

# SOA Approach to Enterprise Integration for Product Lifecycle Management

Strategic capability to integrate systems, processes, and data

Examples based on real-life customer situations

PLM Services 2.0 code samples running on WPS



Rufus Credle  
Michael Bader  
Khirallah Birkler  
Martin Harris  
Mark Holt  
Yoko Hyakuna

Dr. Lutz Lämmer  
Estela McCarty  
Lionel Mommeja  
Dr. Marcos Novaes  
Buddy Raines  
Dr. Vijay Srinivasan





International Technical Support Organization

**SOA Approach to Enterprise Integration for Product  
Lifecycle Management**

October 2008

Archived

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

### **First Edition (October 2008)**

This edition applies to WebSphere Process Server V6.1, WebSphere Integration Developer V6.1, WebSphere Enterprise Service Bus (WESB) V6.1, and WebSphere Service Registry and Repository 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> .....	ix
Trademarks .....	x
<b>Preface</b> .....	xi
The team that wrote this book .....	xi
Become a published author .....	xv
Comments welcome .....	xv
<b>Part 1. Background</b> .....	1
<b>Chapter 1. Product Lifecycle Management overview</b> .....	3
1.1 The need to collaborate and integrate .....	5
1.2 The challenge .....	7
1.3 The solution .....	9
1.4 The future of the PLM integration framework .....	11
<b>Chapter 2. SOA reference architecture for Product Lifecycle Management</b> .....	13
2.1 Decomposing the PLM domain .....	14
2.2 SOA reference architecture for PLM .....	16
2.2.1 Application Integration layer .....	18
2.2.2 The Data Integration layer .....	20
2.2.3 BPM .....	22
2.2.4 Business Service Governance .....	23
2.3 Applying SOA to each PLM discipline .....	24
2.3.1 Engineering Change Management .....	24
2.3.2 Product Data Management .....	26
2.3.3 Simulation Data Management .....	27
2.3.4 Manufacturing Data Management .....	28
2.3.5 Service Data Management .....	28
<b>Chapter 3. Product Lifecycle Management Standards overview</b> .....	31
3.1 Standardized product data and metadata models .....	34
3.1.1 Product data .....	34
3.1.2 Product metadata .....	37
3.2 Architecture of OMG PLM Services .....	45
3.2.1 OMG Model Driven Architecture .....	45
3.2.2 OMG PLM Services architecture .....	47
3.2.3 Architecture of OpenPDM .....	60

3.3 Summary .....	61
<b>Part 2. Scenarios and patterns .....</b>	<b>63</b>
<b>Chapter 4. PLM Services 2.0: The neutral Product Lifecycle</b>	
<b>Management integration .....</b>	<b>65</b>
4.1 Applying PLM Services .....	66
4.1.1 Adopting a neutral PLM integration language .....	66
4.1.2 Using neutral PLM services across application domains .....	69
4.2 Using the PLM Services Informational Model .....	71
4.2.1 PLM base .....	72
4.2.2 Part_identification .....	72
4.2.3 Part_structure .....	74
4.2.4 Document_and_file_management .....	76
4.2.5 Shape_definition_and_transformation .....	77
4.2.6 Classification .....	77
4.2.7 Properties .....	78
4.2.8 Alias_identification .....	78
4.2.9 Authorization .....	78
4.2.10 Configuration_management .....	79
4.2.11 Change_and_work_management .....	80
4.2.12 Process_planning .....	80
4.2.13 Multi_language_support .....	80
4.3 Using the PLM Services Computational Model .....	80
4.3.1 PLM Services connectors .....	81
4.3.2 PLM_object_factory class .....	81
4.3.3 PLM_resource_adapter abstract class .....	81
4.3.4 PLM_connection_factory service .....	82
4.3.5 PLM_service interface .....	82
4.3.6 PLM_connection service .....	82
4.3.7 PLM_general_connection interface .....	83
4.3.8 PLM_message_connection interface .....	84
4.3.9 Query conformance points .....	85
4.4 How to use PLM Services for the product structure traversal .....	94
4.5 How to use PLM Services in the ECM scenario to synchronize the SAP BOM .....	97
4.5.1 ECM participant proposal for a change .....	97
4.5.2 ECM participant comments .....	98
4.5.3 ECM participant approval .....	99
4.5.4 ECM participant detailing and comments .....	100
4.5.5 Usage examples .....	101

<b>Chapter 5. Implementing end-to-end Business Process Management</b>	103
5.1 Business Process Management	106
5.1.1 Adopting process standards	108
5.1.2 Demand for process flexibility and innovation with control	110
5.1.3 A wide range of requirements to address	110
5.1.4 Addressing complexity: processes and participants	111
5.1.5 Maintaining a business user focus	111
5.1.6 BPM: Tactical versus strategic approaches	111
5.1.7 Dynamic processes, BPM and SOA	112
5.1.8 BPM and Enterprise Architecture (EA)	112
5.1.9 Addressing specific business needs with BPM	113
5.1.10 Expertise	114
5.1.11 Empowering LOB and IT to work together, with BPM enabled by SOA	114
5.2 Implementing Business Process Management	114
5.2.1 BPM tooling	118
<b>Chapter 6. Achieving business flexibility with a service-oriented architecture</b>	121
6.1 Achieving business flexibility with a service-oriented architecture	122
6.1.1 Service Design	123
6.1.2 Service Creation	123
6.1.3 Service Connectivity	123
6.1.4 Process	123
6.1.5 Interaction and Collaboration Services	124
6.1.6 Information Services	124
6.1.7 BPM	124
6.1.8 SOA Governance	125
6.1.9 SOA Security and Management	125
6.2 Service Design	126
6.2.1 Service Identification	128
6.2.2 Service Specification	133
6.2.3 Service component specification	137
6.3 SOA Governance	145
6.3.1 SOA Governance project requirements	146
6.3.2 SOA Governance requirements at Trucks Inc.	149
6.3.3 Applying the SOA realization patterns at Trucks Inc.	150
6.3.4 Regulation of New Service Creation	151
6.3.5 Getting more Reuse of Services	153
6.3.6 Enforcing Standards and Best Practices	156
6.3.7 Using IBM WebSphere Service Registry and Repository	159

<b>Part 3. Overview of working examples</b> .....	175
---	-----

<b>Chapter 7. Implementing PLM Services 2.0-based solutions on IBM middleware</b> .....	177
7.1 Preparing the development environment .....	178
7.1.1 Installing WebSphere Integration Developer V6.1 .....	180
7.1.2 The Prostep iViP PLM Services 2.0 reference client .....	181
7.2 PLM Service 2.0 library and Java utilities .....	181
7.2.1 Importing PLM Services library and Java utilities in WebSphere Integration. ....	182
7.2.2 PLM Services library .....	183
7.2.3 Java wrapper classes for the PLM Services 2.0 data types .....	189
7.2.4 A simple PLM connection factory implementation .....	193
7.2.5 Utility classes to implement a PLM Services 2.0 server .....	194
7.3 Implementing a PLM Services 2.0 server .....	200
7.3.1 Implementing PLMConnectionFactory service component .....	204
7.3.2 Implementing PLMGeneralConnection service component .....	207
7.3.3 Implementing PLMMessageConnection service component .....	208
7.3.4 Running the PLM Services 2.0 server. ....	212
7.4 Implementing PLM Services 2.0 connectors .....	217
7.4.1 Exposing PDM systems as PLM Services 2.0 resources .....	217
7.4.2 Handling requests/events from the PDM system .....	218
7.4.3 PLM Services 2.0 connector example. ....	219
<b>Chapter 8. A framework to develop and run flexible PLM Services 2.0-based solutions</b> .....	221
8.1 Motivation .....	221
8.2 Subscription-based routing .....	223
8.2.1 Matching subscriptions .....	223
8.2.2 Interface requirements for the business logic .....	224
8.2.3 User credential propagation .....	225
8.2.4 Configuration data model .....	226
8.2.5 SubscriptionManager Administration .....	228
8.2.6 Sequence diagram of a sample request (query) .....	229
8.2.7 Building components that use the framework .....	231
8.3 Reviewing the implementation .....	232
8.3.1 Reviewing SubscriptionManager implementation .....	232
8.4 Building and reviewing a query scenario. ....	237
8.4.1 Building business logic that plugs into the framework. ....	237

8.5 Running the scenario .....	248
<b>Chapter 9. Design and implementation of an engineering change scenario including SAP integration .....</b>	<b>255</b>
9.1 Scenario overview .....	256
9.2 Engineering Change Request (ECR) process .....	258
9.2.1 Choreographing the message flow .....	260
9.2.2 The human task components .....	268
9.2.3 The ECR Web client .....	269
9.3 SAP backend integration .....	270
9.3.1 Configuration .....	273
9.3.2 Development .....	291
9.4 Running the scenario .....	390
9.4.1 Configuring user and group .....	390
9.4.2 Running the scenario .....	394
9.4.3 Running the scenario with attached SAP system .....	397
<b>Chapter 10. SOA realization of integrated PLM solutions: Working with examples .....</b>	<b>403</b>
10.1 Dynamic service invocation (integration with WSRR) .....	404
10.1.1 WSRR role .....	404
10.1.2 WSRR and PLM Services 2.0 .....	405
10.1.3 A typical business scenario requirement .....	405
10.1.4 Overview of the technical implementation .....	408
10.1.5 Leveraging WSRR .....	408
10.1.6 Laboratory: Building the sample mediation module .....	409
<b>Appendix A. Additional material .....</b>	<b>465</b>
Locating the Web material .....	465
Using the Web material .....	466
How to use the Web material .....	466
<b>Related publications .....</b>	<b>467</b>
Other publications .....	467
Online resources .....	469
How to get Redbooks .....	471
Help from IBM .....	471
<b>Index .....</b>	<b>473</b>



# 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 must 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:

Alerts®	IBM Component Business	Redbooks (logo)  ®
BladeCenter®	Model™	System i®
Component Business Model™	IBM®	System x™
developerWorks®	Rational®	WebSphere®
	Redbooks®	

The following terms are trademarks of other companies:

Rad, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

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

J2EE, Java, JRE, JSP, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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



# Preface

This IBM® Redbooks® Publication documents the Product Development Integration Framework (PDIF) architectural approach towards Enterprise Integration around the Product Lifecycle Management application space. The primary focus being the integration of Product Data Management (PDM) systems to other PDM systems and PDM systems to Enterprise Resource Planning (ERP) systems, mainly SAP®.

The objective of this Redbooks Publication is to document the lessons learned (best practices) and articulate the Service Oriented Architecture (SOA) and Integration Framework that the PDIF team has developed over numerous customer engagements.

This Redbooks publication is defined for an audience of Business Analysts, Software Engineers, IT Architects, and IT Specialists.

## The team that wrote this book

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

**Rufus Credle** is a Certified Consulting IT Specialist at the ITSO, Raleigh Center. In his role as project leader, he conducts residencies and develops Redbooks publications about network operating systems, ERP solutions, voice technology, high availability and clustering solutions, Web application servers, pervasive computing, IBM and Original Equipment Manufacturer (OEM) e-business applications, IBM System x™, System x, and IBM BladeCenter®. Rufus' various positions during his 28-year IBM career have included assignments in administration and asset management, systems engineering, sales and marketing, and IT services. He holds a BS degree in business management from Saint Augustine's College.

**Michael Bader** is an IT specialist working for IBM SWG in Boeblingen, Germany. He has worked with IBM for more than nine years, starting as a Software Engineer developing the WebSphere® Adapter for mySAP™.com®. After five years in software development, he worked for several years as a software consultant with a focus on business process management. Michael joined Dr. Novaes PLM team in 2007 as an IT Architect/Specialist. His areas of expertise include SOA, BPM, PLM Services, and ECM. Michael holds a Diplom-Informatiker degree in Computer Science from the University of Stuttgart, Germany.

**Mark Holt** leads IBM technology strategy for PLM. He has led IBM solutions strategy efforts in areas of PDM and PLM for the last 10 years. Mark has worked in and around the IBM Product Lifecycle Management (PLM) space his entire career. Mark spent the middle part of his career supporting a few large aerospace and defense customers mostly around their PLM deployments. Mark joined IBM Federal Systems as part of the development team for submarine and avionic systems after majoring in Mechanical Engineering at Lehigh University.

**Yoko Hyakuna** is an instructor and course developer for WebSphere Education in United States. She has 10 years of experience in Software development field, and has 6 years of experience in WebSphere products. Her areas of expertise include WebSphere Process Server, WebSphere Integration Developer, and SOA. She holds a Master's degree in statistics from Texas A&M University.

**Estela McCarty** is a Computer Consultant for the eBSC located in La Gaude, France. For the past five to six years, she has been involved with the Web-enabling solutions on the System i® servers and conducted technical workshops for IBM Clients and Business Partners across EMEA, the US, and Asia Pacific. She is a certified IBM Solution Developer for System i and has participated in three System i Redbooks publications. Estela was an IBM consultant with IBM New Zealand before she joined IBM La Gaude as a consultant. She has worked with several WebSphere-based products and has working experiences in the telecommunications, manufacturing, insurance, and education industries. In her first year with the PLM area, she has been involved in installing and configuring the execution environments for integration solutions that use WebSphere products and ISV solutions.

**Lionel Mommeja** is the lead architect of the PLM Center of Excellence at the IBM La Gaude, France. He is a certified IT solutions architect (Open Group # 3313) with a varied background in his 16 years of solutions development. He has worked for major customers in Europe in positions that required him to plan, architect, design, implement, and test enterprise software solutions. His areas of expertise includes business integration, industry solution design, and implementation.

**Dr. Marcos Novaes** is an Executive IT Architect with IBM, based in Somers, NY. He is currently the manager of a team dedicated to the application of SOA technology to the field of PLM, many members of this team are also authors in this publication. Dr. Novaes was also a member of the steering committee that has proposed the OMG PLM Services 2.0 standard. Dr. Novaes has contributed to several technical publications and filed 42 patents in fields related to distributed computing and communication protocols. He joined IBM in 1996, shortly after obtaining his PhD Degree in Computer Science from the University of North Texas.

**Buddy Raines** is Chief Technology Officer for the IBM Extended Product Lifecycle Management (ePLM) solutions organization. Additionally, he is responsible for ePLM's Business Development and Solutions Team charged with defining the foundational architecture and Industry-focused PLM solutions. He is a Licensed Professional Engineer, a PMI Certified Project Manager, a IBM Senior Certified IT Architect, and a IBM Distinguished Engineer. Mr. Raines has two patents, seven published technical disclosures, and has presented at numerous external conferences on Product Data Management and Product Development Reengineering. Mr. Raines received his degree in Industrial Engineering/Operational Research.

**Dr. Vijay Srinivasan** is the Chief Standards and Solutions Officer for PLM, and the Program Manager for PLM Research, Standards, and Academic Programs at IBM. He is also an Adjunct Professor in the Engineering School at Columbia University, New York, where he teaches courses in CAD/CAM, and in the University of North Carolina at Charlotte. He is a member of several national and international standards committees in the areas of product specification, verification, and data exchange. He has published widely on these topics and is the author of the book *Theory of Dimensioning: An Introduction to Parameterizing Geometric Models*.

**Dr. Lutz Lämmer** is a Senior PLM Consultant and works at PROSTEP AG as Key Account Manager in Wolfsburg, Germany. He has more than ten years of experience in PDM integration technology and ISO and OMG standardization. He is also an Adjunct Professor at Technische Universität Darmstadt, where he teaches courses in high performance computing and simulation methods. He has written extensively on PDM integration technology, standards for data exchange and sharing and parallel computing and simulation in engineering. He holds a degree in Civil Engineering from the Technische Universität Dresden.

**Khirallah Birkler** is an IT Architect for IBM SWG Germany. He joined IBM ten years ago and started at the central site of IBM that installs, tests, and assesses all new SAP products, releases, and tools. In the last six years, he has been working for the Software Group Business Partner Technical Strategy and Enablement in the Boeblingen lab. In his role as WebSphere SAP Solution Architect he is responsible to evaluate, design, and architect solutions where IBM and SAP software can be used in conjunction. Khirallah supports worldwide key customers to assess scenarios where IBM software can be placed to complement SAP infrastructures. He has ten years experience in the J2EE™ software development area. He is an IBM certified SOA expert for various WebSphere products and has eight years of experience with SAP Basis, from 4.0 to the latest NetWeaver releases. He holds a Bachelor of Science degree in Information Technology Management from the University of Cooperative Education, Stuttgart.

**Martin Harris** is an IBM certified Senior Business Architect within the IBM Software Group organization. Based in the UK, he has over 10 years experience working in the IBM Product Lifecycle Management (PLM) team and 5 years in the Aerospace and Defence industry prior. He specializes in Business Process Management (BPM), specifically within aerospace and defence as well as automotive industries. Mr. Harris is a regular contributor to process standards organizations development activities as an industry and domain subject matter expert and frequently presents at conferences and events on the topics of BPM as well as PLM.

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

Tamikia Barrow, Margaret Ticknor  
International Technical Support Organization, Raleigh Center

William P. Day, Industry Solutions Program/Project Manager, SA-I707: ITSO  
Redbook development  
IBM Providence - USA

Arnaud Desprets, IT Architect SOA - WebSphere Specialist - Web Services  
Specialist - Security  
IBM Montpellier France

Lore Eisenstaedt, Solutions Architect, SWG Industry Solutions - BPM Specialist  
IBM Chicago USA

Etienne Charpentier, IT Specialist, Product Development Integration Framework  
(PDIF)  
IBM LaGaude France

Christian Hansen, Solution Architect, Service Oriented Architecture  
IBM LaGaude France

Guido Adolphi, Consultant Product Data Management  
PROSTEP AG, Darmstadt - Germany

Bhargav Perepa Venkata Ramana, Certified WebSphere IT Specialist, IBM  
Federal Software Group  
IBM Fairfax USA

Kay Freund, IBM SWG Market Manager for SOA and Frameworks, Industrial  
Sector  
IBM Austin

## 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 Redbooks publication 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

## Background

Part 1 of this Redbooks publication provides background information to help understand Product Lifecycle Management, how IBM enables service-oriented architecture for Product Lifecycle Management, and Product Lifecycle Management Standards.

