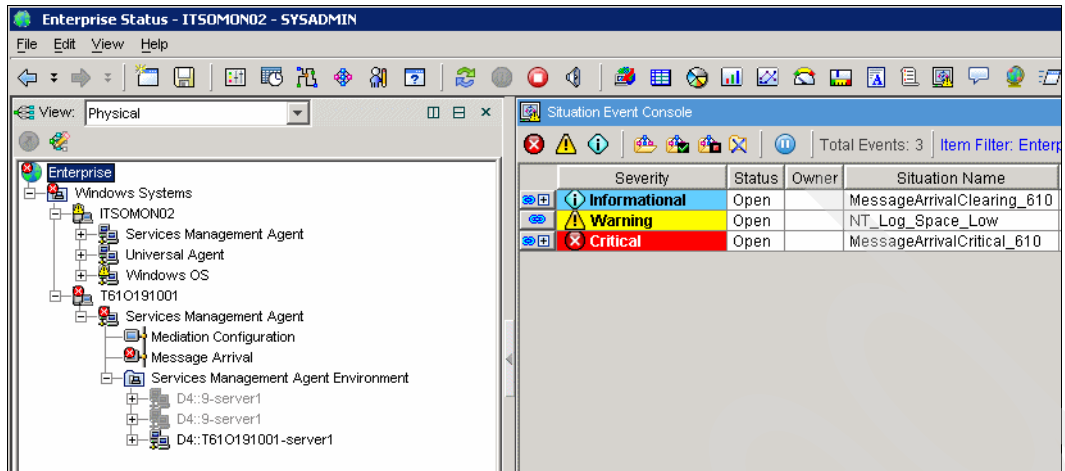


Figure 16-14 WebSphere Enterprise Portal alerting the increase in message flow

In advanced scenarios, this event can trigger various actions as configured in the Tivoli Enterprise Monitoring Server.

Different views that the Services Management Agent Environment provides, gives information about services operations, messages, faults, response time, and other details. You can enable historical data to store relevant monitoring data to analyze the performance of this composite application.

ITCAM for SOA monitors the overall performance-specific information of the hosting environment while the Performance Manager of the WebSphere Business Services Fabric provides business context transactions detail, such as how a business service was invoked, through which channel, and how an endpoint is performing over a given channel along with the endpoint-specific performance details for a specific transaction. Figure 16-15 on page 479 shows a sample transaction with endpoint-specific response times.

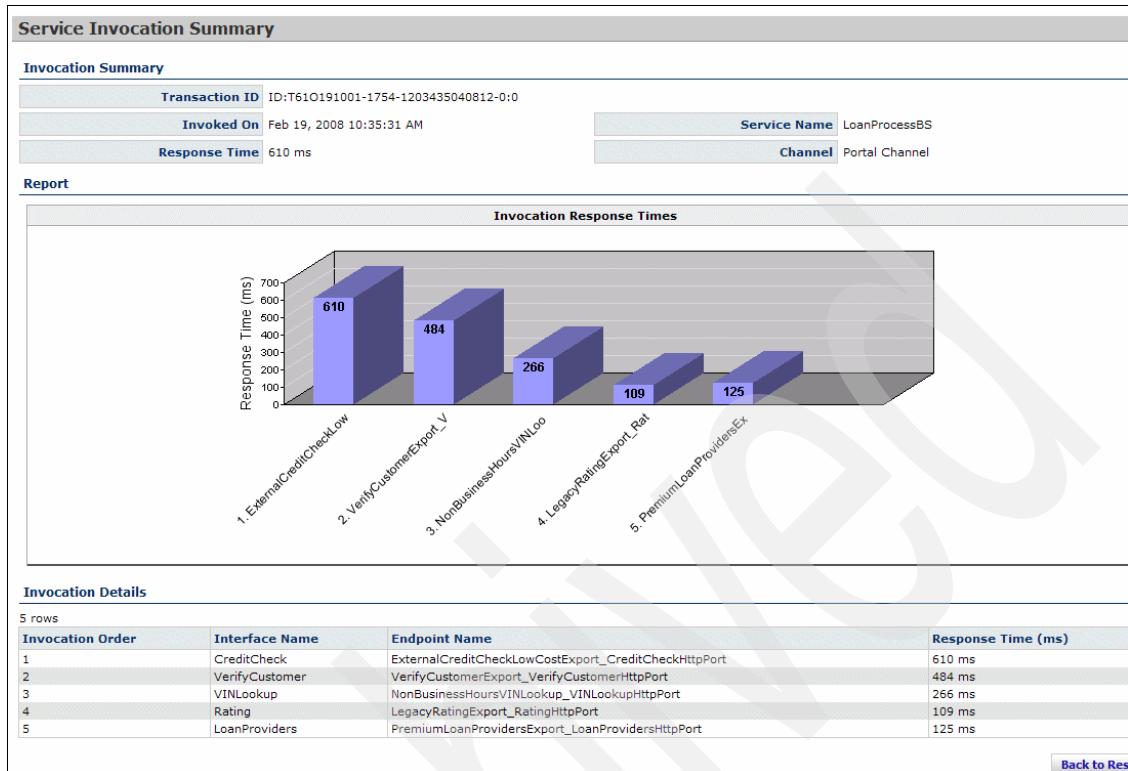


Figure 16-15 Performance Manager report showing transaction and endpoint specific details

ITCAM for SOA and Performance Manager built with WebSphere Business Services Fabric complement each other to obtain a holistic management view of the composite business application.

## 16.6 Conclusion

In conclusion, a careful design needs to be in place for enterprises to manage composite business application environments. You can monitor the Dynamic Assembler, because it is an SCA component, using an SCA data collector. Performance Manager reports and ITCAM for SOA reports are complementary to one another and can be used to get a holistic view of the operating environment.

In some of the advanced usage scenarios of ITCAM for SOA, you can configure Tivoli Enterprise Console® (TEC) to trigger alerts or corrective actions into the loan process implementation. You can configure the loan process composite application to consume these events to take appropriate actions, for instance, if

the node on which the endpoint is running is down, an event can be triggered from ITCAM to WebSphere Business Services Fabric, and if there are inactive endpoints that provide the same capability, you can make that endpoint active and create Fabric Business policies at runtime to ensure that the Dynamic Assembler picks up the activated endpoint.

## Integrating with Lightweight Directory Access Protocol

In this chapter, we discuss the integration of WebSphere Business Services Fabric (Fabric) with federated user registries using the IBM WebSphere Virtual Member Manager.

In particular, we look at how we can integrate a Lightweight Directory Access Protocol (LDAP) directory, such as IBM Tivoli Directory Server, with Fabric to configure various types of users, their associated roles, and business services.

## 17.1 Introduction

Different organizations have different ways of representing their organizational entities, such as people, login accounts, business units, business roles, domains, user groups, access controls, security roles, and so on. Different applications within an organization might also use different types of repositories with varying levels of security and functional capabilities.

A common way to store user identities is to use an LDAP directory. LDAP is based on the X.500 standard and provides simple and widely accepted protocol standard for directory access. Refer to the following Web site to obtain information about LDAP and its use:

[http://www.ldapman.org/articles/intro\\_to\\_ldap.html](http://www.ldapman.org/articles/intro_to_ldap.html)

IBM Tivoli Directory Server provides an on-demand identity infrastructure through compliance to LDAP.

You can effectively use IBM Tivoli Directory Server to store and retrieve:

- ▶ Users identification specific details (username, phone book, social security number, and so on)
- ▶ Organizational hierarchy
- ▶ External customer contact information
- ▶ Software application specific configuration details
- ▶ Infrastructure services information, such as e-mail aliases
- ▶ Public digital certificates, security keys, and so on.

For the purposes of this chapter, we assume that an organization-wide LDAP directory is configured using IBM Tivoli Directory Server to store hierarchical user identities along with other information. Different business applications can interface with this enterprise asset to retrieve user identities.

WebSphere Business Services Fabric depends on the Virtual Member Manager (VMM) of WebSphere Process Server to abstract the way users are represented and identified. VMM is a new functionality that is provided in the WebSphere Application Server V6.1 to:

- ▶ Support multiple pluggable repositories
- ▶ Provide a repository-independent programming interface
- ▶ Achieve a single view of multiple user repositories in a federated model



While federated repositories give a lot of flexibility in the way organizational entities are represented and integrated across multiple applications, there are some limitations that we must pay attention to:

- ▶ There can only be *one* user registry configured as a target for administrative purposes, for example, a user in a file-based repositories has to be configured as a “Primary Administrator” who is entitled to do the necessary user administration using the admin console.
- ▶ The user name must be unique across different repositories, for example, a user cannot have the same user ID in different LDAP registries, even under different organizational structures.
- ▶ All federated repositories (or their backup repositories) should always be up and available. If any one of the repositories are down, WebSphere Application Server will not authenticate the administrative user.

There are other limitations that you can review at:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rwim\\_limitations.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rwim_limitations.html)

It is important to take note of the limitations while you design the integration of any user registry with WebSphere Business Services Fabric.