Archived



# Product Lifecycle Management overview

This Redbooks publication introduces the IBM approach to enable a service-oriented architecture (SOA) for Product Lifecycle Management (PLM). So, what is PLM and why is it important to a company's business?

PLM can come into focus quickly if we think about products that we see and use everyday. Think about the car you drive, or the trucks on the road, planes, ships, your MP3 player, the desk at which you work, even the clothes and shoes you wear. These products are all the end result of some ideas that were conceived to satisfy some customer requirements or to meet a marketplace solution opportunity, and in the end something that someone would purchase.

PLM is a strategic capability to integrate systems, processes, and data across the complete product life cycle in a way that enables the use of information for the following reasons:

- ▶ To better conceive ideas to meet the marketplace demands
- ▶ To accurately transform these ideas into products that can be produced
- ▶ To source and manufacture the products
- ▶ To continue with support and enablement of the products throughout their in service life

A PLM system, as shown in Figure 1-1, includes the tools, applications, IT systems, manufacturing systems, company business processes, and data that span the full life cycle of a product from initial idea through the end of the product life in service.



Figure 1-1 Scope of PLM continues to expand and mature

Competition is more intense than ever before, and companies are struggling to maintain their competitive advantage in an increasingly complex business environment, characterized by globalization, fierce competition, mergers and acquisitions, and a wide range of specific customer demands. The global IBM CEO study (Figure 1-2 on page 6) of nearly eight hundred top CEOs has identified innovation as key to business success. CEOs not only focus on product innovation, but also extend the need for innovation to their business models and the processes applied across the enterprise. Major business leaders think of innovation as an economic and social imperative, especially critical in breaking out of established environments and boundaries. For manufacturing companies, PLM directly targets the collaboration and integration requirements of innovation. Its relevance has expanded across the full product life cycle. Companies are looking beyond engineering to use the expanded PLM capabilities to innovate their processes and business models to bring high value products to market.

How a company implements PLM often affects their strategic ability to be competitive in the marketplace. However, a complete PLM system is not something a company can buy off the shelf. Applications and authoring tools used within the PLM system can be purchased from multiple software vendors. The same is true for the computing and networking platforms on which they execute. The PLM landscape typically grows through introduction of different systems or applications over many years, some developed within the company, and some introduced through corporate acquisitions of other business units or companies. Maybe even more importantly, the processes used within a company make it unique. A well-implemented PLM system allows a company the following advantages:

- ▶ To create new and innovative products
- ▶ To assess their functional capabilities and ability to satisfy what the market is demanding in the early stages of product development
- ▶ To use this information to create detailed engineering and manufacturing designs
- ▶ To support the maintenance and changes to the product throughout its in service life

## 1.1 The need to collaborate and integrate

The inability to collaborate well across product development teams, coupled with the lack of integration of product development and manufacturing systems often prevent companies from effectively innovating. It is not just an issue internally within a company. It extends to the network of design and production partners, particularly as more OEM's transfer up to 70% of the product design and manufacturing to partners or suppliers. However, many global companies are not ready to fully collaborate with business partners, vendors and customers. Collaboration requires the integration of information and processes. Integration also presents big challenges as companies must align the overall IT landscape to enable innovation.

**The IBM CEO study confirms that innovation improves results, but that there are still gaps in companies' ability to integrate and collaborate**

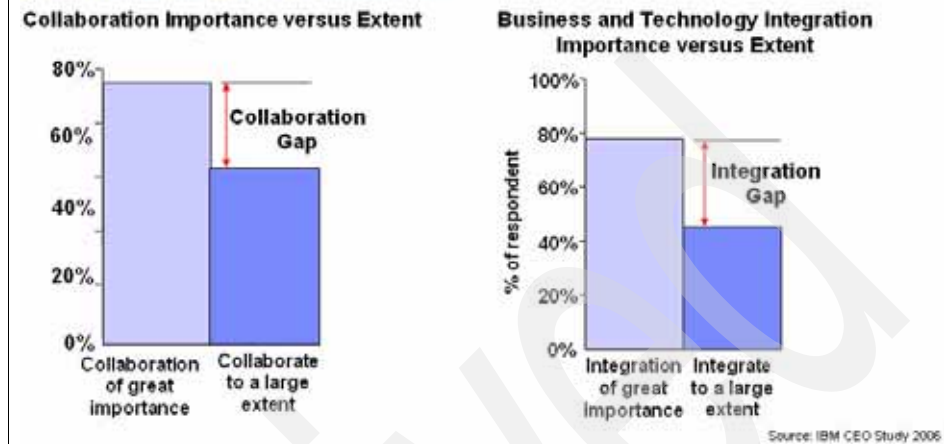


Figure 1-2 IBM PLM is focused on addressing integration and collaboration gaps

PLM presents the opportunity to accelerate innovation because it strikes at the heart of the issues of collaboration and integration. The focus of PLM has expanded with companies shifting their business models from simply developing and selling a product to addressing the full customer requirements of the product across its complete life cycle. Product information has to combine customer requirements and market needs to achieve performance targets across the entire life cycle. Governments have become even more involved in legislating responsibility for a product through retirement or disposal. PLM must cover the full scope of the product life cycle from the initial idea in portfolio planning, to structuring and development of the product, through testing and interacting with marketing, sales and finance, through service of the product in the field, until product retirement or disposal.

The need for a more integrated and collaborative PLM environment is also being driven by the increased electronics and software content of today's products. The ever increasing complexity of mechanical, electronic, and software content requires a systems engineering approach to product development. Realizing the necessary business transformation to achieve a product development approach based on systems engineering practices and methodology requires an ability to manage business processes and information in a contextual relationship from one end of the business to the other. This requires an end-to-end collaborative and integrated PLM environment.

## 1.2 The challenge

Computed-Aided Design (CAD) and other authoring tools such as Electronics Design Automation (EDA) and Computed-Aided Engineering (CAE) software traditionally rely on tightly integrated proprietary data management platforms to directly and easily manage interaction and support concurrent engineering by multiple people within a design or engineering discipline. A known PLM challenge has emerged with the use of these tools that are not well-integrated with enterprise Product Data Management (PDM), Enterprise Resource Planning (ERP), and Customer Resource Management (CRM) systems.

The integrated PLM challenge is further compounded when you consider that the same situation exists within each partner or supplier involved in the product development life cycle. Today, most companies experience impacts to their productivity, cycle time, and quality due to the disconnected state of their current PLM environment (Figure 1-3). There is a need to define and execute company-specific business processes that span the different application silos. Business benefits from multi-disciplinary business processes such as commonality and reuse, cost and weight roll-up, supply chain integration, as well as early analysis and simulation cannot be achieved with the typical current PLM environments.

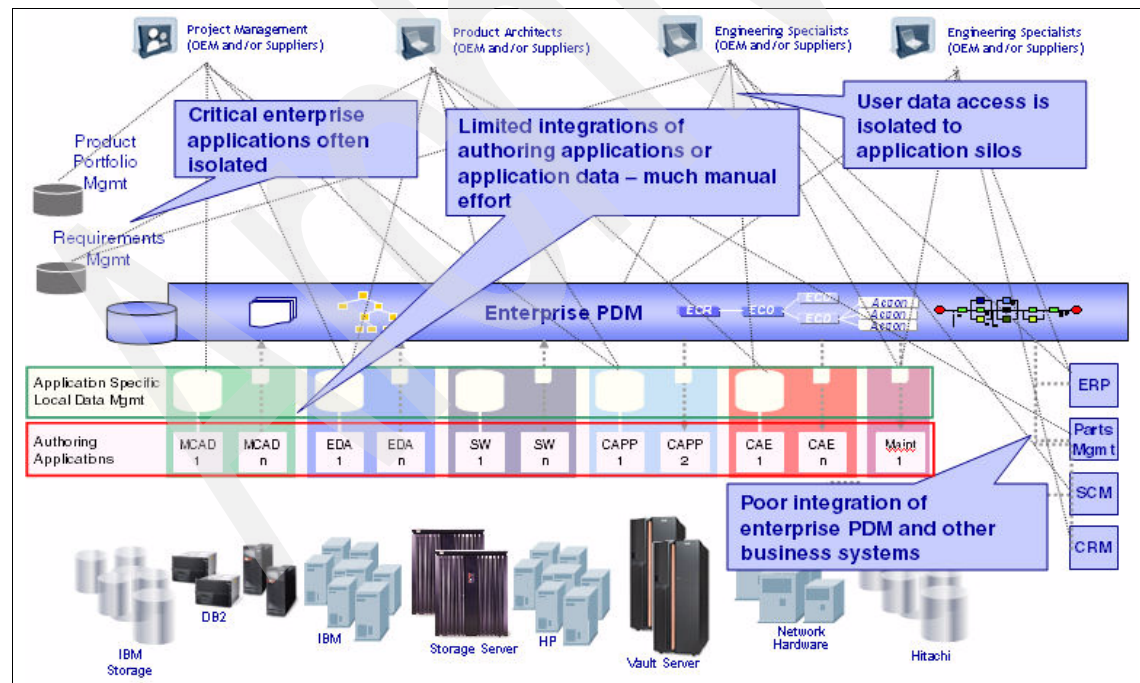


Figure 1-3 Common PLM environment

Implementing product development business processes across the mixed application platforms as well as across the extended partner and supplier network presents complicated challenges (Figure 1-4) that are not well addressed by existing enterprise PDM systems.

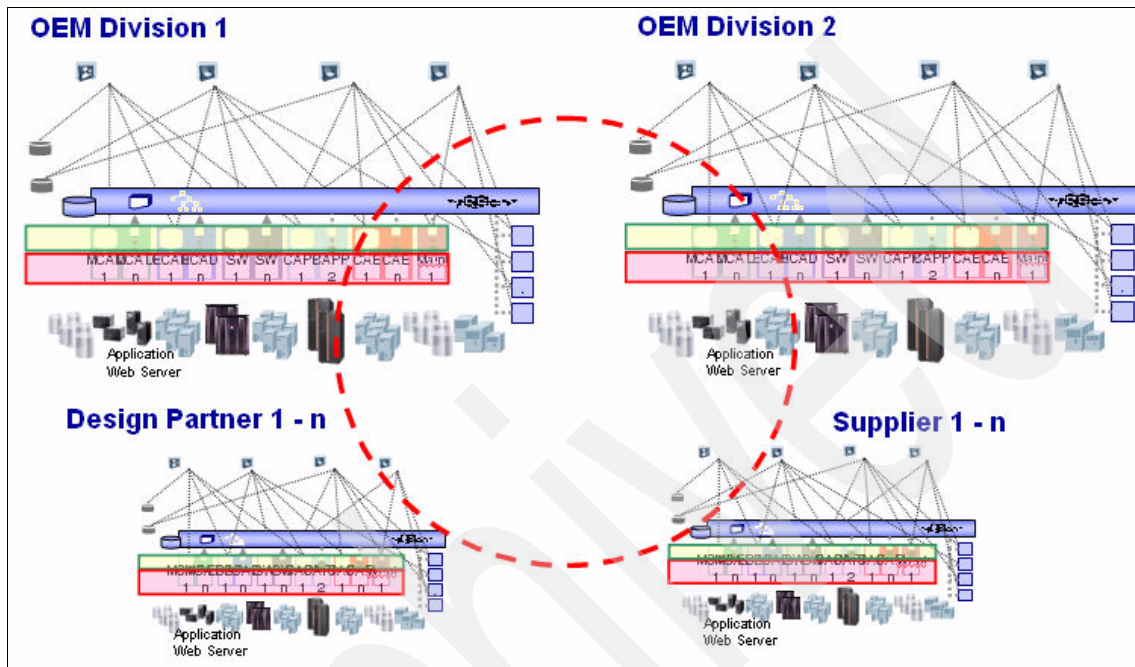


Figure 1-4 Complexity compounded

Within a single company, different applications often exist to address the same business function. This situation develops with the integration of different business units, global design efforts where different tools were used in different geographies, as well as across companies formed by mergers or acquisitions. As an example, an automotive company shared that they have 30 different repositories for requirements. Understanding, much less linking, all of the different requirements related to a new design in that environment presents difficulties. It becomes easy to understand why a lot of the time is spent just looking for data, a source of major inefficiencies.

## 1.3 The solution

IBM is approaching the problems of collaboration and integration of the PLM environment with a SOA approach for PLM. A fundamental underlying principle of this effort is based on open IT and industry standards. An SOA approach to PLM must make PLM information available to any person or system that needs information and allow information flows to be orchestrated to support company-specific business processes. Ultimately, an SOA environment for PLM allows relationships between information in multiple different applications to be defined and managed to provide insight that has only been possible before when all data resided in a single system.

During its internal re-engineering activity in the early 90s, IBM began to address the same issues that we are talking about in this Redbooks publication, namely missing connectivity around key processes. ERP and PDM integration relied on portals for exchanging engineering data through a Web-based interface to our design partners. The approach cut the IBM development time in half and saved roughly eight hundred million dollars in re-use. While the many point to point integrations put in place help achieve significant savings, they were too complex and costly to maintain. To reduce this complexity, IBM moved to a service-oriented architecture, and today, we recommend the same to our clients. The change to SOA also directly addresses issues such as integration with common component libraries that serve commonality and re-use and drive additional business savings and benefits.

SOA depends on common services on an Enterprise Service Bus (ESB) and a lightweight interface to suppliers and design partners. The automatic instantiation of services in the IT infrastructure is key to enabling business flexibility and improving innovation capabilities. An SOA breaks a business function into repeatable business tasks that can be exposed and re-used. The users can control the information they need and can quickly re-orchestrate a process based on changing business needs. With the right tools and infrastructure, the development process may be improved continuously. Rolling up information such as cost and weight from multiple sources may then involve a federation of data to provide required information at the right time to the right person to make a decision, versus direct manual accesses to multiple systems. Open standards and a flexible infrastructure are absolutely critical.

IBM is working with PLM leaders and business partners, in combination with IBM integration technology, to support best practices and processes through a SOA integration framework for PLM (Figure 1-5 on page 10). Instead of approaching PDM and ERP integration with one-time, one-off services offerings, a framework based on standards can provide and enable re-usable assets and guide the tasks quickly and reliably. Instantiation of PLM specific semantics and knowledge capital into a PLM framework supports deployment of a company wide PLM



information and process backbone. In a service oriented approach, the processes and best practices are based on open standards and remain the same allowing migration to next levels of PDM or ERP systems without changing the whole infrastructure. Such a framework is critical for our clients and involves defining patterns and working with the right partners to harden repeatable assets. Reusing business process logic, employing reusable application interfaces and relying on a standards-based neutral object model are the foundation principle of the IBM integration framework for PLM.

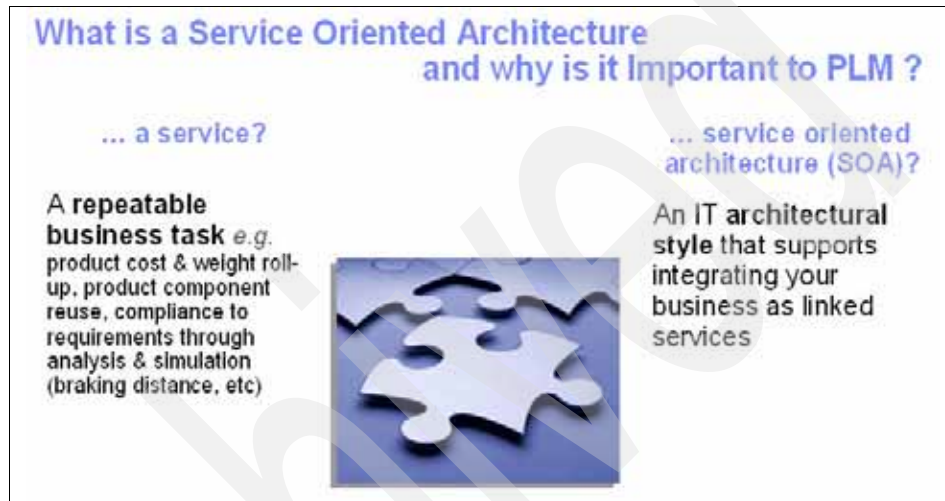


Figure 1-5 SOA impacts every aspect of IT and business

A collaboration layer allows for role-based access to the data that exists across a company's heterogeneous PLM environment. A process layer supports company-specific business processes, allowing for process definition, process management, and process changes supporting continuous improvements.

At the core of the integration framework (Figure 1-6 on page 11) are core PLM services that provide reusable functions necessary to implement the desired business processes. Services in the area of Bill of Material (BoM) management or cost-to-weight roll-up, or configuration management for cross-PLM information relationships support enterprise-level PLM business processes. Services may be developed by IBM or our business partners. Many of the existing PLM application providers are moving to provide more direct service interfaces. Additionally, IBM partners with other companies, such as PROSTEP AG, to provide the required information integration services and connectivity.



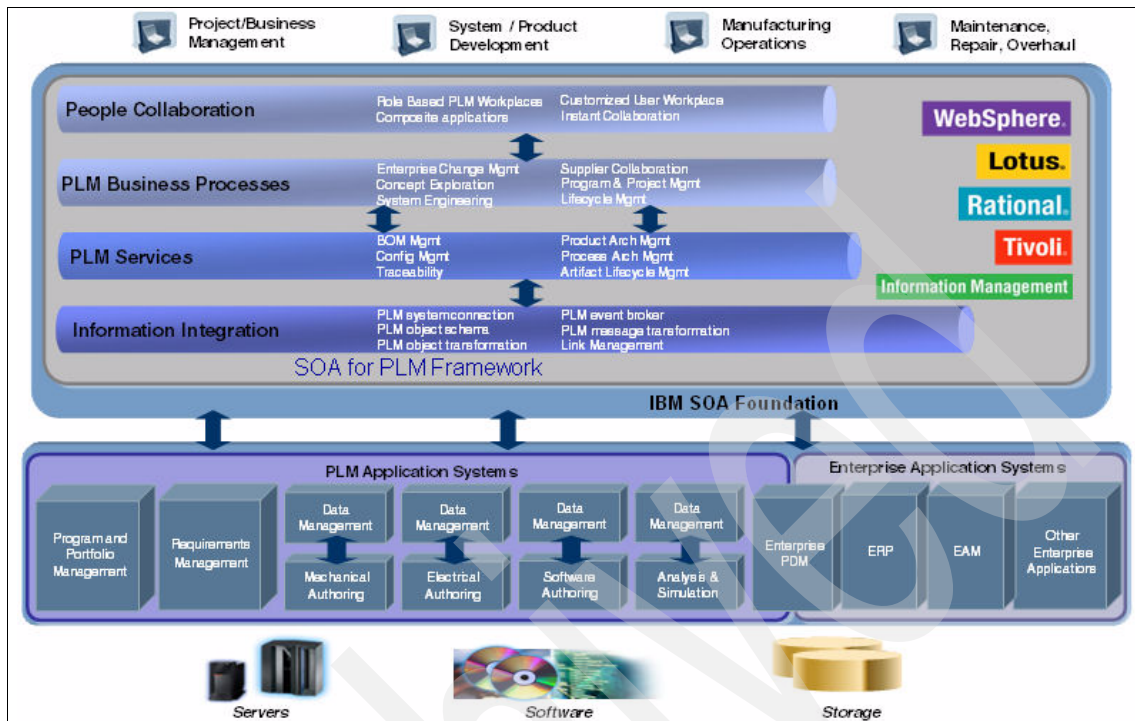


Figure 1-6 SOA for PLM environment

The disconnect between manufacturing and engineering is a critical concern of many of our clients. The focus on enterprise integration and collaboration between the areas includes process orchestration and role-based access to data through our portal and collaboration tools to maintain synchronization between the BOM and PDM system. As systems plug into a common open standards bus, they gain access to necessary data in a light way using various portal products.