In the following sections, we discuss how you can integrate IBM Tivoli Directory Server with WebSphere Business Services Fabric to manage various types of users in an organization. We discuss this topic by considering the example of the ITSOBank that has a single instance of the LDAP user registry, as shown in Figure 17-1 on page 484.

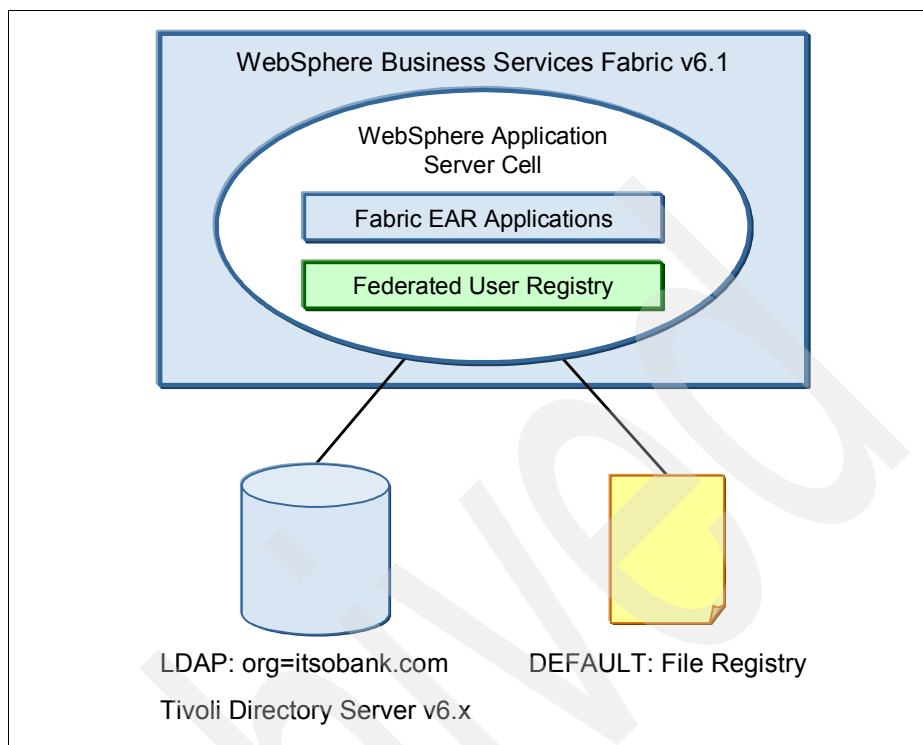


Figure 17-1 Federated user registry for ITSOBank

## 17.2 Integrating Fabric with IBM Tivoli Directory Server

The integration of WebSphere Business Services Fabric with a federated user repository is taken care of by the Virtual Member Manager, as we explained earlier. In this section, let us assume that the WebSphere Process Server on which Fabric is installed is previously configured with the default security settings with a primary administrator. When you configure default security, WebSphere Process Server configures a file-based repository as the user repository. A primary administrator user ID is responsible for managing the WebSphere Process Server environment. We recommend that you maintain this primary administrator ID as a super administrator for granting other administrative rights for users from the LDAP registry.

The user identities in the ITSOBank are logically classified into different groups, as shown in the Figure 17-2 on page 485.

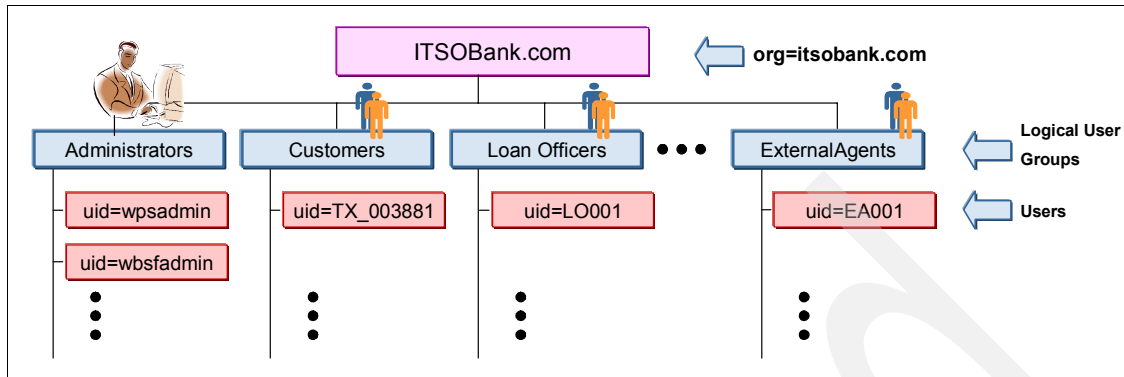


Figure 17-2 LDAP tree structure for ITSO Bank

Refer to the LDAP Data Interchange Format (LDIF) file `ITSOBank_LDAP_TREE.ldif` in the additional materials for more information about how the users and groups are setup for our scenario. To obtain the additional materials supplied with this book, refer to Appendix A, “Additional material” on page 503.

### 17.2.1 Foundation Pack administrator configuration

A super or primary administrator who is responsible for the IT operational infrastructure would configure the WebSphere Business Services Fabric on a new or existing WebSphere Process Server profile. Let us assume that at the time of installation, this primary administrator uses his/her user ID for which the WebSphere Process Server profile is configured. The following steps explain how a primary administrator configures the Fabric console access:

1. Figure 17-3 on page 486 shows an installation window, which at the time of installation of WebSphere Business Services Fabric Foundation Pack, accepts the user ID and password for the WebSphere Process Server profile.

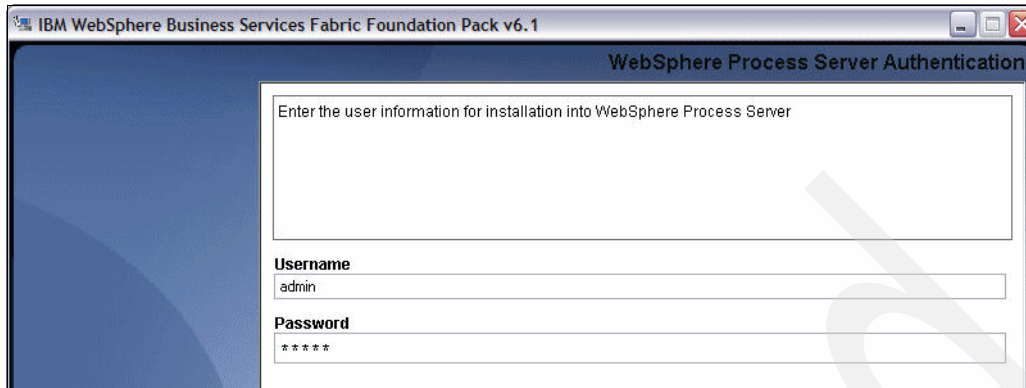


Figure 17-3 WebSphere Process Server authentication during Foundation Pack installation

2. As we discussed in the previous chapters, the primary administrator holds the authority and responsibility to assign himself as an Administrator in the Fabric\_Tools EAR application, which is done by mapping the primary administrator ID to the Administrator's role, as shown in Figure 17-4. Alternately, you can also create a Group and make the primary administrator the part of that Group and map the entire Group, as we discussed in Accessing the Fabric Tools console in 6.6.1, "Accessing the Fabric Tools Console" on page 112.

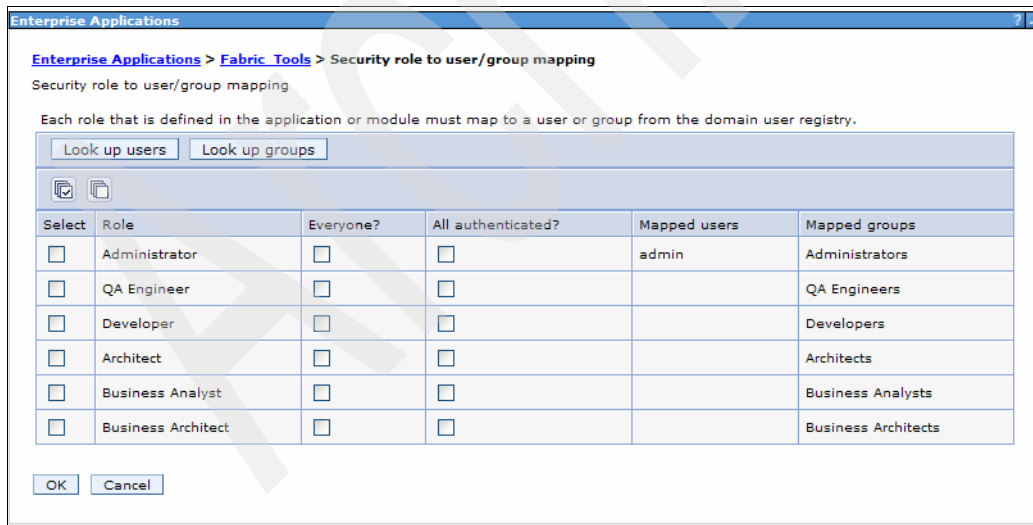


Figure 17-4 Primary Administrator user ID mapped as Fabric administrator

3. Login to the Fabric console application, and assign all administrative services from the main page to one administrative ID, as shown in Figure 17-5.