## 1.4 The future of the PLM integration framework

We are concentrating efforts on model driven systems development to get behind the complexity of systems modeling and design. We are exploring the key technologies required to help companies drive model-driven development, and the underlying infrastructure needed to support that approach. Our clients need to manage the relationships between all of these artifacts throughout the life cycle of a product (Figure 1-7 on page 12). Trying to link all the requirements and use cases with all the mechanical parts and software becomes a complex endeavor to manage over time.

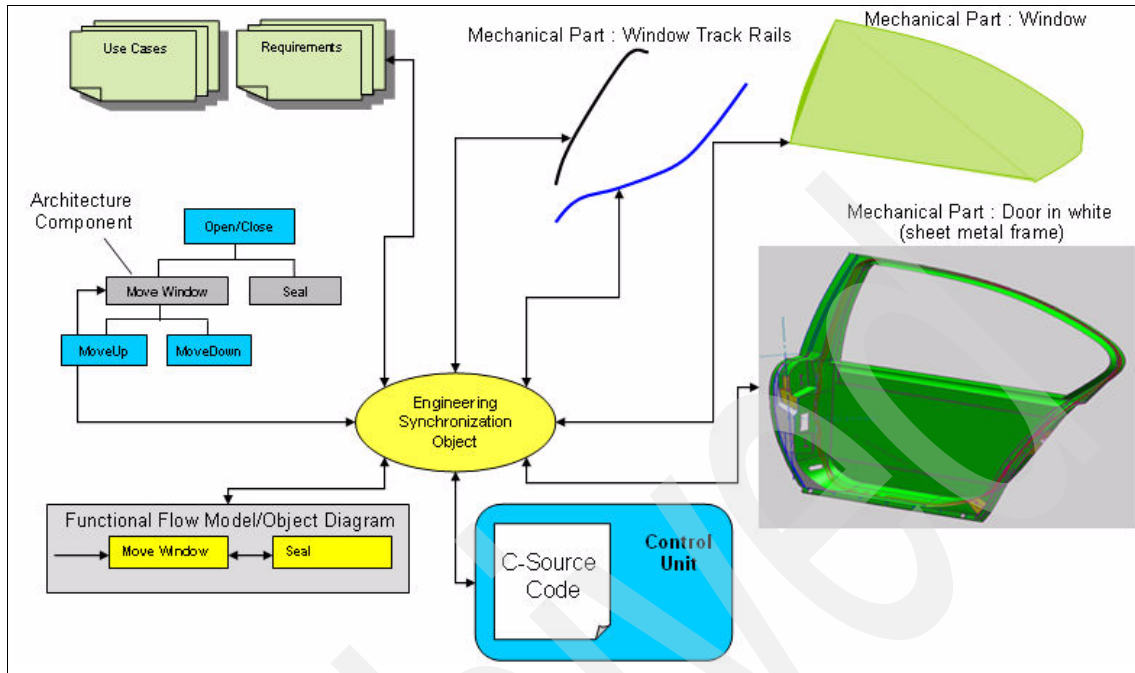


Figure 1-7 Managing information relationships across the PLM

To solve automotive warranty issues, we have to enable connectivity and management of complex systems. Eventually, most upgrades relate to embedded software and electronics. In this case, we must extend management of systems beyond traditional CAD and PDM integration. We need a more federated view, the management and persistence of data over time, and the access by a large group of people in a controlled and secure way to protect our client's intellectual capital.

Our PLM integration framework continues to expand and provide a broader PLM semantic based on open standards as well as a larger portfolio of services to support the complex PLM challenges of our customers.

# SOA reference architecture for Product Lifecycle Management

The first chapter discussed how diverse the Product Lifecycle Management (PLM) environment is, and the need for integrating the multitude of processes and information sources that are part of the complex PLM ecosystem. This chapter examines how SOA technology can be applied to fulfill many of these objectives. This chapter is structured as follows:

- ▶ Section 2.1, “Decomposing the PLM domain” on page 14 explains the how the PLM Ecosystem can be divided into a number of PLM disciplines.
- ▶ Section 2.2, “SOA reference architecture for PLM” on page 16 introduces a reference architecture for the application of SOA technology in the domain of PLM.
- ▶ Section 2.3, “Applying SOA to each PLM discipline” on page 24 explains how SOA technology can be applied to each PLM discipline.

## 2.1 Decomposing the PLM domain

The PLM domain can be decomposed into several disciplines. The list below is not exhaustive, but it contains many of the most important PLM disciplines:

- ▶ **Product Data Management**  
This discipline refers to the management of product data during the design phase of a product
- ▶ **Simulation Data Management**  
This discipline refers to the management of numerical intensive simulation data for each product configuration
- ▶ **Manufacturing Data Management**  
This discipline refers to the management of Bill of Materials used at the manufacturing phase
- ▶ **Service Data Management**  
This discipline refers to the management of service data during the life of product assets
- ▶ **Engineering Change Management**  
This discipline refers to a process that controls the flow of change requests throughout the life cycle of a product

Notice that the top four disciplines are related to data management in the various phases of the product life cycle. The last discipline, Engineering Change Management (ECM) is a Business Process Management (BPM) discipline that spans all the phases of the product life cycle.

Figure 2-1 on page 15 illustrates the relationships of these disciplines to the product life cycle.

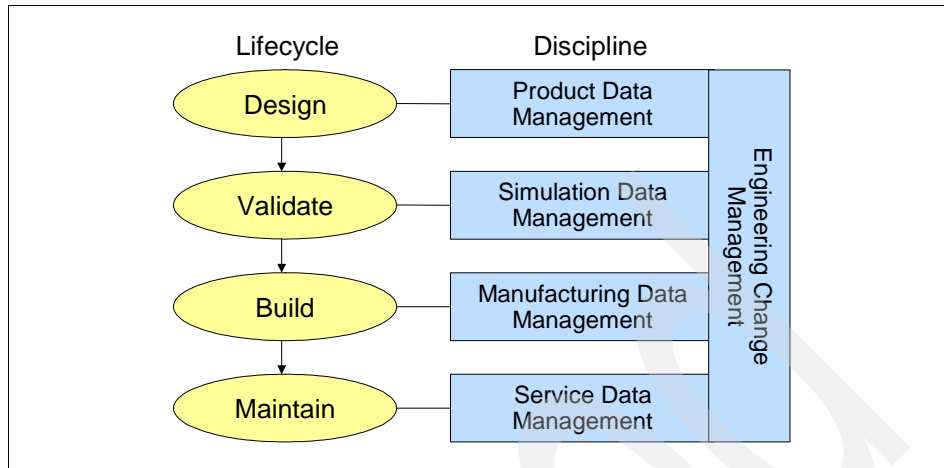


Figure 2-1 PLM disciplines and the product life cycle

Figure 2-1 shows how the four data management disciplines correspond to each phase of the product life cycle, while the ECM discipline spans all phases. We can now discuss how to use SOA technology to solve many of the challenges that were discussed in Section 1.2, “The challenge” on page 7, where we discussed the diversity of the application landscape and the complexity of the IT environment. These challenges make it difficult for the data and processes to flow smoothly among the product life cycle phases. We can categorize the main problems discussed in Section 1.2, “The challenge” on page 7 as follows:

- Application Integration problems

There are many applications used in the PLM environment. This diversity tends to segment the data into silos, and makes it difficult to build business processes that span multiple applications.

- Data Integration problems

The data is scattered over the enterprise, not only among different applications, but also among different line of business. Also, many products are manufactured using a network of suppliers and collaborators, which aggravates the data integration problem.

- BPM problems

Each phase of the product development life cycle involves the coordination of several business processes that depend on manual operations. The poor execution of such business processes limits their optimization and therefore limits the ability of manufacturers both to reduce their product life cycles and to achieve innovation leveraging business process transformation.

► Business Service Governance problems

The three problems above indicate the extraordinary complexity of the PLM environment, with a plurality of applications, data sources, and business processes. Tackling each of these problems individually never achieves the optimization of the PLM landscape. It is necessary to build a governance strategy by which each application, data source, and business process is represented as a business service, and a governance model must be created for the efficient use of all components.

In the sections that follow, we show how each of these problems can be solved using SOA technology, resulting in an uniform collection of business services that can be developed and managed by an efficient governance model.

## 2.2 SOA reference architecture for PLM

This section discusses the SOA architecture with which we can build the base of the PLM landscape. This architecture was built during years of practice in real projects, using the latest SOA technology available. The PLM integration architecture we developed brings together the core PLM disciplines as shown in Figure 2-2 on page 17. This architecture proposes four base integration layers, which support the various PLM disciplines and lead to an efficient PLM Infrastructure (Figure 2-2 on page 17).

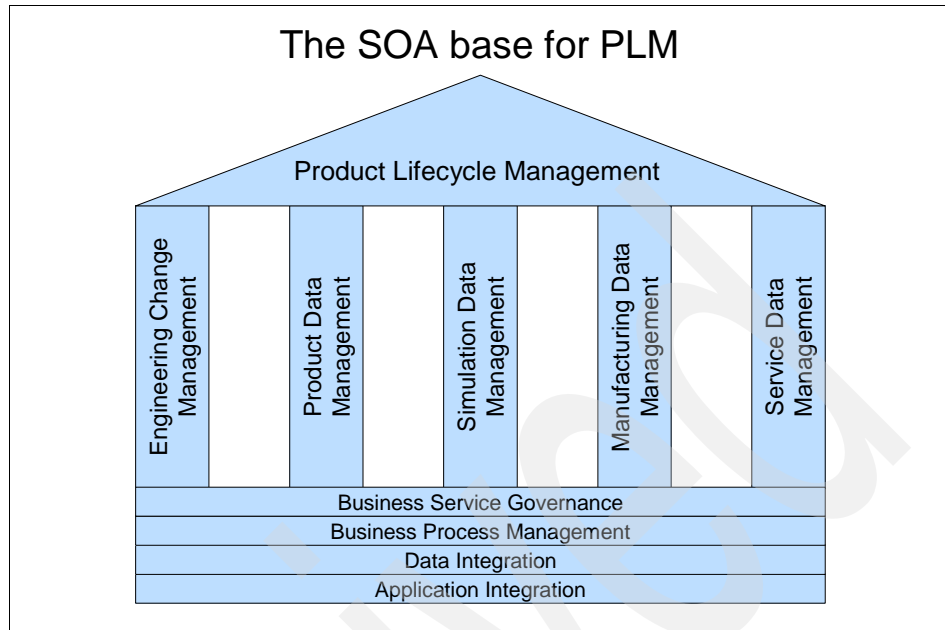


Figure 2-2 PLM Integration Architecture

The implementation of the PLM Integration Architecture uses the Enterprise Service Bus (ESB) pattern for application integration. This integration pattern is known as the SOA Architecture for PLM, shown in Figure 2-3.

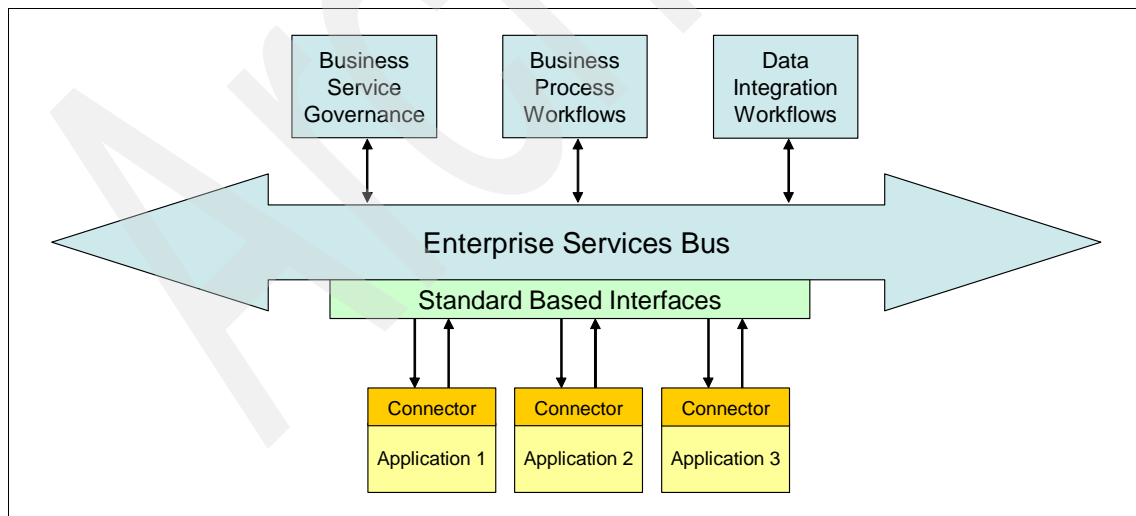


Figure 2-3 SOA Architecture Pattern for PLM

In Figure 2-3 on page 17, the SOA Architecture Pattern for PLM is based on the ESB architectural pattern. The Application Integration layer is shown in the lower portion of the bus, while the three other layers are presented by the three boxes attached to the top of the bus. We cannot simply connect these components to the bus without the proper design and data model. We would be creating complexity as opposed to reducing it. In particular, it is extremely important that the application interfaces to one single standard-based data model and protocol.

In order to implement the protocol and data model convergence for all applications, the architecture above uses connectors that can translate back and forth from the application data model to the standard-based model. The standard must be carefully chosen. In the domain of PLM extensive work has been done by the International Standards Organization (ISO), the Object Management Group (OMG) and the Open Application Group Integration Specification (OAGIS). Chapter 3, “Product Lifecycle Management Standards overview” on page 31 discusses all these standards in detail. In this publication, we have selected the OMG PLM Services 2.0, which is based on the ISO AP214 Standard for Exchange of Product Data (STEP) and the ECM recommendation of the Verband der Automobilindustrie (VDA).

The following sections discuss how each layer is connected to the bus and the role they play in the overall architecture.

### **2.2.1 Application Integration layer**

The Application Integration layer of our SOA architecture addresses the need to connect the various applications to the ESB. Typically, a connector is used to provide SOA connectivity to applications. In other cases, applications may provide SOA interfaces already. Regardless of the way in which the connectivity happens, the most important consideration is that, whenever possible, the interface used to the ESB must be based on an open industry standard. This ensures that the data model and the data access semantics used by the other layers match the other layers. Also, the adoption of an open industry standard reduces the complexity of the IT landscape by reducing the number of APIs and data models used in the enterprise.

In addition to the adoption of an open industry standard for connectivity and data modeling, another important consideration is bi-directional connectivity. The bi-directional capability states not only that the ESB must be able to make requests to the application, but also that the application must be able to make requests to the ESB. This bi-directional capability is needed in many scenarios, but it is only recently that connector standards have been updated to provide bi-directional capability. For example, this capability was only included in the latest release of the Java Connector Architecture (JCA) standard, JCA 1.5.



Consider the following two common application integration scenarios that demand bi-directional capability:

- ▶ An application user uses some GUI in order to execute an action, such as creating a new item. Let us suppose that some of the information needed to create this item must be extracted from an older Enterprise Information System. With bi-directional connectivity the application can call the bus in order to retrieve the required parameters.
- ▶ A user updates some data in the application. Let us assume that we would like to generate some event so that this update can trigger some actions in other applications. We would be able to achieve this by structuring the application to into the Bus to inform of the update.

The adoption of an open industry standard and the bi-directional capability can now be combined. That is, the same open industry standard used for providing connectivity into applications can provide connectivity to the bus. We call this *symmetric bi-directional capability*. In Part 3, “Overview of working examples” on page 175, we provide an example of how to build a symmetric bi-directional connector using the OMG PLM Services standard.

In our architecture, the bi-directional capability can be provided by a JCA 1.5 connector, or by a simpler Web services connector with bi-directional capability, as we have done in Part 3, “Overview of working examples” on page 175.

A service call that is initiated from the bus targeting the application is called an *outbound* service. Conversely, a service call initiated by the application and targeting the bus is called an *inbound* service. This terminology considers the bus as the reference point concerning the service direction, and not the application. This terminology, which we have adopted in this book, is consistent with the JCA 1.5 terminology.

Figure 2-4 depicts a symmetric bi-directional connector providing both inbound and outbound connectivity to the ESB using an open industry standard.

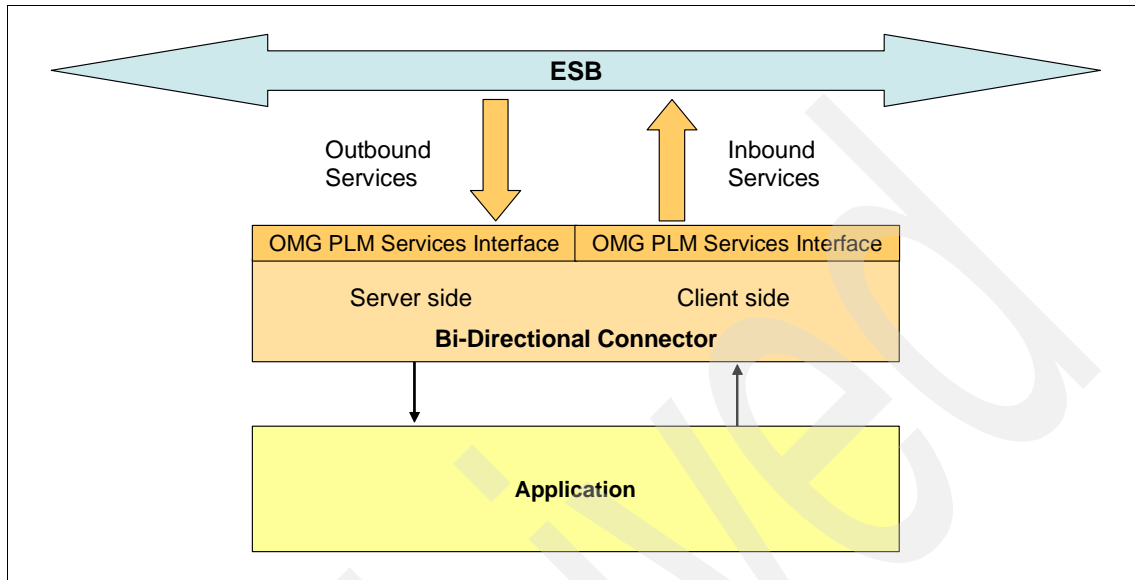


Figure 2-4 A Symmetric Bi-directional Connector

The use of this connector is discussed in Part 3, “Overview of working examples” on page 175.

## 2.2.2 The Data Integration layer

The Data Integration approach we have used in this book is known as *Information as a Service*. Rather than attempting a massive data consolidation strategy that would involve the extraction of all the data from several applications into a central database, the Information as a Service approach implements federated queries, which can provide a unified view of the data contained in several applications. This approach avoids the replication in a central database, therefore avoiding data synchronization and consistency problems.

This approach is best represented by an example. Suppose that we have two PDM systems in the enterprise, one used by engineers working on a truck’s cabin, and another system that is working on the chassis and electrical system. This situation is actually common, because different tools are usually used to create surfaces and electrical wirings. In our example, we normalize both applications to a common standard (OMG PLM Services 2.0), which provides a query interface. We can now build a service that provides the same standard query interface, and can provide an integrated view of the data stored in both

systems (the entire truck). A client using this service would not notice that the data is actually being consolidated from two different systems. This is the beauty of the Information as a Service approach.

Typically, the Information Service has to execute a workflow that extracts information from the two systems, and then produce an integrated view. Such workflows are implemented as Business Process Execution Language (BPEL) workflows, which are executed by the WebSphere Process Server. Figure 2-5 illustrates this example.

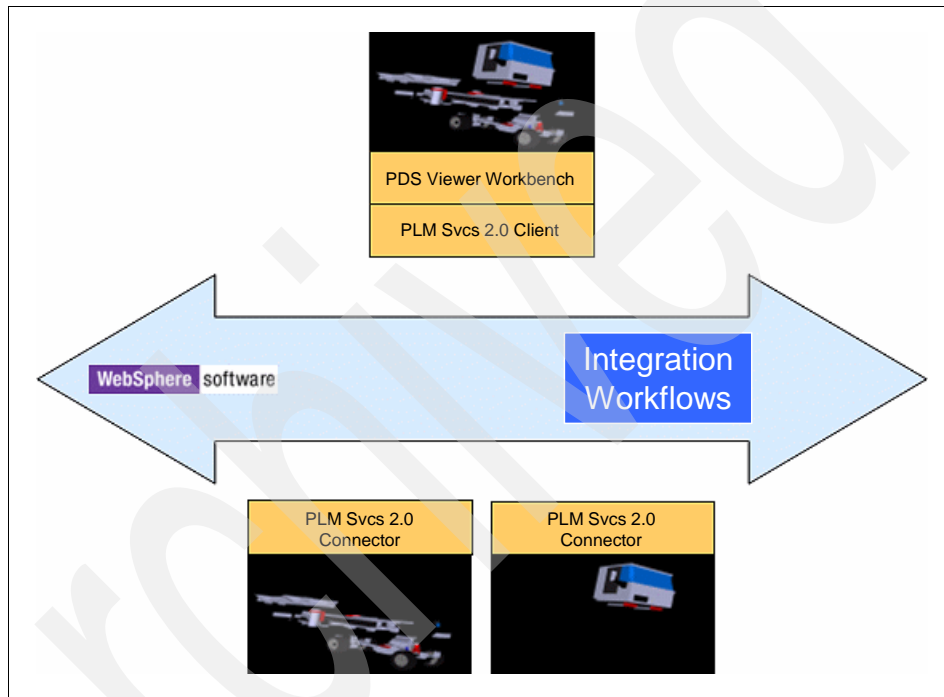


Figure 2-5 Information as a Service

The scenario shown in Figure 2-5 is implemented in Part 3, "Overview of working examples" on page 175.

### 2.2.3 BPM

BPM (BPM) is the ability to optimize and manage the multitude of business processes that exist in an enterprise. Recently, many SOA-based tools have emerged for implementing BPM. The synergy between BPM and SOA technology comes from the exploitation of workflow tools. For example, the tool most widely used for creating SOA applications is based on workflows developed with the BPEL. On the other hand, business analysts have also traditionally relied on workflow technology to model enterprise processes. With the adoption of SOA technology, it now becomes possible to transform high level business process workflows into actual SOA implementations in BPEL.

Another important development in the area of BPM has been the enablement of Business Process Monitoring. During the Business Process Definition phase, the analyst can create Key Performance Indicators (KPIs), which are used in real time to monitor the performance of the business process. The KPIs are indicators of the business performance of the process, such as cost, time, milestones, and so forth. During the execution of the business process, the process analyst is able to monitor the business performance of the process and detect possibilities for improvement. The analyst is then able to re-engineer the business process, which is then sent for re-implementation by IT specialists. Because it is possible to derive the BPEL implementation directly from the business process model, this process is straightforward.

This practice enables a continuous improvement cycle that constantly transfers the business analysts requirements to IT specialist for implementation. The processes are then monitored and re-engineered when needed in subsequent iterations. This methodology is shown in Figure 2-6 on page 23.

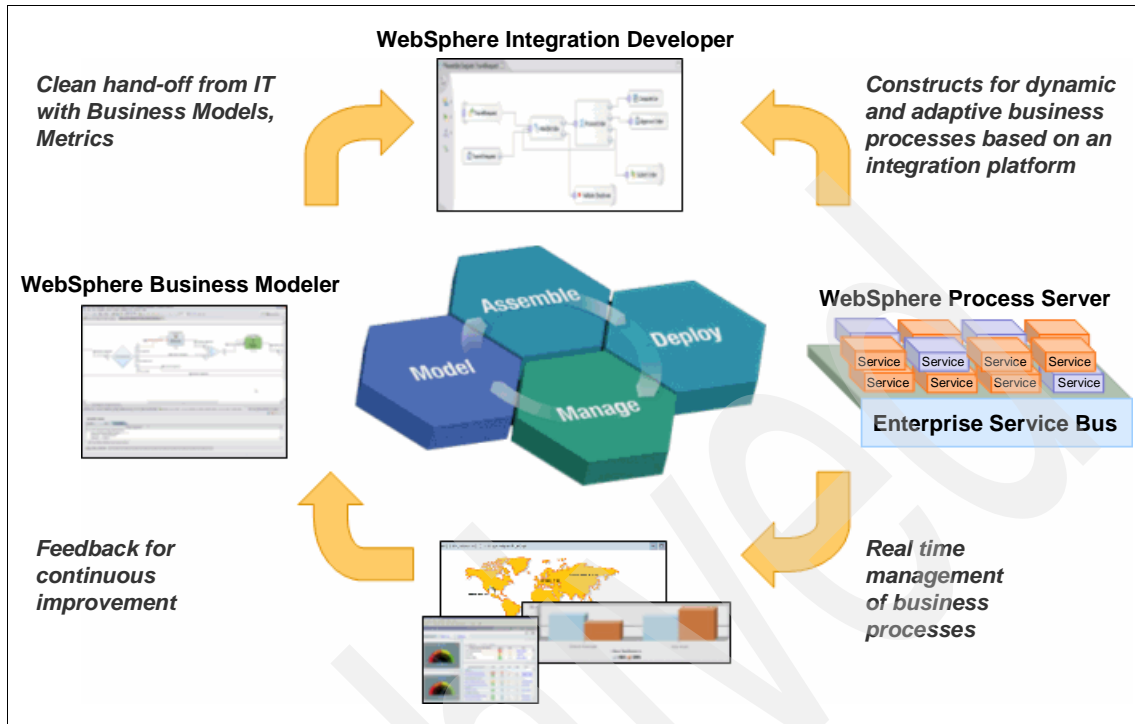


Figure 2-6 Enabling a complete life cycle of business process

Chapter 5, “Implementing end-to-end Business Process Management” on page 103 provides an example of the application of BPM to the PLM environment.

## 2.2.4 Business Service Governance

The top layer of our SOA architecture for PLM is related to Business Service Governance. Note that in the other three layers we have produced a phenomenal number of services:

- ▶ Applications encapsulated as services
- ▶ Information offered as a service
- ▶ Business processes with service interfaces

As the adoption of SOA technology grows within an enterprise, so does the number of business services that are deployed. Not only is the number of services large, but new versions of services are likely to be continually introduced. Finally, the number of applications using such services is also likely to grow, as does the need for a Business Service Governance.

Business Service Governance is a complex topic, because it involves all phases of SOA governance. Fortunately, this subject is covered in detail in a previously published Redbooks publication, *Implementing Technology to Support SOA Governance and Management*, SG24-7538.

For the purposes of this Redbooks publication, we are particularly interested in the Service Endpoint Management aspect of SOA governance. In Part 3, “Overview of working examples” on page 175, we discuss two techniques for handling endpoint resolution using Service Registry Lookups.

## 2.3 Applying SOA to each PLM discipline

This section describes how SOA can benefit each of the following PLM disciplines:

- ▶ “Engineering Change Management” on page 24
- ▶ “Product Data Management” on page 26
- ▶ “Simulation Data Management” on page 27
- ▶ “Manufacturing Data Management” on page 28
- ▶ “Service Data Management” on page 28

### 2.3.1 Engineering Change Management

One of the most important areas of PLM is Engineering Change Management (ECM). The ECM process regulates the business process by which an engineering change is requested and is propagated throughout the product development pipeline. This process is extremely important both in the context of a single enterprise as well as an enterprise collaboration scenario. In the case of enterprise collaboration, it is important to adopt a standard for both the ECM process and its associated data types.

At the time of publication, the following two standards are considered the most important ECM standards:

- ▶ VDA 4965  
Proposed by the Verband der Automobilindustrie (VDA)
- ▶ Configuration Management II (CMII)  
Proposed by the Institute for Configuration Management

Part 3, “Overview of working examples” on page 175 illustrates an example that uses the VDA 4965 standard. To implement a ECM business process, it is necessary to adopt a protocol capable of handling the business process requests and data types. The OMG PLM Services 2.0 standard is a SOA protocol that was

designed to support the VDA 4965 business processes. The subsequent sections of this publication contain a discussion of the OMG PLM Services 2.0 standard and its relationship to the VDA 4965 standard. Part 3, “Overview of working examples” on page 175, has the actual implementation of an engineering change scenario using the VDA 4965 ECM standard and the OMG PLM Services 2.0 protocol.

Figure 2-7 shows a high level view of the VDA 4965 ECM process and its milestones. The figure shows a deep view of step three of the process, “Specification and Decision of Change” where we see the Engineering Change Request (ECR) subprocess and milestones. In Part 3, “Overview of working examples” on page 175 we see an instance of the ECR process.

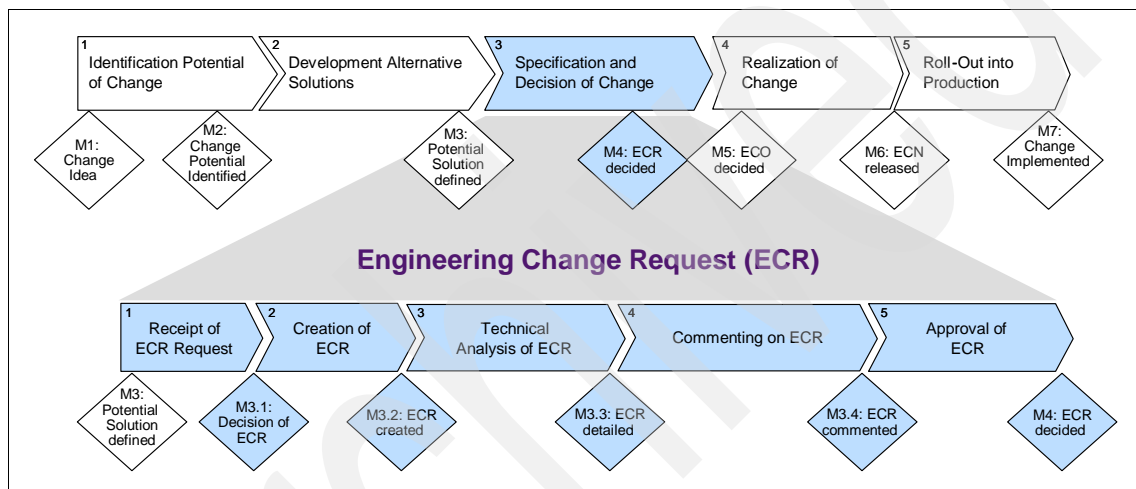


Figure 2-7 The ECM process and its milestones

The Engineering Change Process is a business process that can span many applications, and potentially many different enterprises in a supplier collaboration scenario. This is why it benefits from SOA technology. The business process can be expressed and implemented in the BPEL, while the OMG PLM Services 2.0 standard offers a common data model and protocol that can enable this process to span multiple applications and multiple participant enterprises.

## 2.3.2 Product Data Management

Product Data Management (PDM) is one of the most important PLM disciplines. It includes all aspects related to the management of the product data, such as the Product Data Structure (PDS), the product configuration, and so forth. The ISO AP214 STEP standard has been specially designed as a common PDM data model. The OMG PLM Services 2.0 standard, in turn, provides SOA interfaces for the manipulation of PDM data represented in the STEP format.

The convergence to a common PDM format, and the standardization of the SOA interface for the access of PDM data brings many benefits. It allows Original Equipment Makers (OEMs) to exchange data with their supplier networks using a common data model, and the standardization of the SOA access interfaces allows for the data exchange process to be automated using the BPEL. The automation of the product data exchange process brings the following benefits:

- ▶ Reduces errors that were previously induced by manual processes
- ▶ Provides the automatic application of data translation rules, such as the mapping of part numbers in the case where more than one part number convention is used
- ▶ Shortens the design life cycle and the time to market

In addition to the application in the product data exchange domain, there are several other uses of SOA in the area of PDM. Part 3, “Overview of working examples” on page 175, demonstrates a scenario called Product Data Structure Consolidation. This scenario is common in the automotive and aerospace industries. The scenario assumes that a given product is stored in more than one PDM system. This occurs often because manufacturers frequently use different tools to design the surface of a product and other components such as a power train or engine. This is because some tools are best suited to design surfaces while other tools are based on parametric technologies. These parametric technologies are better suited to produce many instances of a part with different parameters, such as a collection of gears with different diameters, as in a power train. In another example, it could be that an airplane maker subcontracts the manufacturing of several sections of an airplane, and as a result, the airplane is split among different PDM systems. In both scenarios the manufacturer ends up with the challenge of having to consolidate the data from several PDM systems to get a complete picture of the car or airplane.

In Part 3, “Overview of working examples” on page 175, we demonstrate how to implement a Product Data Structure scenario by consolidating the PDM data that is stored in two different PDM systems, and producing a unified product data structure. We used SOA in this example not only to implement the BPEL workflow that makes the consolidation, but also to provide a service acting as a real PDM system that has the entire product. What the service is doing is



providing a facade to the process that consolidates the data contained in more than one system. This is a typical example of SOA applied to data federation, or *data as a service*. In Part 3, “Overview of working examples” on page 175, we consolidate the data from two servers, but SOA is capable of aggregating data from a large number of sources, and it is possible to employ runtime decisions based on policies as to which data sources must be used.

### 2.3.3 Simulation Data Management

Simulation Data Management is a discipline that covers handling of the input and output data associated with simulation processes. As the majority of the design processes today rely on digital mock-up technology (DMU), manufacturers can benefit from the possibility of obtaining key aspects before a physical prototype is built. Aspects such as the fuel efficiency of a car, aerodynamics of an airplane, or weight of a turbine can be obtained from a DMU model using simulation techniques. Manufacturers rely on this technology to ensure that the future product complies with design requirements, compliance rules, and so forth. This technology has become so pervasive that the development of a large product, such as car or an airplane, requires a huge number of simulations, which have to be repeated whenever an engineering change occurs. The result is the generation of terabytes of simulation data that has to be managed, versioned, and associated with a specific design change.

This is where the application of SOA technology can help. In step 3 of the VDA ECM depicted above, we see an activity labeled “Technical Analysis of the Engineering Change Request.” We have mentioned how a BPEL-based SOA process automates the ECM process. We now also have the ability to implement an automated SOA process that can issue simulation requests to simulation applications, interpret the results, and take the appropriate actions based on a pass/no pass decision. All of this is based on the simulation results.

Of course, such high level business process automation can only be achieved if suitable standards are used for the representation of a simulation data model and access interfaces. Such standards, such as the Simulation PDM (SimPDM) standard proposed by the PROSTEP iViP association, are now achieving maturity. Although the application of this standard is outside of the scope of this Redbooks publication, we can expect that the emerging standards provide huge savings and increased efficiency with the integration of simulation and ECM processes.

## 2.3.4 Manufacturing Data Management

Manufacturing data management relates to the translation of the PDS into a Bill of Materials (BOM) that is used in manufacturing systems for inventory control, procurement, and so forth. In Part 3, “Overview of working examples” on page 175, we demonstrate this ability in transforming a PDS, represented in OMG PLM Service 2.0 format, into a BOM that is then pushed into an Enterprise Resource Planning (ERP) system. While we have used the OMG PLM Services representation in our example, the SOA technology could also be used to transform the BOM into any other format, such as the OAGIS BOM specification.