Figure 17-5 Primary Administrator Service Portfolio after first login

The first login after the installation of the foundation pack is key because this user identity, along with the associated realm, is used to map the primary user of the Fabric tools. Only this primary administrator user ID is given the authority to manage subscriptions to other administrative service portfolio to other users who need administrative access.

## 17.2.2 Configuring the federated user registry

The federated user registry gives you multiple ways to represent different user repositories through a single model. ITSOBank's LDAP registry is added to the existing default file-based user repository.

**Note:** Take care to make sure that this LDAP server is always up and available for the WebSphere Process Server and that there are no duplicate user IDs in the LDAP and the file-based default registry.

Use the following steps to configure the WebSphere Process Server for federated user registry in Virtual Member Manager:

1. Login to the WebSphere Process Server Administration Console using the primary administrator's user ID and password.

2. Click **Security** → **Secure administration, applications and infrastructure** → **Configure**, as shown in Figure 17-6.

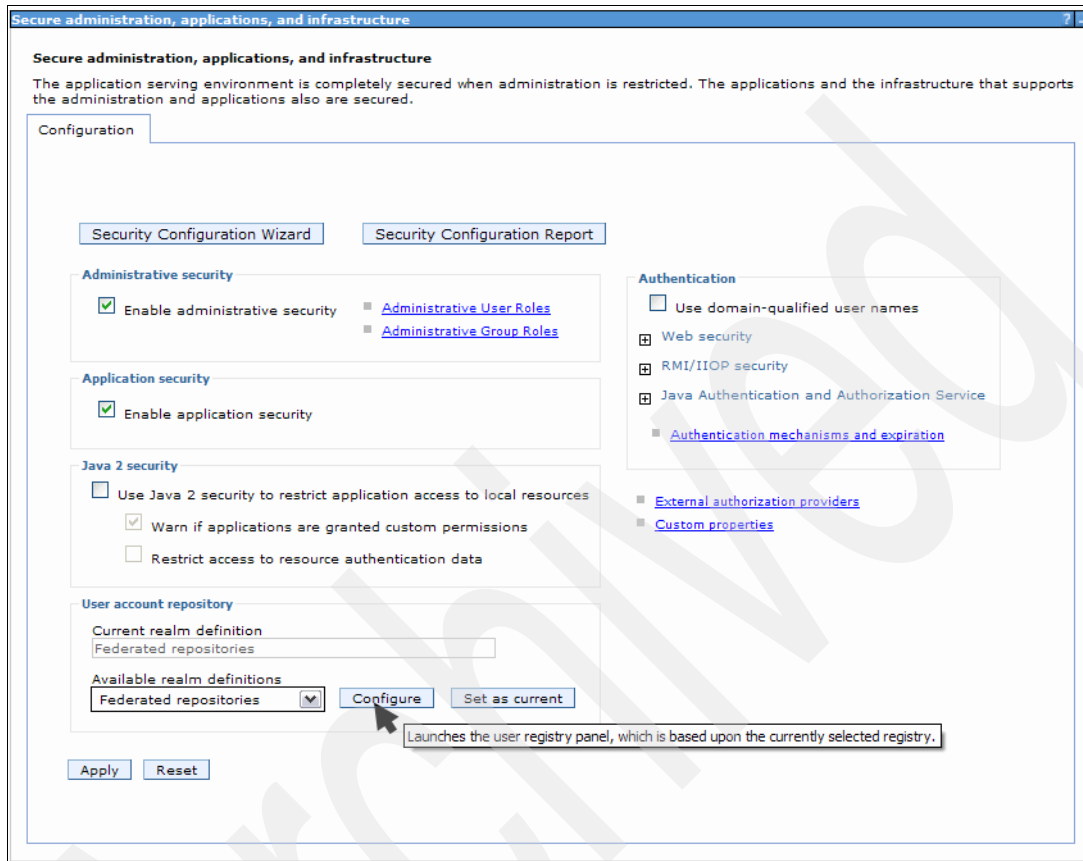


Figure 17-6 Configuring Federated user repositories

3. Click **Manage Repositories** → **Add** button to add a new repository.
4. This opens up a new Configuration page, where you specify the following:
  - a. In the Repository Identifier field, enter ITS0BankLDAP, as shown in Figure 17-7 on page 489.
  - b. From the Directory Type pull-down list, select Tivoli Directory Server Version 6.
  - c. Enter the host name and port address for the primary LDAP server. You can optionally configure the backup LDAP servers for this configuration.
  - d. Enter the username and password for binding with the LDAP directory server in the Security configuration.

Secure administration, applications, and infrastructure

[Secure administration, applications, and infrastructure](#) > [Federated repositories](#) > [Manage repositories](#) > **New**

Specifies the configuration for secure access to a Lightweight Directory Access Protocol (LDAP) repository with optional failover.

Configuration

---

**General Properties**

\* Repository identifier  
ITSOBankLDAP

**LDAP server**

\* Directory type  
IBM Tivoli Directory Server Version 6

\* Primary host name  
tamvm Port  
389

Failover server used when primary is not available:

Delete

Select	Failover host name	Port
<input type="checkbox"/>	tdsbackup	389

Add

Support referrals to other LDAP servers  
ignore

**Security**

Bind distinguished name  
cn=root

Bind password  
\*\*\*\*\*

Login properties  
uid

Certificate mapping  
EXACT\_DN

Certificate filter

☐ Require SSL communications

☒ Centrally managed

[Manage endpoint security configurations](#)

fabricserver.itsobank.com:9043

Figure 17-7 Configuring IBM Tivoli Directory Server V6 in the Federated Repository

- Click **Apply** and **Save**. The new repository is now added to the list of repositories, as shown in Figure 17-8.

Secure administration, applications, and infrastructure

[Secure administration, applications, and infrastructure](#) > [Federated repositories](#) > [Manage repositories](#)

Repositories that are configured in the system are listed in the following table. You can add or delete external repositories.

Preferences

Add Delete

Select	Repository identifier	Repository type
<input type="checkbox"/>	ITSOBankLDAP	LDAP:IDSS
<input type="checkbox"/>	InternalFileRepository	File

Total 2

Figure 17-8 List of Federated Repositories in WebSphere Process Server

6. Click **Federated repositories**. Select the **ITSOBankLDAP** repository identifier, and click **Add Base entry to Realm**.
7. A window opens, as shown in Figure 17-9, where you specify:
  - a. In the Distinguished name of a base entry that uniquely identifies this set of entries in the realm field, type `o=itsobank`.
  - b. In the Distinguished name of a base entry in this repository, type `dc=itsobank,dc=com`.

Secure administration, applications, and infrastructure

Secure administration, applications, and infrastructure > Federated repositories > Repository reference

Specifies a set of identity entries in a repository that are referenced by a base entry into the directory information tree. If multiple repositories are included in the same realm, it might be necessary to define an additional distinguished name that uniquely identifies this set of entries within the realm.

Configuration

General Properties

\* Repository  
ITSOBankLDAP

\* Distinguished name of a base entry that uniquely identifies this set of entries in the realm  
o=itsobank

Distinguished name of a base entry in this repository  
dc=itsobank,dc=com

Figure 17-9 Adding base entry into the realm of federated repository

8. Click **OK**. This new realm is added to the federated repository, as shown in Figure 9 on page 491.



[Secure administration, applications, and infrastructure](#) > **Federated repositories**

By federating repositories, identities stored in multiple repositories can be managed in a single, virtual realm. The realm can consist of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in repository and one or more external repositories.

Configuration

---

**General Properties**

\* Realm name  
defaultWIMFileBasedRealm

\* Primary administrative user name  
admin

**Server user identity**

☒ Automatically generated server identity

☐ Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

☒ Ignore case for authorization

Repositories in the realm:

Select	Base entry	Repository identifier	Repository type
<input type="checkbox"/>	o=defaultWIMFileBasedRealm	InternalFileRepository	File
<input type="checkbox"/>	<a href="#">o=itsobank</a>	<a href="#">ITSOBankLDAP</a>	LDAP:SECUREWAY

Figure 17-10 Federated repository configuration with additional realms

9. Save the configurations and restart the server.
10. You can now verify that the WebSphere Virtual Member Manager can search for users in both repositories:
  - a. Click **Users and Groups** → **Manage Users** in the integrated solutions console.
  - b. Enter the search criteria, and click **Search**. The users from the LDAP registry are also being displayed, as shown in Figure 17-11 on page 492.

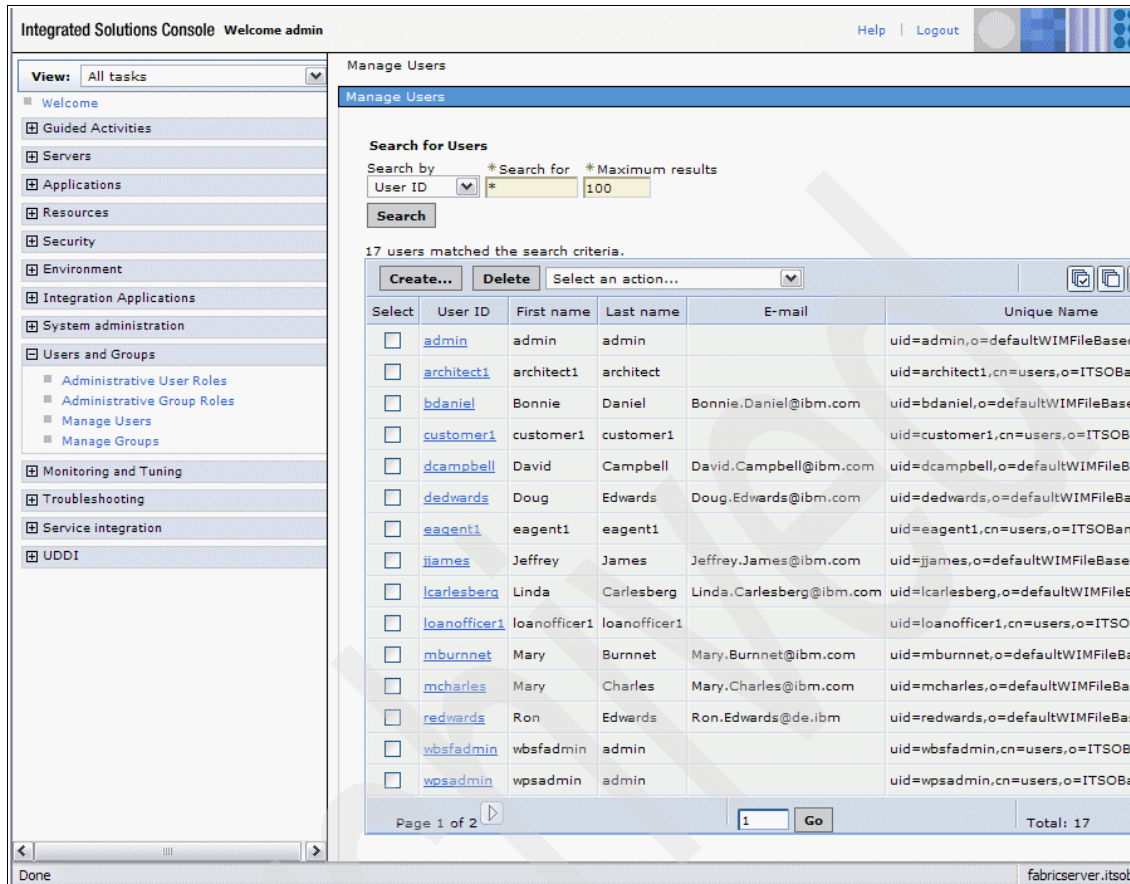


Figure 17-11 Searching for users in the federated user registry

Because WebSphere Business Services Fabric V6.1 primarily depends on the Virtual Member Manager to fetch user identity information, we can now search these users in the Fabric console to subscribe users to different business services.

### 17.2.3 Assigning a new Fabric administrator

Because the user identities are managed centrally by a single enterprise LDAP, you must grant the WebSphere Business Services Fabric administrative privileges to users from the LDAP. This administrator might be different from the primary administrator or super administrator of the operational IT infrastructure.

In this case, you must configure the new user from the LDAP so that this new user ID is part of a specific organization and has all of the necessary business services subscribed.

ITSOBank has an administrative user ID called *wbsfadmin* that is managed either by a single or a group of administrators. To make this user the administrator for the ITSOBank organization:

1. Login to the WebSphere Process Server Integrated Solutions Console as a primary administrator.
2. Click **Enterprise Applications** → **Fabric\_Tools** → **Security role to user/group mapping**.
3. Select the **Administrator** check box, and click **Look up Users**.
4. Enter the search criteria for selecting users, and click **Search**. The users from all of the federated repositories are visible, as shown in Figure 17-12.

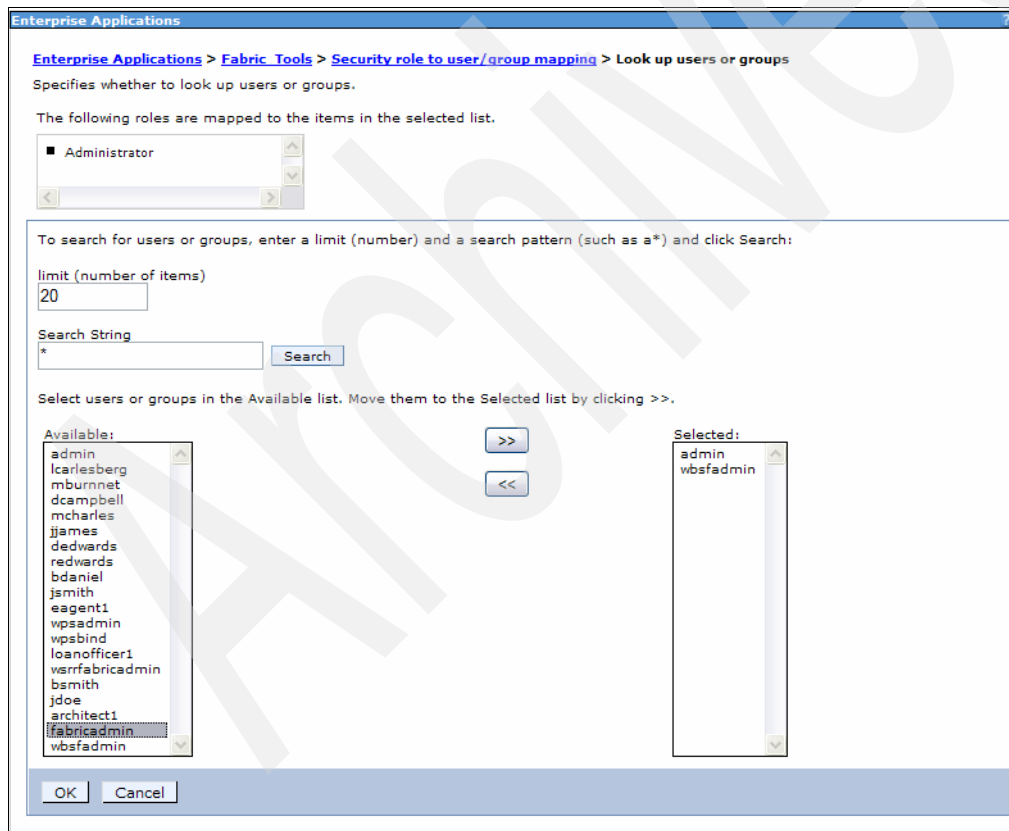


Figure 17-12 Assigning additional administrators

5. Select the **fabricadmin** user ID, and add it to the selected list by clicking the >> button. Click **OK**.
6. Login to the Fabric console as a primary administrator. Click **Subscriber Manager** → **Manage Subscribers**.
7. Select **Users** from the pull-down list for search criteria, and enter the last name of the user as admin, who will be added to the ITSO Bank.

**Note:** The functionality of a wild card search is not enabled for user search at the time this book is written. For provisioning users to different organizations and business services, the administrator should know the last name.

8. Click the identified user, and select the **Organizations** tab for the user. Click **Associate Organizations**, as shown in Figure 17-13.

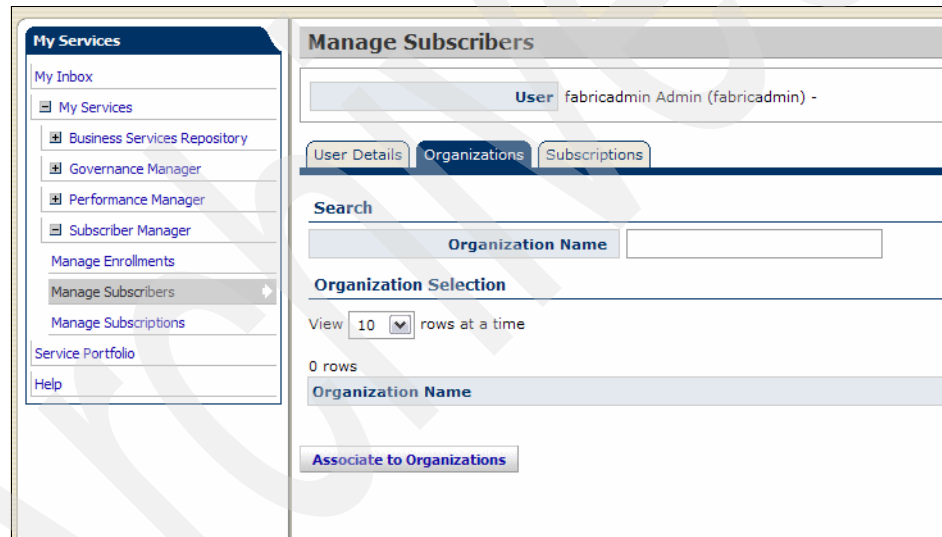


Figure 17-13 Associating a new user to an organization

9. Select the **ITSOBank and System Organization** for this administrative user to enable relevant subscriptions to business services for administration, as shown in Figure 17-14 on page 495.