In addition to the ability to transform a PDS into a BOM, or to produce BOM representations in different standard formats, SOA technology also plays a key role in keeping the BOM stored in the ERP system synchronized with the PDS kept in the PDM system. This synchronization is important, as changes occur frequently. We can use the SOA-enabled Engineering Change Management process to automate the process by which changes are made and all relevant systems updated. In Part 3, “Overview of working examples” on page 175, we demonstrate this ability by providing a SOA-driven process that implements the VDA ECM process, using the OMG PLM Services 2.0 standard, and makes it possible to create an Engineering Change Notice (ECN) in the ERP system automatically and to synchronize the BOM in response to a change in the PDS of the PDM system.

This scenario clearly illustrates many positive aspects of the SOA technology:

- ▶ Business process automation
- ▶ Data model translation
- ▶ Application integration

## 2.3.5 Service Data Management

The Service Data Management discipline relates to the management of product data of specific product instances, typically after the product has been manufactured and sold. For example, an airplane has a long life span, and must be serviced frequently for compliance purposes. As the airplane is serviced, it may require changes to its constituent parts. For example, in some airports the original parts may have to be substituted by equivalents. And in some cases entire sections are refurbished, and so forth. The Service Data Management discipline deals with the tracking of the Service BOM, the list of constituent parts for specific instances of an asset, as well as its maintenance history.

This discipline is also tied to the Engineering Change Management process. For example, suppose that a manufacturer detects a possible flaw given the results of specific compliance simulations. It is now necessary to make a recall and find all instances of use of the given part.

Another interesting application of SOA to the SLM domain is related to using BPM tools that provide business intelligence, as discussed in Part 2, “Scenarios and patterns” on page 63. In this case, we can instrument the Engineering Change processes to produce events that can be used as KPIs. The benefits of this technology are enormous. For example, suppose that every time that a part needs replacement, the SLM application emits an event to the part manufacturer. The part manufacturer can then use business even correlation techniques in order to determine if the failure rate is above the tolerance threshold, in which case an engineering change management process may be triggered to redesign the defective part. The SOA-enabled ECM process then runs all pertinent simulations of the newly designed part. Eventually, the design is approved, the BOM is changed in the ERP system, the new part is manufactured, and the SOA-enabled ECM process notifies all users of that part of the change. This example shows how a SOA-enabled ECM process helps automate all stages of the product life cycle, and also provides valuable business process intelligence in the process.

In Part 3, “Overview of working examples” on page 175, we discuss a business scenario in which all these features are used, and we give a real example of a SOA-enabled ECM based on open industry standards.



# Product Lifecycle Management Standards overview

Broadly speaking, Product Lifecycle Management (PLM) deals with the creation, modification, and exchange of product information throughout a product's life cycle. The information systems that make up PLM used to be predominantly Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), Computer-Aided Engineering (CAE), and Product Data Management (PDM). While this still continues to be the case, the scope and definition of PLM are expanding and maturing to meet the demands of an increasingly complex network of partners spread globally and bound together by common financial objectives. These partners have to collaborate by sharing and exchanging product information, and integrating their engineering and business decision support systems. They need flexibility to implement engineering and business processes that can change rapidly as their business environments change.

While business leaders recognize the need for wider collaboration and integration, they also realize that they are far from reaching their objectives. One cause for this predicament is that, until recently, information technology tools and strategies were not up to the challenge of sharing information across such a complex network in a flexible and robust way.



This is about to change due to the convergence of the following three important developments:

- ▶ Maturity of standardized product data and metadata models, and standardized engineering and business processes
- ▶ Emergence of service-oriented architecture (SOA) for information sharing
- ▶ Availability of robust middleware to implement them

A PLM infrastructure that combines the fruits of these three developments is bringing us closer to realizing the dreams of business and technical integration, and partner collaboration.

At this stage, it is useful to look at the distinction between information exchange and information sharing. It can be illustrated using a simple banking example. Table 3-1 shows how financial data can be exchanged and shared by a bank and a customer. In the Data Exchange scenario, the bank has the master data in its information system. It prints a copy at a given point in time, in a human readable format, and physically mails it to the customer. In the Data Sharing scenario, the customer initiates the process by requesting his current account balance through a telephone connection. The bank then provides that information, in near real-time, by looking up this information in its master database.

*Table 3-1 Data exchange versus data sharing*

Data Exchange Characteristics	Data Sharing Characteristics
<ul style="list-style-type: none"> <li>▶ Initiated by data originator</li> <li>▶ Transformed into a neutral format</li> <li>▶ Content determined by discrete event in time</li> <li>▶ Redundant copy of data created</li> </ul>	<ul style="list-style-type: none"> <li>▶ Initiated by data receiver</li> <li>▶ Data on demand</li> <li>▶ Data access levels embedded in protocol</li> <li>▶ Appears as single data source</li> <li>▶ Read (real-time) and update capabilities</li> </ul>
	

This simple banking example can be extended to product information exchange and sharing. The major characteristics that distinguish product information exchange and product information sharing are the centrality of the information and the ownership of that information. In the Data Exchange scenario, the engineering information system that maintains a master copy of the product information makes a copy of that information and sends it to the user.

This is the scenario that is in play whenever a technical drawing or a geometric model of a product (either in a proprietary format or in a standardized model) is exchanged. Any person or information system that imports such an exchange file effectively owns that information, which is not synchronized with the master copy of the information maintained in the original software system. As both parties have the same information only at the time of the information exchange, it is safe to assume that in the future, the systems (or persons) that exchanged such information will have different versions of it.

In the Data Sharing scenario, on the other hand, there is a control over the ownership of the information. The master copy of the information can be accessed and revised under controlled circumstances. Ideally, this is the scenario that one would like to strive for. Initial attempts to achieve this goal were not successful, and they taught us some important lessons. One of the lessons is that we have to pay attention to the granularity of the product information.

There are detailed product technical data, such as technical drawing and geometric models, which are exchanged and shared by a limited number of engineering specialists. Then there is metadata about the product (such as part number, bill of material (BOM), authorization, effectivity, material, provenance, and so forth) that are shared by a much wider group, often with people and systems outside the traditional engineering departments. Engineering tools and systems such as CAD, CAM, and CAE handle the bulk of the detailed product data, while enterprise-wide tools and systems such as PDM, Enterprise Resource Planning (ERP), and Customer Relationship Management (CRM) deal with the coarser metadata. Interaction between engineering and business information systems often takes place in the context of the metadata.

Several standards development organizations have responded to this realization by issuing open standards for product data and metadata. They are also standardizing elements of engineering and business processes (such as engineering change management) that use the product information. The information technology industry is providing an infrastructure (independent of any specific application) based on open standards to share such information over the Internet in the form of Web Services.

This chapter is organized as follows:

- ▶ Section 3.1, “Standardized product data and metadata models” on page 34
- ▶ Section 3.2, “Architecture of OMG PLM Services” on page 45
- ▶ Section 3.3, “Summary” on page 61

This section summarizes the state of standardization and provides justification for Object Management Group (OMG) PLM Services.

## 3.1 Standardized product data and metadata models

In the opening section of this chapter, we mentioned that open standards for product data and metadata play a important role in an integration framework for PLM. We describe both of them in this section.

### 3.1.1 Product data

Standardizing the specification and verification of product geometry data has a long and successful history. These standards originated in the pre-information age (circa 1930s). They have undergone modernization, primarily through the development of mathematical theories and computational techniques, as industry moved in to the information age.

The responsibility for developing and maintaining these standards rests with national professional societies, such as the American Society of Mechanical Engineers (ASME) in the United States, and with the International Organization for Standardization (ISO) internationally. These standards cover the bulk of the detailed geometric data that is shared and exchanged across the enterprise and partner network. Examples of such geometric data can be seen in Figure 3-1 on page 35 and Figure 3-2 on page 36. The syntax and semantics of various symbols and annotations found in these figures are defined using prose, graphics, and mathematics in the standards of ASME and ISO.

It is possible to exchange the standardized product data found in Figure 3-1 on page 35 and Figure 3-2 on page 36 as graphics and image files. In fact, it is the most common practice today. An open standard for Portable Document Format (PDF) has been issued by ISO for exchanging and archiving two-dimensional graphics (such as in Figure 3-1 on page 35). ISO is currently working on an open standard for three-dimensional graphics (such as in Figure 3-2 on page 36) so that they can be manipulated to provide three-dimensional interactivity.

The graphics can also be structured to selectively turn some elements on or off. The underlying three-dimensional part geometry can be sectioned and selected distances queried (measured). While these files convey the data syntax faithfully in the form of symbols and stylized indications, they require a human being at the receiving end to interpret the semantics of these technical specifications.



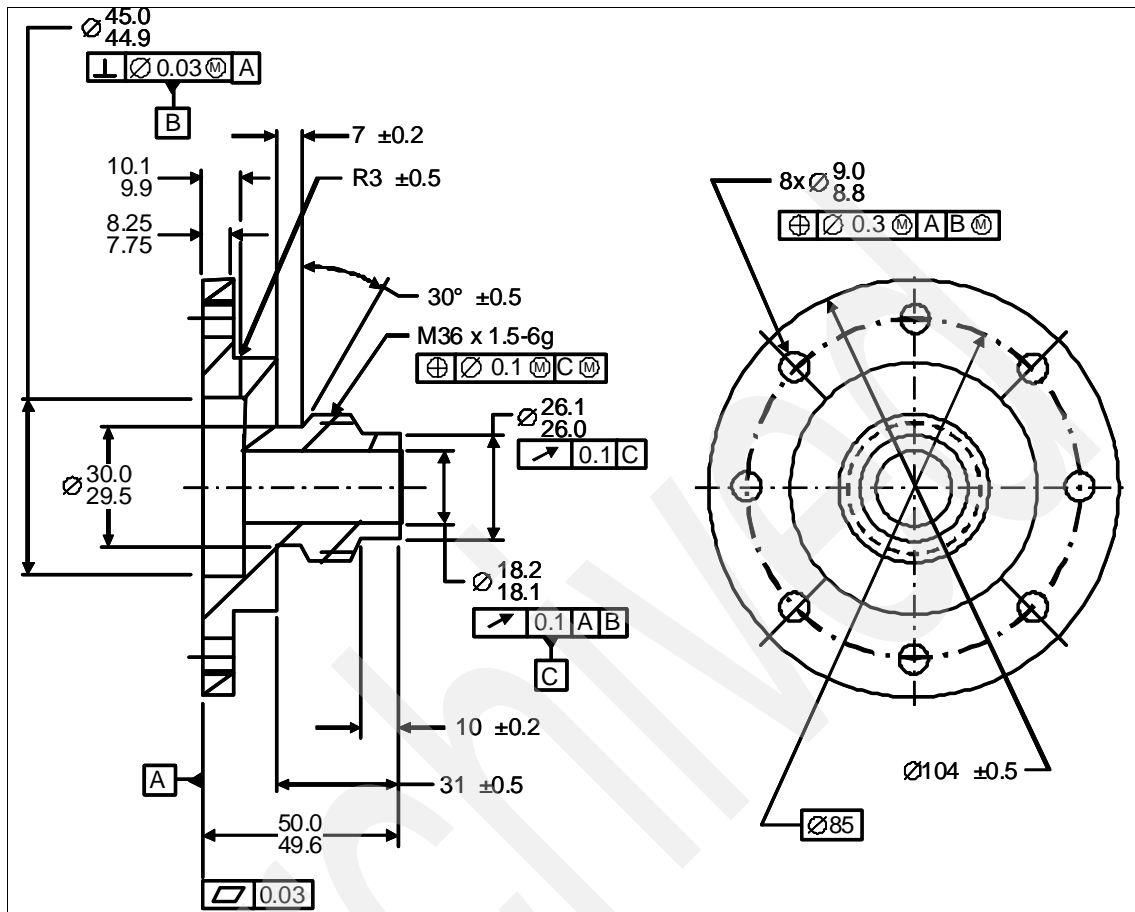
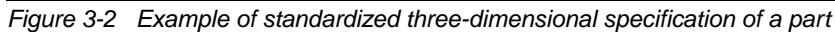


Figure 3-1 Industrial drawing with dimensioning and tolerancing annotations on projected views of a part (dimensions are in mm)

To alleviate this problem, ISO has also developed semantic standards in the ISO 10303 series. Commonly known by its acronym, STEP (Standard for the Exchange of Product data model) is a rich family of open standards for product data (and metadata, as we see in the next subsection). It is a product of the information age. It takes a rigorous approach to semantic standards by standardizing the data models using schemas, and complements more popular modeling and markup languages such as Unified Modeling Language (UML) and eXtensible Markup Language (XML).

STEP, and its predecessor, the Initial Graphics Exchange Specification (IGES), can be used as open standard formats to exchange three-dimensional geometric product data among different PLM systems. In actual practice, people use

**ISO Copyright notice:** “The terms and definitions taken from ISO 16792:2006 Technical product documentation - Digital product definition data practices, Figure 3-2, are reproduced with the permission of the International Organization for Standardization, ISO. This standard can be obtained from any ISO member and from the Web site of ISO Central Secretariat at the following address: <http://www.iso.org>. Copyright remains with ISO.”



### 3.1.2 Product metadata

Metadata is data about data. In our context, it is a sparse description of detailed data contained in files or technical data packages. Traditionally, PDM systems defined and managed the metadata. This left the detailed geometric data and so forth to the CAD, CAM, and CAE systems, as seen in the way ISO has classified standardized product data models in, for example, STEP AP (Application Protocol) 214. Table 3-2 describes the contents of five conformance classes (CC) of STEP AP214. CC1 and CC2 cover the part and assembly geometry data. CC6 and CC8 cover metadata needed by PDM systems that can treat geometric model data as files.

There are several other conformance classes, including CC21, that can be used for engineering change management (ECM) process, but the four mentioned in the previous paragraph are the ones that are currently supported by CAD and PDM vendors. Note that STEP AP214 itself does not use the terms data and metadata. We use these terms to clarify the current and future developments in product-related information standards.

Table 3-2 details the association of data with engineering objects and metadata with business objects. Engineering objects remain firmly in the domain of engineers and their tools (such as CAD, CAM, CAE systems). Creation, modification, and interpretation of these objects require engineering expertise. On the other hand, the business objects deal with items such as part numbers, BOMs, and administrative information that can be understood and used by both engineers and non-engineers who help run the business.

Table 3-2 Standardized data, metadata, and business processes in STEP AP 214

	Class	Description	Remarks
<b>Data</b>  (Engineering objects)	CC1	Component design with 3D shape representation.	Covers 3D geometry of single parts, including wire-frame, surface, and solid models.
	CC2	Assembly design with 3D shape representation.	Covers 3D geometry of assemblies of parts, including assembly and model structure.
<b>Metadata</b>  (Business objects)	CC6	PDM without shape representation.	Covers PDM systems that manage geometric models as files. Covers administrative data of parts, assemblies, documents, and models.
	CC8	Configuration controlled design without shape representation.	Covers CC6, with additional requirements for product configuration control.
(Business processes)	CC21	Specification control & process planning.	Covers CC8, with additional requirements for process planning. Can be used for engineering change management.

AP214 is not the only STEP protocol that deals with such product metadata. Figure 3-3 illustrates an intersection of the following four different STEP APs that contain a common, core PDM Schema:

- ▶ AP203  
This application protocol deals with configuration controlled 3D design of mechanical parts and assemblies.
- ▶ AP212  
This application protocol deals with electrotechnical design and installation.
- ▶ AP214  
This application protocol deals with core data for automotive mechanical design processes
- ▶ AP232  
This application protocol deals with technical data packaging core information and exchange.

The STEP PDM Schema as a common subset is maintained by two prominent standards consortia, PDES Inc. and ProSTEP iViP.

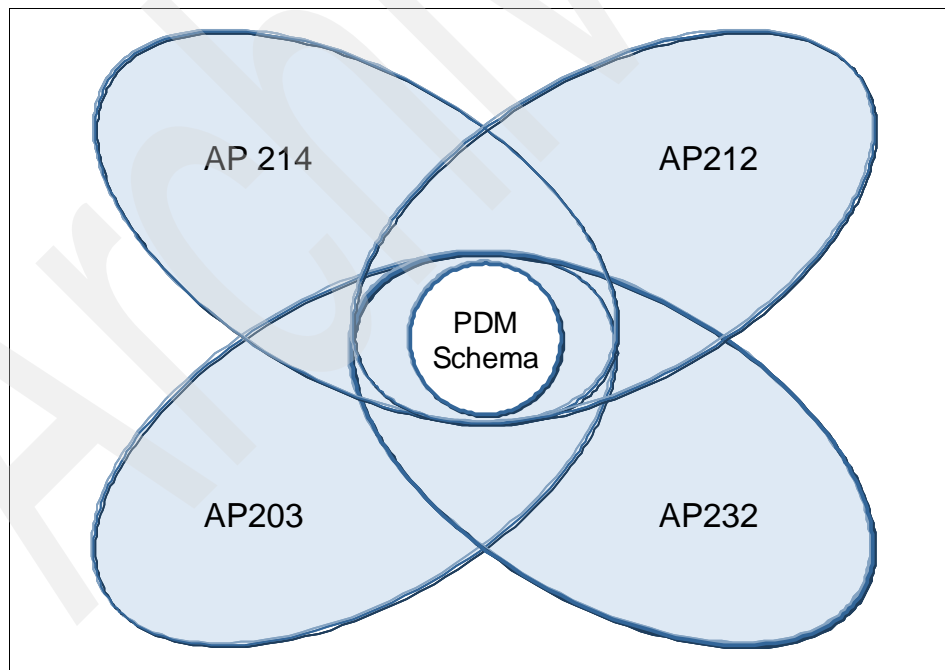


Figure 3-3 PDM Schema culled from the intersection of four STEP APs

The STEP PDM Schema contains some of the most important standardized product metadata models that came out of the ISO STEP efforts. In the STEP PDM Schema, a general product can be conceptually interpreted either as a part or as a document. In this way, parts and documents are managed in a consistent and parallel fashion.

The units of functionality for parts and documents covered by the STEP PDM Schemas include identification, versioning, product structures (including transformations), approvals and authorization, project, work order, work request, effectivities, classification and properties. These are further elaborated in Table 3-3.