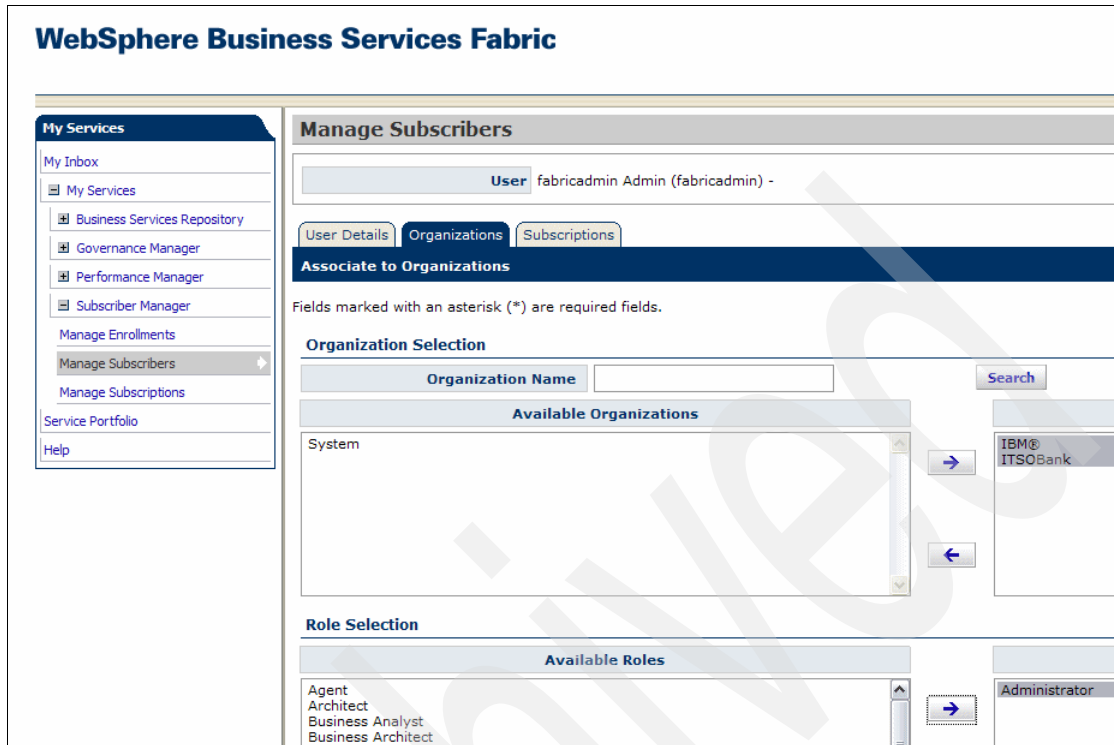


Figure 17-14 Adding the fabricadmin user to ITSOBank and System Organizations

10. Select the **Subscriptions** tab, and select the System Organization, which will display a list of services that the Fabric system offers. Select all of the services that will enable this user to act as a system administrator, as shown in Figure on page 496.

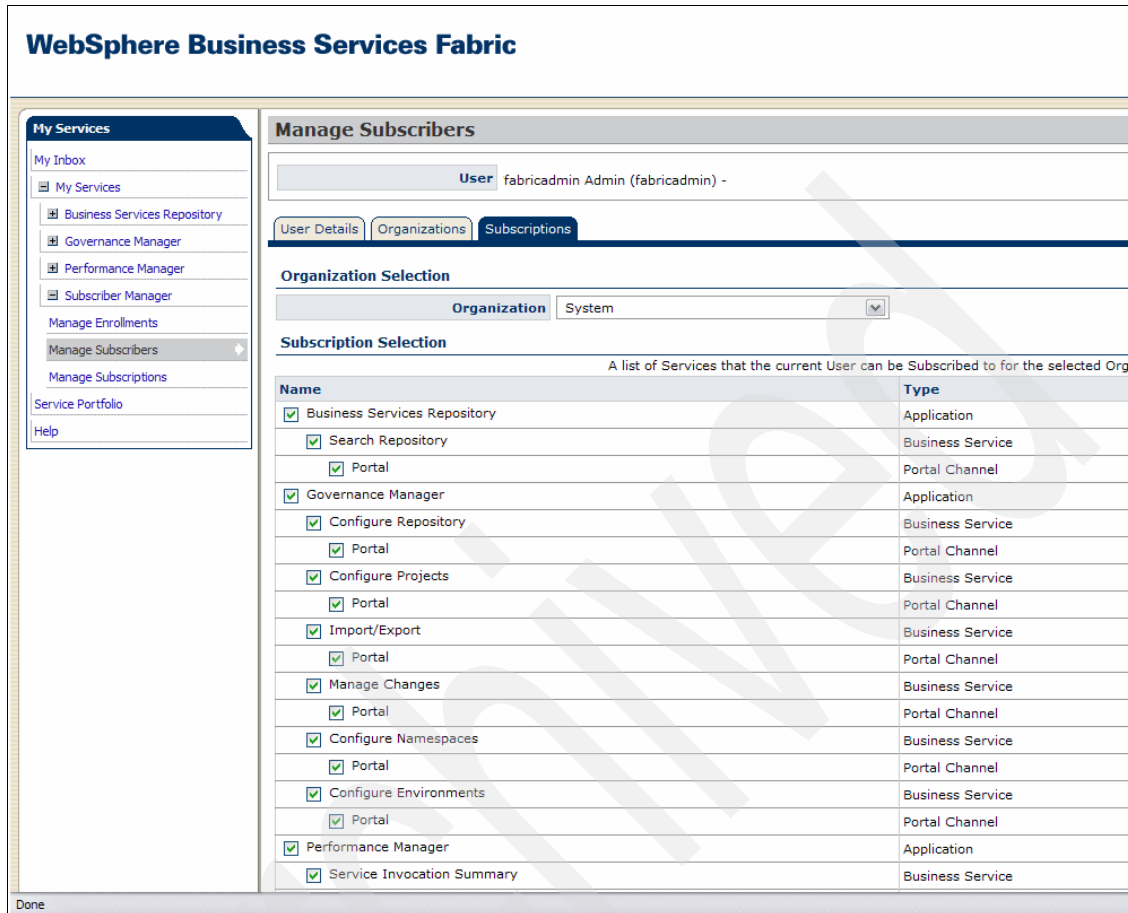


Figure 17-15 Subscribing system services to the new administrator

The new administrator (fabricadmin) can now log into the Fabric console and ensure that all of the Fabric system services are subscribed, as shown Figure 17-16 on page 497.



Figure 17-16 New administrator's subscribed services in the Fabric console

In a similar fashion, you can assign different users and associated roles, such as Customers, Employee's, Architects, and Developers, to the ITSOBank organization along with their entitled business services.

The developers and architects can use their LDAP user ID and password from the WebSphere Integration Developer Tool Pack (Composition Studio) for accessing their specific system-level services. Figure 17-17 on page 498 shows the preferences in the Composition Studio.

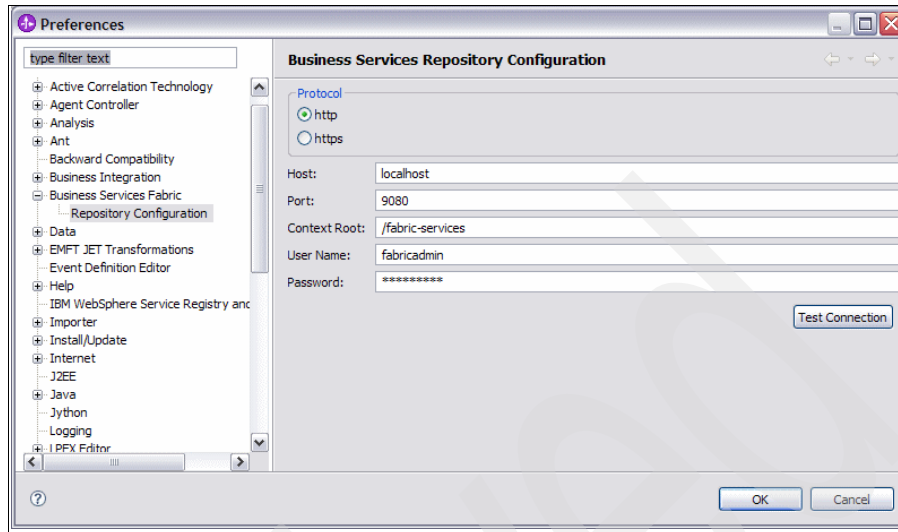


Figure 17-17 LDAP User configuration in the preferences

For the scenario considered in this book for ITSOBank, Table 17-1 lists the roles and user names.

Table 17-1 Sample users configured in the ITSOBank's LDAP directory.

User name	User ID	Group name	Description
Fabric Admin	fabricadmin	FabricAdminGroup	Fabric Administrator within ITSOBank.
WSRRFabric Admin	wsrrfabricadmin	WSRRFabricAdminGroup	Administrator to manage WSRR subscriptions with Fabric.
Joe Smith	jsmith	Customers	Sample customer ID for accessing ITSOBank's loan process.
John Doe	jdoe	LoanOfficers	Sample loan officer ID for accessing ITSOBank's loan process.
Jerry Sander	jsander	Developers	ITSOBank employee and an integration developer.
Bob Smith	bsmith	Developers	ITSOBank employee and an integration developer



In previous chapters, we assumed that the Fabric administrators and developers configured their unit test environment and hosting environment with this federated repository.

## 17.3 Conclusion

In this chapter, we introduced some of the basic concepts of how WebSphere Business Services Fabric uses the federated user repositories that the Virtual Member Manager of the underlying WebSphere Application Server provides. Monitor the various factors that affect the user identity management while integrating WebSphere Business Services Fabric with a federated user repositories. We further looked at how a group of administrators can be assigned for Fabric system management. Fabric uses the information that VMM provides to associate users to organizations and to various business services.





# Part 4

## Appendixes



## Additional material

In this book, we refer to additional material that you can download from the Internet.

### Locating the Web material

The Web material that is associated with this 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/SG247614>

Alternatively, you can go to the IBM Redbooks Web site at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select **Additional materials**, and open the directory that corresponds with the IBM Redbooks form number, SG247614.

### How to use the Web material

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material compressed file into this folder.



# Abbreviations and acronyms

<b>API</b>	application programming interfaces	<b>PME</b>	Programming Model Extension
<b>B2B</b>	business-to-business	<b>POJO</b>	Plain Old Java Object
<b>BOM</b>	Business Object Model	<b>RAS</b>	Reusable Asset Specification
<b>BPM</b>	Business Process Management	<b>RDBMS</b>	relational database management system
<b>BSR</b>	Business Service Repository	<b>RDF</b>	Resources Definition Framework
<b>CBA</b>	composite business application	<b>RHEL</b>	Red Hat Enterprise Edition
<b>DA</b>	Dynamic Assembler	<b>SCA</b>	Service Component Architecture
<b>EAR</b>	exported enterprise archives	<b>SDK</b>	Software Development Kit
<b>ERP</b>	Enterprise Resource Planning	<b>SDO</b>	Service Data Object
<b>ESB</b>	Enterprise Service Bus	<b>SLA</b>	Service Level Agreement
<b>FCA</b>	Fabric Content Archive	<b>SOA</b>	Service Oriented Architecture
<b>FCP</b>	Fabric Content Pack	<b>SOMA</b>	Service Oriented Modeling and Architecture
<b>FMT</b>	Fabric Modeling Tool	<b>UDDI</b>	Universal Discovery Description and Integration
<b>GBS</b>	Global Business Services	<b>UI</b>	user interfaces
<b>IBM</b>	International Business Machines Corporation	<b>UML</b>	Unified Modeling Language
<b>IDM</b>	Interface Design Model	<b>URI</b>	Uniform Resource Identifier
<b>ITSO</b>	International Technical Support Organization	<b>UTE</b>	Unit Test Environment
<b>IVR</b>	Interactive Voice Response	<b>VIN</b>	Vehicle Identification Number
<b>KPIs</b>	Key Performance Indicators	<b>VMM</b>	Virtual Member Manager
<b>LDAP</b>	Lightweight Directory Access Protocol	<b>WSDL</b>	Web Services Description Language
<b>LDIF</b>	LDAP Data Interchange Format	<b>WSRR</b>	WebSphere Service Registry and Repository
<b>LHEL</b>	Linux SUSE Linux Enterprise Server		
<b>LOB</b>	line of business		
<b>LPS</b>	Loan Processing System		
<b>NFR</b>	Non Functional Requirements		
<b>OWL</b>	Web Ontology Language		





# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 507. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151
- ▶ *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240
- ▶ *WebSphere Service Registry and Repository Handbook*, SG24-7386
- ▶ *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413
- ▶ *Implementing Technology to Support SOA Governance and Management*, SG24-7538
- ▶ *Best Practices for SOA Management*, REDP-4233
- ▶ *Case Study: SOA Governance Scenario*, REDP-4384

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](https://ibm.com/support)

IBM Global Services

[ibm.com/services](https://ibm.com/services)

# Index

## Numerics

20022 57

## A

Access protection 148  
ACORD 57, 374  
Acquisitions 8  
Adherence to standards 144  
Adoption of industry and open standards 151  
Agile solutions 16  
Apache Bench 476  
Application suites 147  
Architectural decisions document 144  
Architectural visibility 50  
Assemble 146  
    Access protection 148  
    Application suites 147  
    Constituent service components 147  
    Context specifications 148  
    Integrity control 148  
    Localization constraints 148  
    Ontology extensions 147  
    Packaging 148  
    Resource dependency 148  
    Simulate policies 148  
    Wrapper module 147  
Assembly event listener 75  
Assertions 142, 157  
Assisted provisioning 64  
Atomic Web services 376  
Auto provisioning 39  
Automated service entitlement 16

## B

Banking Payments Content Pack 379  
    Assets 383  
    Component model 385  
    Deployment view 388  
    Logical view 384  
BOM models 376  
Business Analysts 141  
Business Architects 141

Business design 138  
Business design parameters 138  
Business domain 141, 377  
Business innovation 12  
Business intent 34  
Business modeling 141  
    Assertions 142  
    Business Analysts 141  
    Business Architects 141  
    Business domain 141  
    Business policies 142  
    Business Process Models 142  
    Business requirements 141  
    Business Requirements Document 142  
    Business roles 141  
    Business Rules 142  
    Business Use Cases 142  
    Business use cases 141  
    Capture objectives 142  
    Channels 141  
    Goals 142  
    Industry process model 141  
    Key Performance Indicators 142  
    Linear process model 141  
    Non Functional Requirements 142  
    Points of variability 141  
    Process models 141  
    Solution Architects 142  
Business policies 34, 142, 270  
    Business intent 34  
    Business rules 34  
    Technical intent 34  
Business Process Management 8  
Business Process Models 142  
Business process simplification 40  
Business requirements 141  
Business Requirements Document 142  
Business roles 141  
Business Rules 142  
Business Service Domain 247  
Business service domain 163  
    Application suites 163  
    Applications 163  
    Business services 163

- Business Service perspective 68
- Business Service Templates 372
- Business services 16, 18
  - Anatomy 20
  - Business constraints 18
  - Business Function 20
  - Business Metadata 20
  - Canonical Data Model 20
  - Characteristics 19
    - Business policies 19
    - Designed at business level 19
    - Discrete business function 19
    - Industry models 19
    - Metadata 19
    - Multiple communication channels 19
  - Consolidation 20
  - Core business functions 18
  - Normalize 20
  - Service delivery 18
- Business Services Dynamic Assembler 53
- Business Services Fabric Modeling Tool 82
  - Assertions 82
  - Business capabilities 82
  - Channels 82
  - Process map 82
  - Roles 82
  - Transform industry standard schemas 82
  - UML notation 82
  - UML profiles 82
  - UML to OWL transformation 82
  - UML to UML transformation 82
  - XSD to UML import wizard 82
- Business Services Governance Manager 54
- Business services metadata model 43
- Business Services Performance Manager 55
- Business Services Repository 55, 69
  - Conflict detection 70
  - Correlation queries 70
  - Impact analysis 70
  - Namespaces 70
    - Enrollment 71
    - External 71
    - Instance 70
    - Namespace partitioning 70
    - Namespace type 70
    - Schema 71
    - Sponsored 71
  - Ontology 70
- Business services repository 32
- Classification 33
- Creation 33
- Modification 33
- ontologies 33
- Ownership 33
- Business Services Subscriber Manager. 55
- Business Services Viewer 55
- Business Use Cases 142
- Business use cases 141

## C

- Candidate Filter 207
- Candidate filter 74
- Capture objectives 142
- Case management 9
- Channels 141
- Classifications 410
- Clustering 408
- Code repositories 49
- Collaborative development 68
- Component models 143
- Composite business applications 16, 23
  - Agile solutions 16
  - Automated service entitlement 16
  - Business data model 24
  - Business object model 24
  - Business policies 24
  - Business processes 24
  - Business services 16
  - Business-level policies 16
  - Consumer channels 24
  - Content 16
  - Context 16
  - Contract 16
  - Incrementally transform 16
  - integration 23
  - Key attributes 32
    - Composable 32
    - Configurable 32
    - Describable 32
    - Dynamic 32
    - Governable 32
    - Interoperable 32
    - Subscribable 32
  - Metadata model 25
  - Metadata model extensions 25
  - Multichannel service delivery 16
  - Platform neutrality 23