*Table 3-3 Units of functionality covered by the STEP PDM Schema*

<b>Units of Functionality</b>	<b>Content</b>
Part identification	Part as Product <ul style="list-style-type: none"> <li>▶ Product master identification</li> <li>▶ Context information</li> <li>▶ Type classification</li> </ul>
Specific part type classification	Classification of parts and managed documents
Part properties	General part properties External part shape External geometric model structure Relative orientation and location of related geometric models
Part structure and relationships	Explicit assembly BOM Multilevel assembly digital mock-up Different views on assembly structure Relating part shape properties to product structure Other relationships between parts
Document identification	Document as Product <ul style="list-style-type: none"> <li>▶ Document master identification</li> <li>▶ Context information</li> <li>▶ Type classification</li> </ul>
Specific document type classification	Product-related product category and product category relationship
External files	External file identification
Relationship between documents and constituent files	Product definition with associated documents

Units of Functionality	Content
Document and file properties	Product definition or document representation Document content property Document creation property Document size property Document source property Additional document properties Document type classification
Document and file association with product data	Document reference External file reference Constrained document or file reference
Document and file relationships	Sequence relationships between document versions Relationships between document representations Relationships between external files
Alias identification	Associating with additional identifier
Authorization	Organization and persons Approval Dates, times, and event references Security classification Certification
Configuration and effectivity information	Configuration identification Configuration composition management General validity period
Engineering change and work management	Request for work Work order and work definition Project identification Contract identification
Measure and units	Measures with unit specification Unit definition

Two-hundred and ten entities are defined to compose these units of functionality in the STEP PDM Schema. The names of these entities are listed in Table 3-4 on page 41 and Table 3-5 on page 42. Each of these 210 entities is defined in greater detail using the EXPRESS schema language. A quick perusal of Table 3-3 on page 39, Table 3-4 on page 41, and Table 3-5 on page 42 gives the reader an appreciation of the richness of the metadata defined by the STEP PDM Schema. Table 3-6 on page 43 shows the attributes of just one such entity called PRODUCT. In practice, an actual schema instance can contain a lot more entities and covers much more metadata.

Table 3-4 Entities defined in the STEP PDM Schema: Part I

action	cartesian_transformation_operator_3d
action_assignment	certification
action_directive	certification_assignment
action_method	certification_type
action_property	characterized_object
action_property_representation	configuration_design
action_relationship	configuration_effectivity
action_request_assignment	configuration_item
action_request_solution	context_dependent_shape_representation
action_request_status	context_dependent_unit
action_status	contract
address	contract_assignment
alternate_product_relationship	contract_type
amount_of_substance_measure_with_unit	conversion_based_unit
amount_of_substance_unit	coordinated_universal_time_offset
application_context	date
application_context_element	date_and_time
application_context_relationship	date_and_time_assignment
application_protocol_definition	date_assignment
applied_action_assignment	date_role
applied_action_request_assignment	date_time_role
applied_approval_assignment	dated_effectivity
applied_certification_assignment	derived_unit
applied_contract_assignment	derived_unit_element
applied_date_and_time_assignment	description_attribute
applied_date_assignment	descriptive_representation_item
applied_document_reference	dimensional_exponents
applied_document_usage_constraint_assignment	directed_action
applied_effectivity_assignment	direction
applied_event_occurrence_assignment	document
applied_external_identification_assignment	document_file
applied_identification_assignment	document_product_association
applied_organization_assignment	document_product_equivalence
applied_organizational_project_assignment	document_reference
applied_person_and_organization_assignment	document_relationship
applied_security_classification_assignment	document_representation_type
approval	document_type
approval_assignment	document_usage_constraint
approval_date_time	document_usage_constraint_assignment
approval_person_organization	document_usage_role
approval_relationship	effectivity
approval_role	effectivity_assignment
approval_status	effectivity_relationship
area_measure_with_unit	electric_current_measure_with_unit
area_unit	electric_current_unit
assembly_component_usage	event_occurrence
assembly_component_usage_substitute	event_occurrence_assignment
axis2_placement_2d	event_occurrence_role
axis2_placement_3d	executed_action
calendar_date	external_identification_assignment
cartesian_point	external_source
cartesian_transformation_operator	founded_item
cartesian_transformation_operator_2d	functionally_defined_transformation

Table 3-5 Entities defined in the STEP PDM Schema: Part II

general_property	product_definition_context
general_property_association	product_definition_context_association
general_property_relationship	product_definition_context_role
geometric_representation_context	product_definition_effectivity
geometric_representation_item	product_definition_formation
global_uncertainty_assigned_context	product_definition_formation_relationship
global_unit_assigned_context	product_definition_formation_with_specified_source
id_attribute	product_definition_relationship
identification_assignment	product_definition_shape
identification_role	product_definition_usage
item_defined_transformation	product_definition_with_associated_documents
length_measure_with_unit	product_related_product_category
length_unit	promissory_usage_occurrence
local_time	property_definition
lot_effectivity	property_definition_representation
luminous_intensity_measure_with_unit	quantified_assembly_component_usage
luminous_intensity_unit	ratio_measure_with_unit
make_from_usage_option	ratio_unit
mapped_item	relative_event_occurrence
mass_measure_with_unit	representation
mass_unit	representation_context
measure_representation_item	representation_item
measure_with_unit	representation_map
name_attribute	representation_relationship
named_unit	representation_relationship_with_transformation
next_assembly_usage_occurrence	role_association
object_role	security_classification
organization	security_classification_assignment
organization_assignment	security_classification_level
organization_relationship	serial_numbered_effectivity
organization_role	shape_aspect
organizational_address	shape_aspect_relationship
organizational_project	shape_definition_representation
organizational_project_assignment	shape_representation
organizational_project_relationship	shape_representation_relationship
organizational_project_role	si_unit
person	solid_angle_measure_with_unit
person_and_organization	solid_angle_unit
person_and_organization_assignment	specified_higher_usage_occurrence
person_and_organization_role	thermodynamic_temperature_measure_with_unit
personal_address	thermodynamic_temperature_unit
placement	time_interval
plane_angle_measure_with_unit	time_interval_based_effectivity
plane_angle_unit	time_interval_with_bounds
point	time_measure_with_unit
product	time_unit
product_category	uncertainty_measure_with_unit
product_category_relationship	value_representation_item
product_concept	vector
product_concept_context	versioned_action_request
product_context	volume_measure_with_unit
product_definition	volume_unit



Table 3-6 Attributes of the PRODUCT entity in the STEP PDM Schema

ENTITY product	Attribute Population	Remarks
id	type: identifier = string product number identification	The id attribute stores the unique base identifier for a product, the product number.
name	type: label = string	The name attribute stores the nomenclature, or common name, of the product.
description	type: text = string	The description attribute must contain an expanded name or description of the product.
frame_of_reference	type: entity = product_context	Context information is stored in the frame_of_reference attribute. All products must be found in some product_context.

As noted earlier, all data and metadata models of STEP are defined using the EXPRESS language. Although EXPRESS is a powerful data modeling language, it is relatively unknown to most programmers. Moreover, STEP data (for example, an instance population of an EXPRESS schema) are typically exchanged as files using an ASCII character-based syntax defined in ISO 10303-21 (also known as STEP Part 21). The Part 21 syntax, though effective for the task at hand, lacks extensibility, can be hard for humans to read, and perhaps most limiting, is computer interpretable by a relatively small number of software development toolkits that support STEP. These deficiencies are remedied by providing links between STEP models, and XML and UML models.

In order to capitalize on XML's popularity and flexibility, and to accelerate STEP's adoption and deployment, ISO has developed a standard for representing EXPRESS schemas and instance populations in XML. This standard, ISO 10303-28, *Industrial automation systems and integration -- Product data representation and exchange -- Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas*, is available for purchase from the following Web page:

[http://www.iso.org/iso/iso\\_catalogue.htm](http://www.iso.org/iso/iso_catalogue.htm)

ISO 10303-28 not only enables developers to use low-cost, ubiquitous XML software tools to implement file-based exchange and visualization of STEP instances, but also facilitates the use of STEP information in emerging areas such as XML-based Web services.

The ISO STEP community developed another standard, ISO 10303-25, *Industrial automation systems and integration -- Product data representation and exchange -- Part 25: Implementation methods: EXPRESS to XMI binding* (also known as Part 25), for transforming EXPRESS schemas into UML models. This standard is available from the following Web page:

[http://www.iso.org/iso/iso\\_catalogue.htm](http://www.iso.org/iso/iso_catalogue.htm)

Part 25 enables developers to use their familiar UML tools to see the contents of STEP (EXPRESS) schemas and to specify relationships between STEP information models and the other UML models they use. A Part 25 mapping from EXPRESS schema to the XML Metadata Interchange (XMI) format could produce a UML class diagram.

The STEP family of standards is quite extensive and is undergoing constant development and enlargement. This Redbooks publication touches upon only those members of this family that are relevant to the product information sharing service described in this chapter.

The ISO STEP community is not the only organization that has been developing standardized product metadata models. The growth and popularity of the Internet has resulted in several other open and nimble standards development organizations. This includes the following organizations developing and supporting standards that overlap and extend the work of ISO:

- ▶ Object Management Group (OMG)
- ▶ Organization for the Advancement of Structured Information Standards (OASIS)
- ▶ Open Applications Group Inc. (OAGi)

One particular development that is relevant is the emergence and use of Business Object Documents (BODs) as Open Applications Group Integration Specifications (OAGIS) from OAGi. For the purpose of this chapter, however, we confine our description to those standards that are relevant to OMG's PLM services.

## 3.2 Architecture of OMG PLM Services

The product information sharing service described in this Redbooks publication is a software system built on three levels of architecture:

- ▶ Model driven architecture

This architecture, now standardized by the OMG, is applicable to all distributed software system development. See 3.2.1, “OMG Model Driven Architecture” on page 45.

- ▶ PLM Services standard

This architecture, now standardized by the OMG, is built on other standards. See 3.2.2, “OMG PLM Services architecture” on page 47.

- ▶ Industrial software system for sharing product metadata

This system, not standardized by the OMG, is architected using these two standards and commercially available middleware. See 3.2.3, “Architecture of OpenPDM” on page 60.

### 3.2.1 OMG Model Driven Architecture

We are interested in standardized web services that support a service-oriented architecture. One of the earliest efforts in this field was spearheaded by the OMG before the Internet became a commercial success. Starting in the early 1990s, OMG standardized the object request broker (ORB) and a suite of object services. This work was guided by its Object Management Architecture (OMA), which provided a framework for distributed software systems. It also delivered the Common Object Request Broker Architecture (CORBA) as a part of that framework. CORBA was an architecture designed to enable software components written in multiple computer languages and running on multiple computers on different hardware platforms to interoperate. In fact, hard lessons learned in the design and implementation of CORBA provided valuable input to the evolution of successors to this architecture.

The OMA framework identifies types of components that are combined to make up a distributed system. Together with CORBA, this framework specifies types of connectors and the rules for their use.

In 2001, OMG adopted a second framework called the Model Driven Architecture (MDA). It is an approach to using models in various levels of abstraction in software development.

MDA has three important viewpoints and associated models:

- Computation independent viewpoint

This viewpoint focuses on the environment of a software system, and the requirements for the system. The details of the structure and processing of the system are hidden or as yet undetermined.

A computation independent model (CIM) is a view of a system from this computation independent viewpoint. A CIM does not show details of the structure of software systems. A CIM is sometimes called a domain model. A vocabulary that is familiar to the practitioners of the domain in question is used in its specification.

- Platform independent viewpoint

This viewpoint focuses on the operation of a software system while hiding the details necessary for a particular implementation platform. A platform independent view shows the part of the complete specification that does not change from one implementation platform to another. A platform independent view may use a general purpose modeling language, or a language specific to the area in which the system is used.

A platform independent model (PIM) is a view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type.

- Platform specific viewpoint

This viewpoint combines the platform independent viewpoint with an additional focus on the details of the use of a specific platform by a software system.

A platform specific model (PSM) is a view of a system from the implementation platform specific viewpoint. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

In the model hierarchy described above, one model is transformed into another by a model transformation. For example, a PIM and other relevant information can be combined to produce a PSM. Figure 3-4 on page 47 illustrates the three models of OMG's MDA and their transformation from more abstract models to more concrete models. Note that MDA is applicable to software development in several application domains, such as manufacturing, finance, telecom, and health care. In the manufacturing industry, this standardized hierarchical-model-driven architecture guided the development of the standardized PLM Services described in Section 3.2.2, "OMG PLM Services architecture" on page 47.

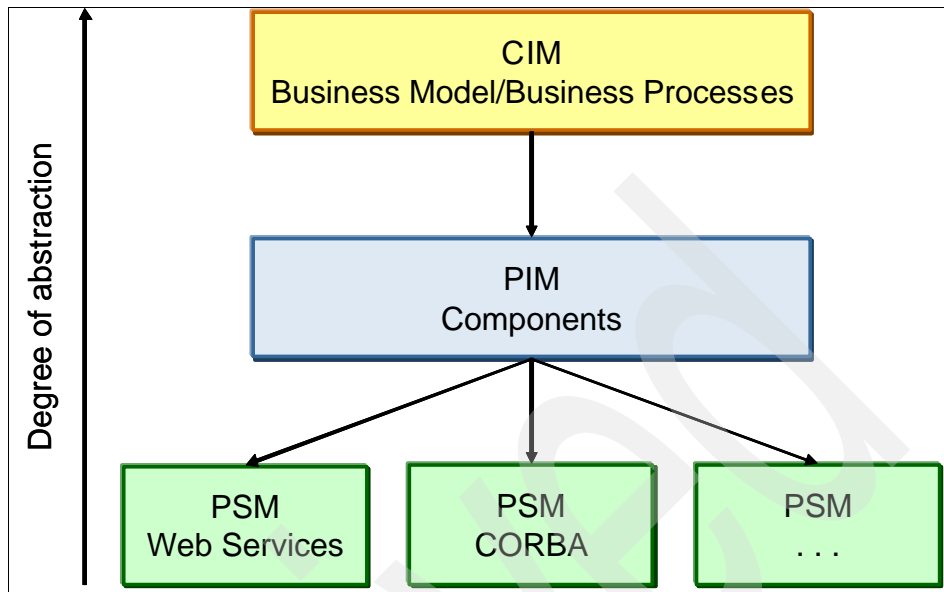
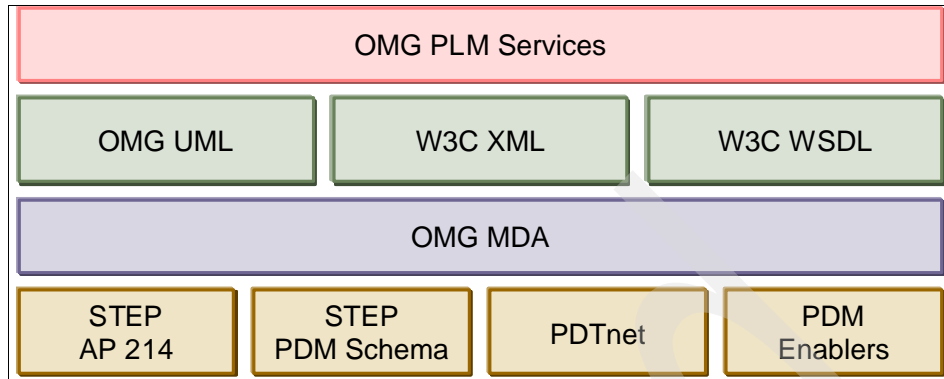


Figure 3-4 OMG's Model Driven Architecture

### 3.2.2 OMG PLM Services architecture

PLM Services is an OMG standard specification developed by an industrial consortium under the umbrella of the ProSTEP iViP Association. Currently, OMG has issued the PLM Services 2.0 specification, a superset of its earlier PLM Services 1.0 specification, supersedes PLM Services 1.0. This standard specification is architected so that it is based on other standards, as shown in Figure 3-5 on page 48. Several of the base standards at its bottom layer, such as STEP AP214 and STEP PDM Schemas, are described in Section 3.1.2, "Product metadata" on page 37. These are ISO-standardized metadata models of industrial products. The reader might be quite familiar with these ISO standards. We also see PDM Enablers and PDTnet as other standards and specifications in this bottom layer. The reader might be less familiar with these. The following sections describe these lesser-known standards and specifications briefly.



*Figure 3-5 OMG PLM Services is architected as a standard specification based on other standards*

### **PDM Enablers**

This OMG standard specification is intended to provide standardized interfaces to PDM systems, or other software systems providing similar services, from other manufacturing software systems. Extensive in its scope and contents, the PDM Enablers specification is organized into twelve interface description language (IDL) modules, as shown in Table 3-7 on page 49. These modules provide various UML diagrams and CORBA interfaces. In addition, the PDM Enablers specification provides mapping of product development processes (broken down into eleven sub-processes outlined in Table 3-8 on page 50) to these modules. A quick perusal of Table 3-7 on page 49 and Table 3-8 on page 50 reveal the amount of work that has gone into the OMG standardization of product metadata and product development process elements. The ISO STEP PDM Schema standards and the OMG PDM Enablers are different in purpose, scope, abstraction level, and operational characteristics. However, they have common and complementary components that could be used for a PLM Services standard.