- Process model 25
- Semantic glossaries 23
- Service agreements 16
- Service interface specifications 23
- Subscribers 16
- Composite policy 37
- Composition Studio 67
  - Applications 69
  - Applications suites 69
  - Business Service perspective 68
  - Business services 69
  - Collaborative development 68
  - Composite services 69
  - Correlations queries 69
  - Endpoints 69
  - Interfaces 69
  - Policies 69
  - Policy simulations 69
- Conditions based on content-based assertions 160
- Conditions based on model dimensions 160
- Conflict detection 70
- Constituent service components 147
- Constituents 10
- Consumption channels 44
- Content 16
- Context 16
- Context Extractor 206
- Context extractor 73
- Context information 156
- Context specification 156
- Context specifications 148
- Contextual fitness 144
- Continuous improvement 12, 151
- Contract 16
- Correlation queries 70
- Correlations queries 69
- Cross-enterprise business processes 16
- Custom applications 375
- Custom schema 375

**D**

- Data Architects 144
- Data Collector 463
- Data models 143
- Decision rights 151
- Declarative knowledge 43
- Deploy 148
  - Hosting environment 148
- Integration infrastructure 148
  - Subscription Manager 149
- De-provisioning 39
- Disparate applications 370
- Dynamic Assembler 48, 71
  - Candidates 71
  - Configuration 76
    - Enable Result Caching 76
    - Enable Verbose Logging 77
    - Fire Invocation Events 77
  - Extensions 73
    - Assembly event listener 75
    - Candidate filter 74
    - Context extractor 73
    - Endpoint filter 74
    - Extension execution 75
    - Policy guard 73
    - Response listener 74
  - Ranked 71
  - SCA 72
  - Sequence diagram 71
  - Tiered 71
- Dynamic Assembler component 201
- Dynamic assembly 26
  - Content 28
  - Context 28
  - Contract 28
- Dynamic business-service personalization 327
- Dynamic enablers 327

**E**

- Enable Result Caching 76, 204
- Enable Verbose Logging 77
- Enable Verbose logging 205
- End Date 77
- Endpoint assertion marked as required 158
- Endpoint Filter 207
- Endpoint filter 74
- Enforcement engine 52
- Enrollments 38–39, 64
- Enterprise Architects 144
- Enterprise ecosystem 375
- Enterprise Monitoring Agent 463
- Enterprise Resource Planning (ERP) 462
- Enterprise Service Bus (ESB) 406
- Error identification 370
- eTOM 374
- Event Listener 208

Extend content based assertions 157  
eXtensible Access Control Markup Language (XAC-ML) 414  
Extension execution 75  
External WSRR project type 425

## F

Fabric Administrative Console 53  
Fabric Business Service Model 162  
Fabric Content Archive (FCA) 155  
Fabric Modeling Tool 165  
Fabric Modeling Tool  
    Business Glossary Model 169  
    Fabric profile 168  
    Glossary profile 168  
    Glossary transformation 181  
    Ontology model 175  
    Pre-requisites 165  
        Extensibility Features 165  
        Fabric Core Stubs 165  
    RDF Schema 183  
    Web Ontology Language 183  
Fabric ontology core model 162  
Fabric ontology extension models 144  
Fabric policies 77  
    Context, content, contract 77  
    Key attributes 77  
        End Date 77  
        Priority 77  
        Start Date 77  
        Target 77  
    Natural ordering 79  
    Policy lock 80  
    Policy priority 80  
    Policy simulation 80  
    Scope 79  
Fabric Software Development Kit 53  
File systems 49  
Fire Invocation Events 77, 204

## G

Global competition 8  
Globalization 8  
Goals 142  
Govern 151  
    Adoption of industry and open standards 151  
    Continuous improvement 151  
    Decision rights 151

Maturity of service orientation 151  
Measures 151  
Policies 151  
Service auditing 151  
Service consumption 151  
Service development 151  
Service diagnostics 151  
Service discovery 151  
Service funding 151  
Service modeling 151  
Service monitoring 151  
Service ownership 151  
Service performance 151  
Service provisioning 151  
Service publishing 151  
Service registration 151  
Service security 151  
Service shared usage 151  
Service versioning 151  
Governance domain 163  
    Environments 163  
    Fabric projects 163  
    Namespaces 163  
Governance Manager 62  
    Configure environments and repository 62  
    Configure namespaces 62  
    Configure projects 62  
    Configure resource federation 62  
    Import and export 63  
    Manage changes 63  
Governance profile 419

## H

Hierarchical representation 373  
Hierarchical tree 411  
HL7 57, 370  
Holistic management view 479  
Holistic relationship 138  
Hosting environment 148

## I

IBM Business Services Foundation Pack 52  
IBM Business Services Tool Pack 52  
IBM Global Business Services 51  
IBM Information Framework model 381  
IBM Web Services Navigator 471  
Identified assertions 159  
IFW 380

- Impact analysis 70
- InboundResponse 74
- Independent Software Vendors 51
- Industry best practices 374
- Industry Business Glossary 372, 375
- Industry Business Object Model 372, 377
- Industry Business Service Templates 374
- Industry Capability Map 373
- Industry Common Services 372, 376
- Industry consolidation 8
- Industry Content Packs 369
  - Assets 373
  - Business Service Templates 372
  - Industry Business Glossary 372, 375
  - Industry Business Object Model 372, 377
  - Industry Business Service Templates 374
  - Industry Capability Map 373
  - Industry Common Services 372, 376
  - Industry Process Map 373
  - Industry Service Interfaces 372, 375
  - Knowledge Assets 377
    - Developer's Guide 377
    - How-to Guide 378
    - Installation Guide 377
    - Reference Architecture 377
    - Reference Implementation 378
  - Reference Architecture 371
  - SCA 372
- Industry content packs 56
- Industry Core Model 154
- Industry dictionaries 376
- Industry Glossary Model 154
- Industry participants 370
- Industry Process Map 373
- Industry process model 141
- Industry reference models 49
- Industry Service Interfaces 372, 375
- Industry Source Model 154
- Information models 143
- Integer tier 207
- Integration architecture 144
- Integration infrastructure 148
- Integrity control 148
- Integrity failures 63
- Interface Design Model (IDM) 381
- ISO 57
- ISO 20022 370, 380
- ISO 20022-Data Dictionary 380
- ITSOBank 86, 326

- Apply Loan 87
- Business challenges 95
  - Increasing business agility and flexibility 95
  - Innovate the business process to remain competitive 96
- IT challenges 96
  - Ability to monitor and manage the performance of business services 97
  - Business logic is embedded in multiple locations 96
  - Costly and time consuming business process redeployment 96
  - Impact of assimilating merger and acquisition systems and applications 98
  - Impact on the services utilization 98
  - Normalized service interfaces based on industry standards 97
  - Service entitlements 97
  - Service versioning 97
- Merger and acquisition 96
- Providing incremental business changes 96
- Rapid changes in business 95
- Simplifying the process changes and increasing business agility 95
- Calculate Risk Rating 87
- Credit Verification provider 88
- Customer 87
- Loan Officer 88
- Loan Processing System 86
- Loan Provider 88
- Perform Credit Check 87
- Perform Vehicle Identification Number (VIN) lookup 87
- Provide Loan 87
- Use case scenario 86
- Vehicle Identification Number (VIN) 86
- Vehicle loan processing system 86
- Verify Customer 87

## J

- J2EE application 408

## K

- KD4ConfigDC.bat 470
- Key Performance Indicators 142
- Key performance indicators 138

## **L**

Lifecycle for developing composite business applications 140  
Lightweight Directory Access Protocol (LDAP) 481  
Linear process model 141  
Load balancing 408  
Loan Processing System 86  
Localization constraints 148

## **M**

Macro-economic changes 8  
Manage 150  
    Monitor performance 150  
    Tivoli Composite Application Manager 150  
Market dynamics 327  
Mediation Primitives 463  
Mergers 8, 326  
Messaging models 376  
Messaging schemas 375  
Metadata governance 46  
Model 141  
Modeling 141  
Multichannel service delivery 16

## **N**

NACHA 381  
NACHA ACH 382  
Namespace 70  
Namespace partitioning 70  
Namespace type 70  
NGOSS SID 57  
Nmble 10  
Non Functional Requirements 142  
Non Functional Requirements (NFRs) 460  
Normalize 20

## **O**

OASIS standard 414  
Object-oriented applications 375  
Onthology  
    Web Ontology Language (OWL) 33  
Ontology 33  
    Behaviors 33  
    Constraints 33  
    Relationships 33  
    Resources Definition Framework (RDF) 33  
    Structure 33

Vocabulary 33  
Ontology extensions 147  
Operational models 143  
Organization 39  
OutboundResponse 74

## **P**

Packaged applications 143, 375  
Packaging 148  
Partner collaboration 9  
Payments Standards Evaluation Group 381  
Performance Manager 66  
    Endpoint service report 67  
    Invocation summary report 66  
    Service utilization report 67  
    Web service performance report 66  
Platform neutrality 23  
Points of variability 141, 156  
Policies 270  
Policy driven processes 11  
Policy Guard 206  
Policy guard 73  
Policy identification 160  
Policy lock 80  
Policy priority 80  
Policy simulation 80  
Policy simulations 69  
Policy Simulator 247  
Policy-driven relationships  
    Bottom-up 49  
    Top-down 49  
Portfolio management 44  
Priority 77  
Process models 49, 141  
Profile 155  
Property 155

## **R**

Ranked 71  
Rapid process change 10  
Rational Data Architect 377  
Rational Software Architect 82, 165  
Realignment of business 326  
Redbooks Web site 507  
    Contact us xvi  
Regulatory constraints 327  
Regulatory requirements 8  
Reject-always-policy 74



Resource dependency 148  
Resources Definition Framework (RDF) 33  
Response Listener 207  
Response listener 74  
REST 410  
Restructuring 326

## **S**

SAP NetWeaver 463  
SCDL files 409  
SDO 208  
Selection policy 37  
Self service provisioning 64  
Semantic glossaries 23  
SEPA 381  
Service agreements 16  
Service consumer domain 164  
    Organizations 164  
    Roles 164  
    Service levels 164  
    Users 164  
Service consumption 151  
Service Data Object 208  
Service development 151  
Service discovery 151  
Service execution behavior 17  
Service interface specifications 23  
Service Level Agreements (SLAs) 460  
Service Management Agent Environment 462  
Service model 143  
Service modeling 151  
Service Oriented Architecture (SOA) 8  
Service ownership 151  
Service provisioning 38  
Service publishing 151  
Service registration 151  
Service Technical Domain 247  
Service technical domain 164  
    Composite services 164  
    Dynamic assembly components 164  
    Endpoints 164  
    Interfaces 164  
Service-oriented development 405  
Services Management Agent 474  
Shifting consumer demands 8  
Siloed business functionality 11  
Simulate policies 148  
SOA Foundation life cycle 138

SOA Foundation Reference Architecture 460  
SOA governance 404  
Social services 9  
Software development kit 57  
Solution Architects 142  
Solution architecture 144  
Solution Modeling 142  
    Architectural decisions document 144  
    Component models 143  
    Data Architects 144  
    Data models 143  
    Enterprise Architects 144  
    Fabric ontology extension models 144  
    Information models 143  
    Integration architecture 144  
    Operational models 143  
    Service model 143  
    Solution architecture 144  
    Use case model 142  
Special assertions 158  
Start Date 77  
Stereotype 154  
Subscriber Manager 64  
    Enrollments 64  
    Subscribers 64  
    Subscription IDs 66  
    Subscriptions 64  
Subscribers 16, 39, 64  
Subscription IDs 66  
Subscriptions 64  
Sub-vertical 377  
System Integrators 51

## **T**

TAM 374  
Target 77  
Taxonomy 375  
Taxonomy of role types 157  
Technical intent 34  
Third-party services 375  
Thread switching 471  
Tiered 71  
Tiers 207  
Tivoli Composite Application Manager for SOA (IT-CAM for SOA) 459  
    Apache Bench 476  
    Data Collector 463  
    Enabling data collection 470

- Enterprise Monitoring Agent 463
- Heterogeneous platform support 462
- Holistic management view 479
- IBM Web Services Navigator 471
- Infrastructure setup 465
- ITCAM for SOA agent 468
- KD4ConfigDC.bat 470
- Manage Enterprise Monitoring Services console 469
- Mediation Primitives 463
- Policy editor 463
- SAP NetWeaver 463
- Service Management Agent 462
- Services Management Agent 474
- Thread switching 471
- Tivoli Enterprise Console (TEC) 479
- Tivoli Enterprise Monitoring Server 467
- Web Services Navigator 463
- Tivoli Directory Server 482
  - Assigning a new Fabric administrator 492
  - Configuring the federated user registry 487
  - Integrating Fabric 484
- Tivoli Enterprise Console (TEC) 479
- Tivoli Enterprise Portal Server (TEPS) 462
- Transformation 154

## U

- UML profiles 82
- UML to OWL transformation 82
- UML to UML transformation 82
- UNIFI 381
- Universal Discovery Description and Integration (UDDI) 404
- Use case model 142
- User-defined taxonomy 410

## V

- Vehicle Identification Number (VIN) 86
- Virtual Member Manager (VMM) 482
- Visual Studio .Net 411
- Vocabulary 33

## W

- Web Ontology Language (OWL) 409
- Web Services Business Process Execution Language 194
- Web Services Navigator 463

- WebSphere Business Services Fabric installation
  - Derby database 103
  - Fabric Database 102
  - Installing Fabric on z/OS 132
  - Installing the Business Services Foundation Pack 105
  - Installing the Business Services Tool Pack 126
    - Verifying the Business Services Tool Pack installation 130
  - Known limitations 133
    - Clustering and failover 134
  - Pre-requisites 103
    - Business Services Fabric Modelling Tool 105
    - Databases 104
    - IBM Rational Software Architect 105
    - Installing the Business Services Foundation Pack
      - Installation wizard 105
    - LDAP servers 104
    - WebSphere Service Registry and Repository 104
  - Typical deployment topology for the Business Services Foundation Pack 100
  - Unit Test Environment 103
  - Verifying the Business Services Foundation Pack installation 111
    - Accessing the Fabric Tools Console 112
  - Troubleshooting your installation 121
    - Activation specification 125
    - Connection factory 124
    - Data sources 123
    - Dynamic Assembler and SCA libraries 126
    - FabricDB database 121
    - Mail session 125
    - SCA service integration bus 125
    - Topic 124
- WebSphere Integration Developer 17
- WebSphere Process Server 16
- WebSphere Service Registry and Repository (WS-RR) 403
  - Access control definitions 449
  - Accessing registry content 410
    - API 410
    - Web UI 411
  - Adding artifacts to a lifecycle 440
  - Adding classifications 437
  - Additional meta-data 410

- Business templates 410, 417
- Classifications 410
- Clustering 408
- Common taxonomy 418
- Configuration profile 415
- Data-centric services 418
- Defining WSRR permissions for Fabric federation 443
- Derived logical components 409
- Eclipse 411
- EJB Remote interface 410
- Enterprise Service Bus (ESB) 406
- eXtensible Access Control Markup Language (XACML) 414
- External WSRR project type 425
- Federation Settings 426
- Filtering federated content 442
- Finding artifacts in WSRR 450
  - Using the Eclipse plug-in 452
  - Using the WSRR Web UI application 450
- Governance profile 419
- Governance tab 441
- Hierarchical tree 411
- J2EE application 408
- Load balancing 408
- Load Documents 429
- Logical model 409, 421
- Manage Roles view 446
- Modifier 414
- Non-binary documents 411
- Notification 414
- Notifier 414
- OASIS standard 414
- Permission action 445
- Permission target 445
- Permissions 413
- Perspectives 411
- Physical documents 409
- Policies 422
- Principals 443
- Process-centric services 418
- Publishing an SCA integration module 433
- Publishing using the Eclipse plug-in 436
- Registry 404
- Relationships 410
- Repository 404
- REST 410
- Roles 443
- SCA integration module 433
- SCDL files 409
- Scenario 1 - Deploying Fabric first, WSRR next 417
- Scenario 2 - Deploying WSRR first, Fabric next 420
- Scenario 3- Deploying both at the same time 423
- Security 413
- Selective federation 446
- Service lifecycle 412
- Service-oriented development 405
- Share user registries 420
- SOA governance 404
- State machine 413
- State Machine editor 413
- Templates 409
- Test Connection 428
- Universal Discovery Description and Integration (UDDI) 404
- User-defined taxonomy 410
- Utility services 418
- Validation 414
- Versioning 421
- Visual Studio .Net 411
- Web Ontology Language (OWL) 409
- WSDL 409
- WS-Policy 422
- WSRR Federation 425
- XML Schema 409
- XPath statements 411
- WebSphere Services Registry and Repository 17
- Wrapper module 147
- WS-BPEL 194
- WSDL 409
- WS-Policy 422

**X**

- XACML 414
- XML Schema 409
- XPath statements 411
- XSD to UML import wizard 82
- XYZBank 326

**Z**

- z/OS 132





**Redbooks**

# Getting Started with IBM WebSphere Business Services Fabric V6.1

(1.0" spine)  
0.875" <-> 1.498"  
460 <-> 788 pages







# Getting Started with IBM WebSphere Business Services Fabric V6.1



**Discover the value of  
composite business  
applications**

**Model, assemble,  
and deploy Fabric  
solutions**

**Learn by example  
with practical  
scenarios**

WebSphere® Business Services Fabric (Fabric) is a comprehensive service-oriented architecture (SOA) offering that is designed to extend the IBM® business process management platform to deliver flexible composite business applications.

In this IBM Redbooks® publication, we provide a complete overview of Fabric, from an architectural introduction, to an installation guide, and a step-by-step scenario that describes how to model, assemble, deploy, and manage composite business applications.

Part 1 of this book is an architectural and technical introduction to Fabric and related concepts.

In Part 2, we use a fictitious business scenario to show you how to build composite business applications in Fabric. We also provide extensive step-by-step instructions.

In Part 3, we discuss how to integrate Fabric with other IBM solutions and technologies, which includes WebSphere Service Registry and Repository, Tivoli® Composite Application Manager for SOA, and Lightweight Directory Access Protocol (LDAP).

## **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](http://ibm.com/redbooks)