Table 3-7 Modules in OMG PDM Enablers

	Module	Remarks
1	PdmResponsibility	Defines minimum interfaces for references to people and organizations as owners, contacts, or some other responsibility
2	PdmFoundation	Defines basic foundation classes for units of behavior such as manageable, lockable, revisionable, iterable, statable, and security classifiable
3	PdmFramework	Provides PDM-specific object and relationship classes. Collects basic units of behavior into higher level PDM-specific abstract objects such as ItemMaster, ItemRevision, and ItemIteration
4	PdmBaseline	Describes a collection of items and their relationships to enable their configuration to be reconstructed and audited
5	PdmViews	Provides framework and explicit classes for indicating what items and relationships apply or qualify in particular use cases
6	PdmDocumentManagement	Describes a conceptual model for document management such as check-in and check-out
7	PdmProductStructureDefinition	Defines parts and the BOM relationships between items for discrete manufacturing
8	PdmEffectivity	Provides a range in which an item revision, component, or other qualified item must be used in production
9	PdmChangeManagement	Provides classes for engineering change issue, engineering change request, engineering change order, and engineering change notice
10	PdmManufacturingImplementation	Describes objects for manufacturing process specification and relationships between process specification, items produced, manufacturing location, and engineering change orders
11	PdmConfigurationManagement	Extends the PdmProductStructureDefinition Provides the ability to define the features or specifications that distinguish the different configurations and the ability to define rules to select specific components based on the feature values
12	PdmSTEP	Provides interface to STEP import and export capabilities to read and write data conformant to specific APs such as AP203 and AP214

Table 3-8 OMG PDM Enablers' eleven sub-processes to support product development

	Sub-processes	Description
1	Develop strategic product plan	Determines product areas for business, market decisions, design and manufacturing responsibilities Projects cost and volumes
2	Develop product business plan	Determines approach to product design to meet cost and volume estimates Creates development plan with schedules and estimates
3	Develop product definition	Develops product general specifications, conceptual layout, mock-ups and visuals Identifies relationships to existing parts and manufacturing capabilities Establishes development plans and objectives Estimates costs of major components Manages existing parts and modules library
4	Define product marketing configuration and rules	Defines features and options Projects sales volumes and cost by options
5	Develop product design	Develops major component general specifications, major component layout Retrieves, incorporates and establishes reusable parts and major components Establishes design plan and record status Defines module and sub-assembly content with optional components Develops part design Select preferred features
6	Develop process design and procurement agreements	Develop sassembly process design and manufacturing facility plan Assesses manufacturing capabilities; determine part sourcing Procures machine tools, firm tools and services Develops production tooling design, part fabrication process design, and procurement contracts
7	Coordinate design change	Identifies design and process changes Processes external design changes Identifies implantation dependencies Notifies required team members and design changes across all products
8	Evaluate product design	Develop model and prototype of proposed design, and test plans and prototype configurations Runs tests and report results Analyzes test results



	Sub-processes	Description
9	Implement production changes	Plans for future manufacturing and product change implementation Generates material requirements for prototype Alerts@ change and restrict inventory and investment Builds prototype using production facilities Distributes prototype build documentation Schedules tool tryout Establishes change implementation schedule Generates material requirements for pilots Notifies and distributes updated manufacturing documentation
10	Develop product service methods	Develops service assembly procedures and diagnostic procedures
11	Develop service distribution plan	Creates part catalogs and service inventory plans

### PDTnet

PDTnet, also found at the bottom layer of Figure 3-5 on page 48, was a project undertaken by an industrial consortium under the umbrella of the ProSTEP iViP Association in the year 2000 to enable neutral, system-independent product data communication between automobile manufacturers and suppliers based on the ISO STEP AP214 standards and the emerging XML technologies. Three years later, the lessons learned from this project in using STEP product information standards and Internet technologies were documented and used as an input for the OMG PLM Services development. Of particular importance are the PDTnet queries. Table 3-9 lists some of these predefined queries.

Table 3-9 Predefined queries of PDTnet

	Query	Description
1	AliasIdentificationQuery	Returns alias information for a selected item
2	AlternativeSolutionQuery	Returns information about the alternative solutions of a product component
3	ApprovalQuery	Returns approval information for a selected element
4	ApprovalRelationshipQuery	Returns specified relationships of a selected element approval
5	AssemblyStructureQuery	Returns the assembly structure for a selected item
6	AssociatedDocumentQuery	Returns all documents associated with a selected item
7	AssociatedOrganisationQuery	Returns all organizations associated with a selected item

	Query	Description
8	ConfigurationQuery	Returns configuration information for a selected item
9	CreateProductStructure	Creates a product structure including its metadata under a selected target element
10	DateAndTimeQuery	Returns date and time information for a selected item or document
11	DocumentClassificationQuery	Returns the specific document classification for a selected document
12	DocumentPropertyQuery	Returns specified properties of a selected document
13	DocumentStructureQuery	Returns the subdocuments for a selected document
14	DownloadDigitalFile	Returns an URL that downloads the file
15	DownloadSetOfDigitalFiles	Gets a set of files as a package
16	EffectivityAssignmentQuery	Returns effectivity information for a selected element
17	ExternalFileQuery	Returns the external files referred by a selected document representation
18	GetProductStructure	Returns the assembly structure of a selected item and all its metadata
19	DocumentQuery	Returns document information for a selected item
20	IntiateOfflineUpload	Sends selected files to a target system by an offline data transfer
21	ItemClassificationQuery	Returns the classification for a selected item
22	ItemPropertyQuery	Returns the property values for a selected element
23	ItemQuery	Returns a selected item with specified content
24	ItemRelationshipQuery	Returns specified relationships of a selected item
25	ItemWhereUsed	Returns all parent nodes of a selected item
26	OrganizationQuery	Returns organization information for a selected item
27	ProductStructureQuery	Returns product structure
28	SearchInDesignSpace	Returns a list of times that are positioned within a specified design space of a selected item
29	StartNodeQuery	Returns a list of items that can be defined by their part number, part name, and version number

	Query	Description
30	SubscriptionListAcknowledgeContent	Acknowledges changes on elements of the subscription list
31	SubscriptionListAddContent	Adds new elements to the subscription list
32	SubscriptionListGetContent	Returns the current content and state of the subscription list
33	SubscriptionListRemovalContent	Removes elements from the subscription list
34	UploadDigitalFile	Uploads a file to a server
35	UploadSetOfDigitalFiles	Uploads a set of files to a server

The reader has now been exposed to the confusing array of standards and specifications that contributed to the OMG PLM Services. The visual timeline in Figure 3-6 helps to clarify the picture and understand their evolution.

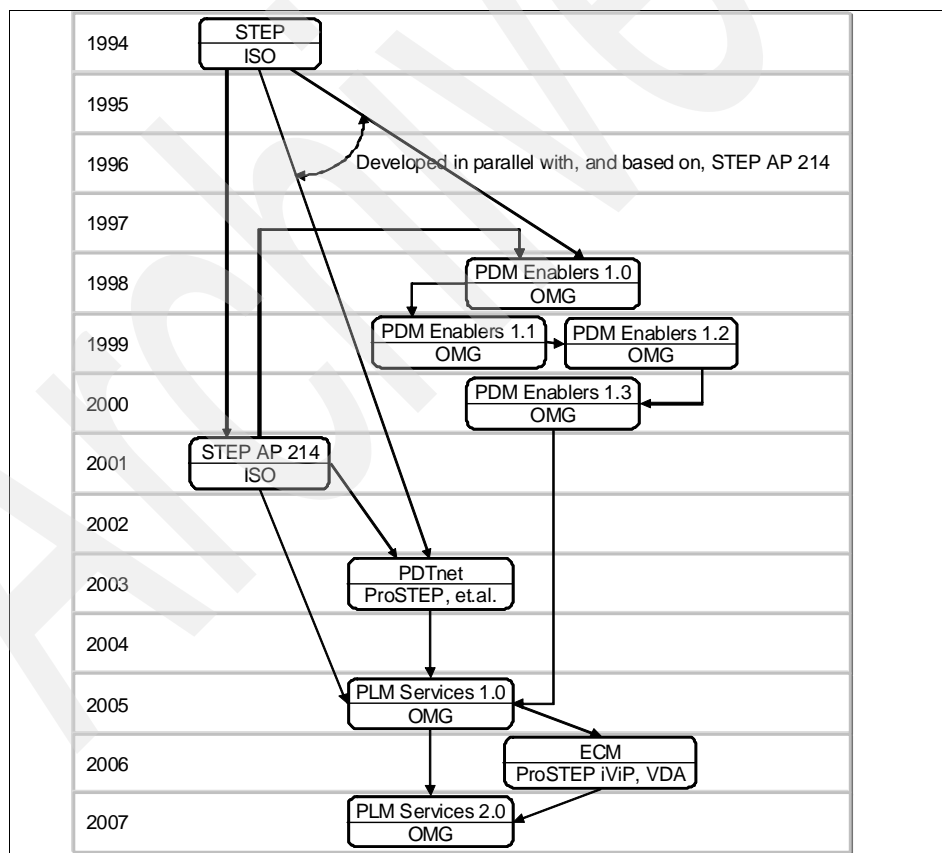


Figure 3-6 A timeline of development of OMG PLM Services and related standards

These developments contributed to the overall architecture of OMG PLM Services as shown in Figure 3-7.

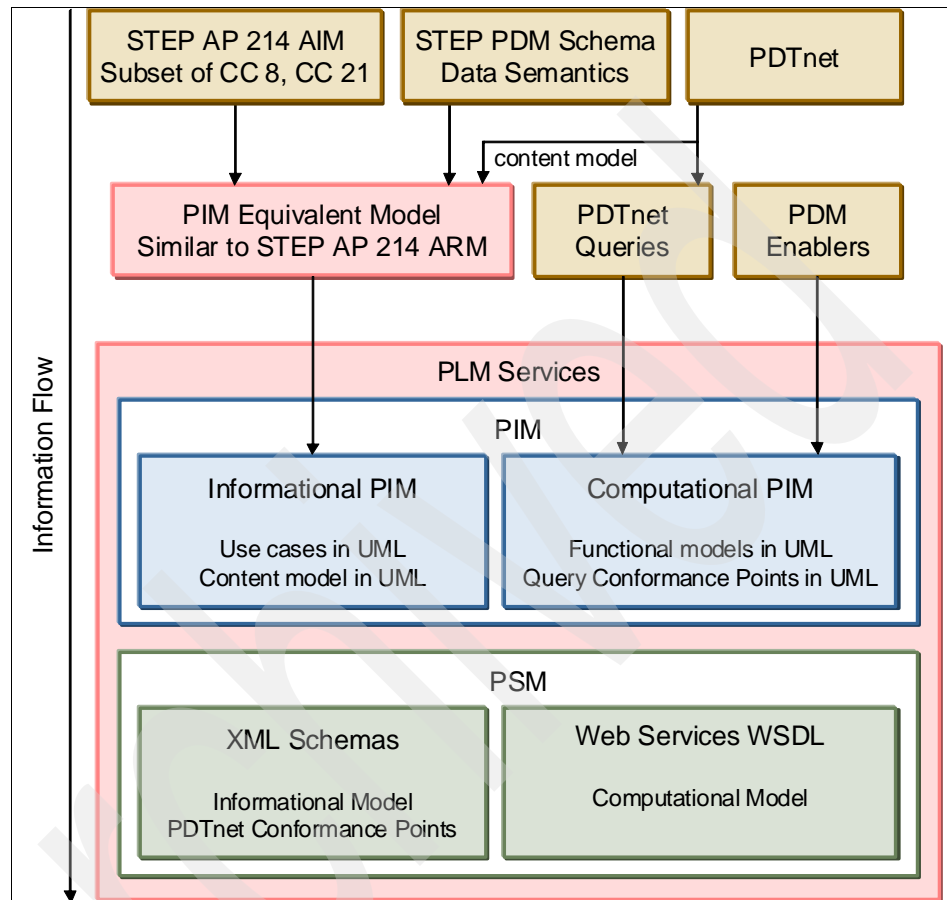


Figure 3-7 Derivation and architecture of OMG PLM Services

Recall that the OMG's MDA demands a strict separation between PIM and PSM. In some sense, the PIM is more important because it can be used to derive a particular PSM if the underlying platform for implementation changes.

The informational PIM, shown in Figure 3-7 on page 54, in OMG PLM Services currently supports the following 27 use cases, modelled using UML:

- ▶ Export assembly data
- ▶ Import assembly data
- ▶ Authentication and start up of session
- ▶ Authorization
- ▶ Start node identification
- ▶ Browsing down product structure data
- ▶ Browsing up product structure data
- ▶ Download product data
- ▶ Download metadata including structures
- ▶ Download a single digital file
- ▶ Generic object query
- ▶ Search in design space
- ▶ Upload product data
- ▶ Upload a single digital file (simple user interaction)
- ▶ Upload metadata including structures
- ▶ Change notification
- ▶ Display content of subscription list and confirm changes
- ▶ Change content subscription list
- ▶ Product class identification
- ▶ Browsing abstract product structures
- ▶ Browsing alternative solutions within an abstract product structure
- ▶ Retrieve configuration data within an abstract product structure
- ▶ Viewing change management information
- ▶ ECM participant proposal for a change
- ▶ ECM participant comments
- ▶ ECM participant approval
- ▶ ECM participant detailing and comments

These use cases are classified as informative in the specification, because they do not dictate the precise use of the formative components of the standard. Rather, they provide empirical evidence and support for the necessity of the formative information models.

Thus complementing these use cases, the formative informational PIM supports the following thirteen product information models as packages:

- ▶ PLM base
- ▶ Part identification
- ▶ Part structure
- ▶ Document and file management
- ▶ Shape definition and transformation
- ▶ Classification
- ▶ Properties
- ▶ Alias identification
- ▶ Authorization
- ▶ Configuration management
- ▶ Change and work management
- ▶ Process planning
- ▶ Multi-language support

A simple example of a standardized assembly data model (in the part structure mentioned above) using UML is shown in Figure 3-8 on page 57.

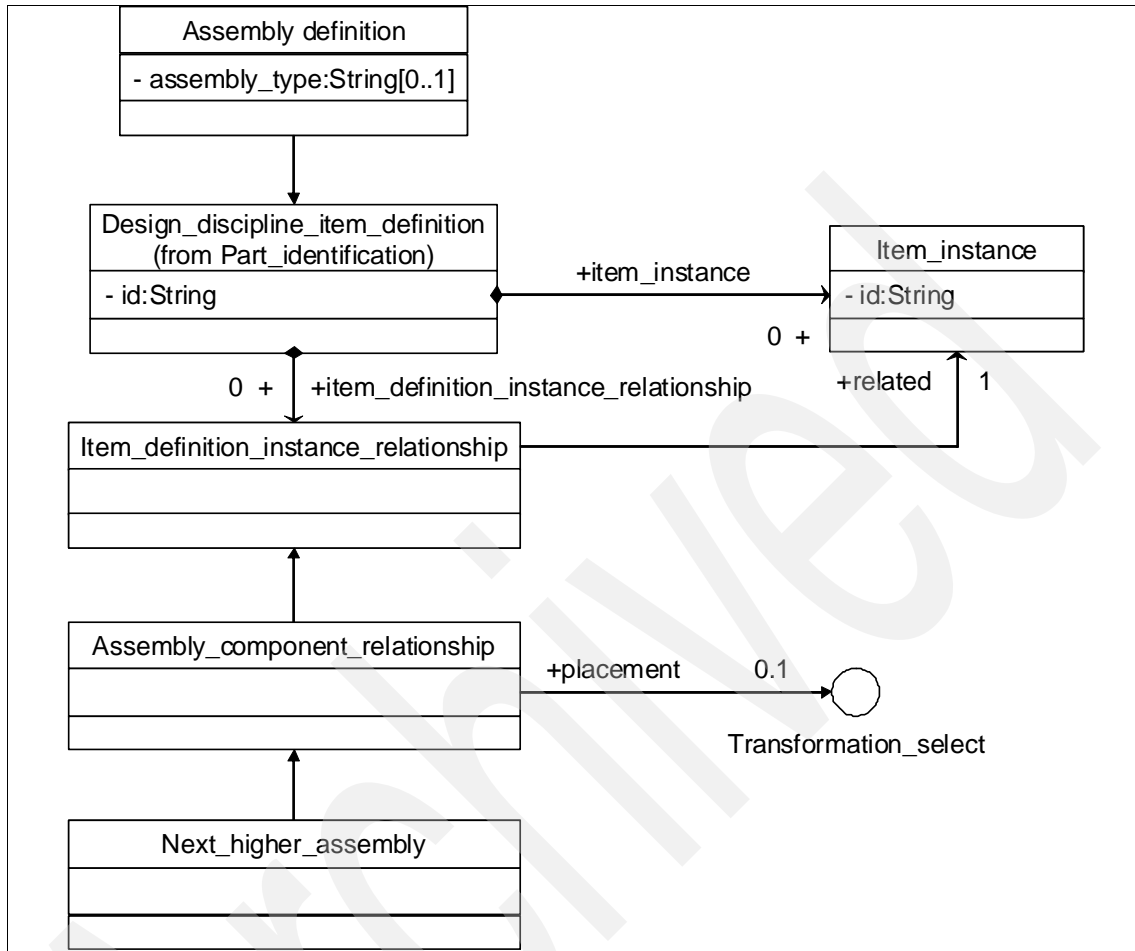


Figure 3-8 Excerpt of a UML model of an assembly structure

While the 13 packages of information models and the 27 use cases focus on the product (meta)data and how it can be used, the formative computational PIM (also pictured in Figure 3-7 on page 54) takes a computational viewpoint. It provides the necessary computational functionality to create, read, update, and delete (usually referred to as CRUD functions in software engineering) instances of the data schemas defined in the information models. It defines mechanisms to query and traverse instances of these data and, therefore, depends on the information viewpoint.

The computational PIM defines PLM connectors and PLM connection interfaces. When the connection has been made, several queries defined in the following query conformance points can be invoked.

- ▶ Utilities queries conformance point  
Provides specialization of an abstract base type called `PLM_query` with the possibility of concatenated, recursive, and batch queries.
- ▶ Generic queries conformance point  
Defines a toolset of classes to query data from a PLM system.
- ▶ XPath queries conformance point  
Provides the possibility to use arbitrary XPath expressions conforming to the W3C XPath specification as queries.
- ▶ Proxy queries conformance point  
Provides the possibility to avoid redundant data transfers in multiple query results.
- ▶ Specific queries conformance point  
Defines a rich set of low level specialized queries. It starts with defining some common queries classes (`Class Specific_query`, `Class Query_with_relying_type_predicate`, and `Class Relationship_query`) as base types for the rest of the specific queries. Table 3-10 on page 59 lists the rest of the 70 specific queries that support PLM-related atomic tasks.
- ▶ PDTnet queries conformance point  
Defines a set of specialized queries that fulfil the requirements of the uses cases described above.
- ▶ Message queries conformance point  
Defines queries derived from an abstract class called `PLM_message_query`.



Table 3-10 Specific queries of OMG PLM Services that deal with PLM-related atomic tasks

Activity_query	Document_version_query
Activity_authorization_query	Effectivity_assignment_query
Activity_element_query	Effectivity_query
Activity_relationship_query	Event_reference_query
Activity_resolved_request_query	External_model_query
Alias_identification_query	File_property_query
Alternative_solution_query	Geometric_model_relationship_query
Application_context_query	Item_classification_query
Approval_relationship_query	Item_classification_hierarchy_query
Assembly_component_placement_query	Item_definition_instance_relationship_query
Associated_activity_query	Item_instance_query
Associated_approval_query	Item_query
Associated_classification_query	Item_use_query
Associated_date_organization_query	Item_version_query
Associated_date_time_query	Item_version_relationship_query
Associated_document_query	Object_by_uid_query
Associated_effectivity_query	Objects_by_uids_query
Associated_file_query	Organization_query
Associated_general_classification_query	Organization_relationship_query
Associated_item_property_query	Person_in_organization_query
Associated_person_organization_query	Person_in_organization_relationship_query
Associated_process_property_query	Person_organization_assignment_query
Associated_project_query	Person_query
Associated_property_query	Product_class_query
Class_structure_query	Product_identification_query
Complex_product_query	Product_structure_query
Configuration_query	Project_assignment_query
Date_organization_assignment_query	Project_query
Design_discipline_item_definition_query	Simple_property_value_query
Document_classification_query	Work_order_query
Document_classification_hierarchy_query	Work_order_is_controlling_query
Document_property_query	Work_request_activity_query
Document_query	Work_request_query
Document_representation_query	Work_request_relationship_query
Document_structure_query	Work_request_scope_query

It must by now be clear to the reader the strong influences of ISO STEP metadata models, OMG PDM Enablers, and the PDTnet project on the PIM of OMG PLM Services specifications, as captured in Figure 3-7 on page 54.

The PSM for PLM Services is defined in XML Schema and in Web Services Description Language (WSDL). The informational viewpoint (XML Schema) and the computational viewpoint (WSDL) are derived from the corresponding formative PIM specifications described above. After mapping the PIM into PSM, these standardized product metadata models can be used as domain-specific

messages in the messaging layer of SOA. The standardized use cases realized by careful selection of queries can be used in the management and composition layer of SOA to compose higher-level business processes.

One of the major additions to PLM Services 2.0 is the support for the ECM process. Several informative use cases address this process. In addition, in the PSM layer, several formative informational models as XML Schemas and computational models in WSDL support the ECM process.

### 3.2.3 Architecture of OpenPDM

OpenPDM is a software product from the PROSTEP AG company. A simple view of the architecture of OpenPDM is shown in Figure 3-9. It is based on the PLM Services specifications, and uses the STEP neutral data models to interface with various proprietary PDM systems. With this type of architecture, it can mediate between a general business process and particular application tools by mapping the generalized functionality of the OMG PLM Services to the appropriate application programming interface (API) of proprietary applications (for example, commercial and in-house PDM systems). These are the domain-specific messages and processes that are needed to complement general SOA-based middleware to provide the product information sharing service.

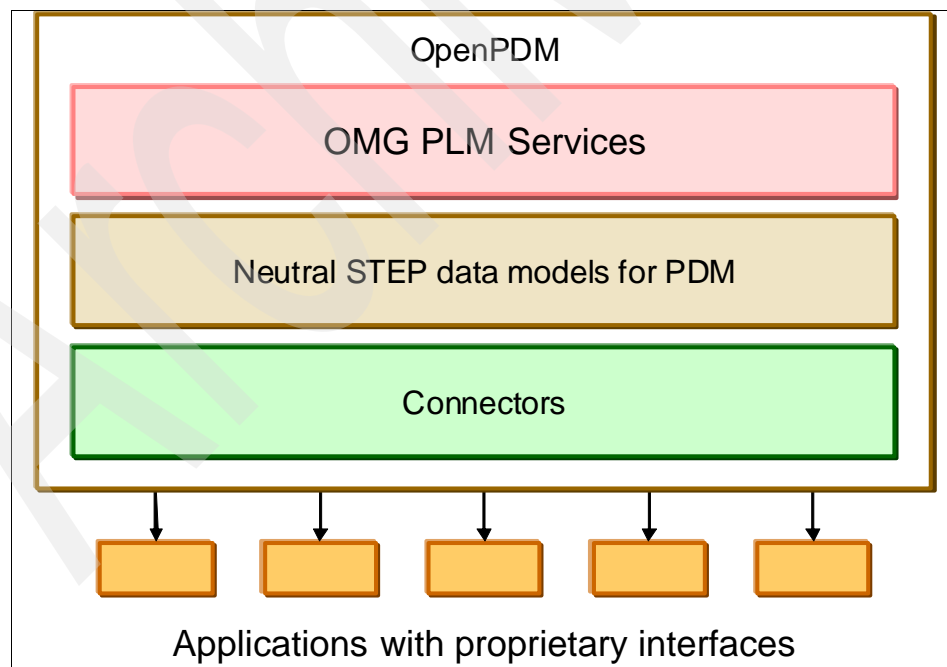


Figure 3-9 Architecture of OpenPDM

### 3.3 Summary

There are several other standards and specifications developed and maintained by organizations such as World Wide Web Consortium (W3C), OMG, OASIS, OAGi, and Web Services-Interoperability (WS-I). Some of these organizations are now engaged in developing standards for PLM using SOA. Of these, the following are noteworthy:

- ▶ OAGIS BODs

The business object documents (BODs) as defined by OAGi are quite wide in scope and are used in several product life cycle phases. They are the closest competitors to the OMG PLM Services specifications. But it is worth noting that the OMG PLM Services specification has greater depth in PDM and has drawn from the strengths of several other serious PLM-related standards. It was worked on for nearly a decade, with contributions from the ISO STEP community, the OMG PDM Enablers, and the ProSTEP iViP's PDTnet project. So it embodies considerable amount of knowledge and customer consensus in PLM.

- ▶ OASIS PLCS Web Services

Product Life Cycle Support (PLCS) is the phrase used for the STEP standard ISO 10303-239. After the initial STEP standard was issued by ISO, a technical committee was formed in the OASIS organization to develop this further. A set of PLCS Web Services has been developed by a private company (Eurostep) as part of the European Union-funded VIVACE project. Eurostep has put this forward on behalf of VIVACE to the OASIS PLCS committee for consideration as the basis for an OASIS PLCS Web Services standard.

- ▶ ISA-95/OAGIS SOA in Manufacturing

ISA-95 and OAGi are jointly working on standards for manufacturing systems integration. They are actively looking into the suitability of SOA for such integration in manufacturing.

The major business case for adopting OMG PLM Services and using OpenPDM is the reduction in cost and time in product information integration projects. Here we need to consider the total time and cost of ownership. These benefits arise from reducing the following costs:

- ▶ Cost of developing and maintaining connectors

OpenPDM provides a standard connector for each relevant PDM system in the market. If an enterprise were to implement these connectors itself, the initial expense for implementation and on-going maintenance would cost more time and money. An independent software vendor (ISV) can provide full maintenance and support for these standards-based connectors as products.

When new PDM versions are released, the ISV can provide tested and optimized connectors to these new versions in a short period of time. These connectors can often be provided at the same time as the release of the new PDM version, if the ISV has a business relationship with the PDM vendor.

- Cost of developing and maintaining nonstandard data formats

In solutions built on IBM WebSphere Process Server, a facility that generates generic business objects (GBOs) is needed. Due to the comprehensive basis of OMG PLM Services, virtually all types of proprietary application-specific PDM data can be represented by generic and standardized entities in the OMG PLM Services schema. This has been validated in many projects using OMG PLM Services and in an even larger number of projects based on the equivalent ISO STEP AP 214 model.

When combined with SOA, we also obtain higher quality-of-service, and better management and execution of the business processes.

An interesting, and important, feature of OMG PLM Services is the rigorous MDA approach applied to its development. It might have been more expedient to develop the product metadata models and business process elements without adopting such a rigorous, three-tiered (CIM, PIM, and PSM) approach and moving more directly to XML schemas, but we believe that the initial investment in developing a platform-independent viewpoint is the correct and more comprehensive approach. We can derive any platform specific model if and when our implementation platform changes. Thus, the shelf-life of our rich, standardized product metadata models and process elements are considerably enhanced due to the MDA.



## Part 2

# Scenarios and patterns

Part 2, “Scenarios and patterns” guides the user through a set of realistic business scenarios and architecture patterns leveraging IBM SOA for PLM. In order to give a realistic flavor to the business scenarios, we use a fictitious company Trucks Inc (see Chapter 6, “Achieving business flexibility with a service-oriented architecture” on page 121).

Examples are taken from real-life customer situations:

- ▶ Customer A: Joint Venture, Merge and intent to aggregate all PLM integration services in PTC Windchill instead of delegating this task to an ESB
- ▶ Customer B: Merge and application portfolio simplification/SOMA study

The expected audience for this part are as follows:

- Business analysts

Responsible for identifying and documenting requirements and analyzing the requirements from a business perspective, business analysts define current and future operational scenarios and processes, models, use cases, plans, and solutions. In addition, they work with their business contacts and the IT architects to ensure the proper translation of business requirements to IT solution requirements.

- IT architects.

This role is likely to be played by someone who is also an SOA architect, probably a senior or lead architect or even a chief architect. SOA architects perform IT architecture tasks based on a service- oriented approach. They identify services from a decomposition of business processes, and ensure that these services fit the business goals and drivers. They specify services to satisfy the architecture's principles and standards. They select appropriate reference architectures and create component models. They perform critical evaluation and selection of the packages, software, and hardware components of the architecture. The architects in this role are also responsible for performance, availability, and scalability of the applications. They also maintain the functional interface to the application infrastructure.

## **PLM Services 2.0: The neutral Product Lifecycle Management integration**

This section introduces the PLM Services V2.0 standard, its development, and the rationale for using the standard for ISVs or SIs within a number of application domains. It emphasizes the neutral nature of PLM Services 2.0 compared to proprietary vendor interfaces. It describes the Informational and the Computational Model of the standard and details to use case examples. It details using the specification for composite query for product information across several enterprise information systems and for the engineering change management (ECM) scenario leading to an SAP bill of material (BOM) synchronization.

The target audience are ISVs and SIs that need to use proven technology and data models for developing third party software or for building up cross domain integration layers to couple ERP and Product Lifecycle Management (PLM) systems.

## 4.1 Applying PLM Services

The OMG PLM Services Version 2.0 (OMG PLM Services V2) standards are the most recent attempt by the European automotive industry to manage increasing demands in collaborative engineering. Adopted as international object management group (OMG) standard, the OMG PLM Services standard enables original equipment manufacturers (OEMs) and suppliers to set up global collaborative environments through the entire product life cycle. This includes the following supported use cases:

- ▶ Browsing and updating product data (for example, part properties, assemblies, and complex product configurations) in distributed PLM systems,
- ▶ Engineering change management
- ▶ Exchange of 3D CAD data in both synchronous and asynchronous mode.

Implementations of the standard target the latest internet technologies (including XML and Web Services) to enable collaboration partners seamless communication online everywhere in the world.

### 4.1.1 Adopting a neutral PLM integration language

As described in Chapter 3, “Product Lifecycle Management Standards overview” on page 31, the PLM Services standard definition is derived from the model driven architecture (MDA) approach from the ISO standard STEP AP214 information model and enriched by a computational model derived from several sources.

The complete model-driven PLM Services standard specification approach is illustrated with the different starting points, the derivation and transformation chain, and the respective standard specifications in Figure 4-1 on page 67.



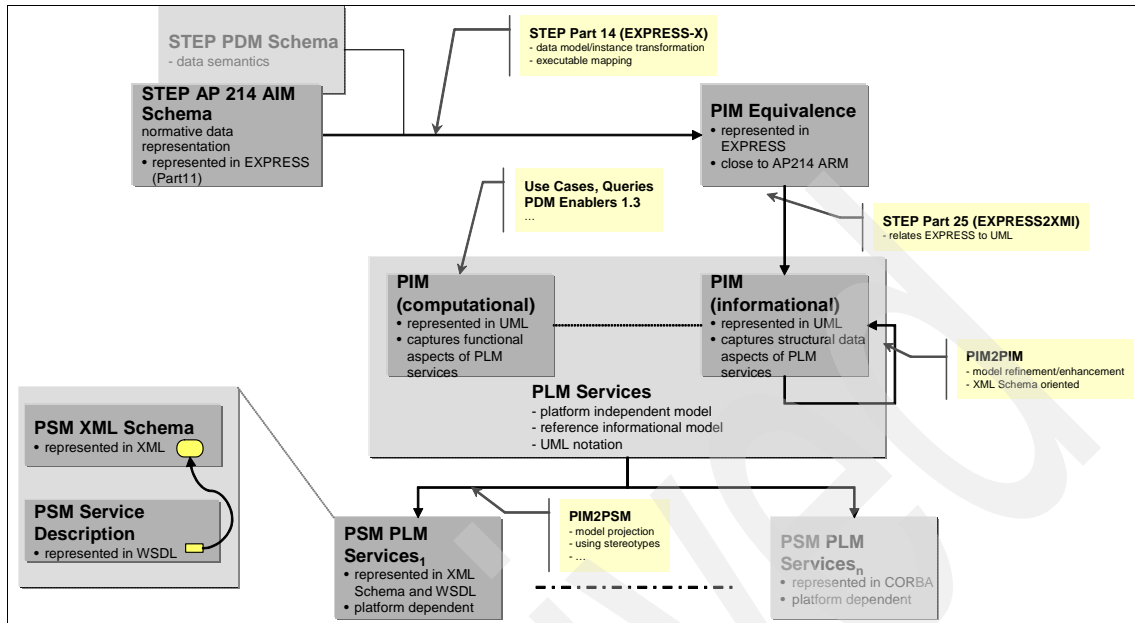


Figure 4-1 Model driven PLM Services standard specification approach

The complete transformation and model chain used during this model-driven approach starts with the PDM schema definition, a data model forming a common backbone of Product Data Management (PDM)-related STEP application protocols. It is defined by the Application Interpreted Model (AIM) level defined on low level data implementation entities in contrast to the Application Reference Model (ARM) level that corresponds to user model elements. The PDM Schema is defined so that it corresponds directly to STEP AP214 CC21. An EXPRESS-X mapping informal to PLM Services standard is defined so that it produces a suitable ARM representation, well-aligned with the STEP 10303-214 ARM model. The normative mapping between the AIM fragments and the ARM elements guarantees a seamless integration between file-based STEP data exchange techniques according to ISO 10303-21 and the new representations defined in PLM Services, for example, with the platform specific derivatives as defined below.

Differences between the informational model defined in EXPRESS and the actual AP214 ARM model are minimal. Undefined or ambiguous modeling entries have been defined by newly introduced elements simply derived from the AIM constructs or introduced to reduce the complexity of the ARM model. Significant examples of this complexity reduction are the simplified property concept and the simplified multi-language concept.

This EXPRESS model is called the platform independent model (PIM)-equivalence model because it is considered to be equivalent to the UML model of the PIM in the informational viewpoint of the PLM Services standard. The necessary transformation of the EXPRESS model into the UML model is described by ISO 10303-25 that was defined at the same time as the PLM Services specification was developed. In fact, PLM services are the first application of ISO 10303-25. This specification defines a mapping procedure to transform EXPRESS elements to XMI elements, the tool-independent representation of UML models. The XMI definition of PLM Services is normative. The foundation for this definition is based on an interpretation of MOF-based meta-modeling techniques without defining a complete model for the EXPRESS language. Differences between both modeling techniques are bridged by domain knowledge and convenient assumptions.

The complete PLM Services informational model specification provides a UML-based notation of one of the most important data models of ISO STEP standard family. This UML model is used with a revised data model element definition which directly establishes software development tools for cost efficient company specific implementations. It is enriched by the textual definitions as given in standard AP214 text with friendly copyright from ISO.

In the next transformation step of the MDA, the definition of a suitable computational model is added. Influenced by the OMG PDM Enablers specifications and the framework required to realize the PDTnet functionality, a basic set of connection management and generic object factories is defined. Generic basic functions form the following conformance points with the help of an layered architecture approach:

- ▶ Generic
- ▶ XPath
- ▶ Specific
- ▶ PDTnet
- ▶ ECM queries

The PDTnet conformance point got its name because it reflects the PDTnet use cases. The layered approach became feasible by replacing the PDTnet XPath-like query definitions with a complete algebraic query language. XPath is defined as a special instance of this facility. Both the informational and the computational models form the PIM. This PIM represents a comprehensive PLM reference model together with a powerful functional mechanism. According to the MDA, this PIM is the starting point for the derivation of one or more platform specific models (PSM).

The OMG PLM Services specification defines one PSM, a PSM with a normative XML Schema representation of the informational model and a representation of the functional model suitable for a Web Services implementation.

The informational model in the XML Schema PSM is split into four parts:

- ▶ The model representing the informational model itself with all the artifacts corresponding to the AP214 ARM modeling elements.
- ▶ The model representing all the artifacts corresponding to the computational viewpoint in the generic, XPath and specific queries conformance points.
- ▶ The model representing the PDTnet queries conformance point.
- ▶ A conformance point dedicated to ECM-related use cases. The conformance points depend on the each other in the given order.

The resulting Web Services description is close to the original PDTnet Web Services description. It follows the requirements of a standard derivation from an UML model and it is given in normative Web Services Description Language (WSDL).

#### **4.1.2 Using neutral PLM services across application domains**

The PLM Services define a data model covering the scope of the following units of functionality in AP214:

- ▶ Part identification
- ▶ Part structure
- ▶ Document and file management
- ▶ Shape definition and transformation
- ▶ Classification
- ▶ Properties
- ▶ Alias identification
- ▶ Authorization
- ▶ Configuration management
- ▶ Change and work management
- ▶ Process planning
- ▶ Multi-language support

The scope of the informational viewpoint was chosen to reflect the industrial requirements identified in the PDTnet project. The specification does not comply to a conformance class of AP214 but it is suitable to implement life applications for browsing product structure, for identifying changes, for maintaining references to work requests, or to track document versions of a CAD model and more. This has been approved throughout the predecessor project and other related activities. Furthermore, investments in this technology are offset by the scope extensions planned for the next version of PLM services so that compatibility is ensured. The following sections describe some of the possible applications of PLM Services in detail.

## **Communication across company borders**

The standard is suitable to implement cross-company scenarios with shared access to distributed PLM data sources. The major focus of the standard is to provide a common, easily available access mechanism to PLM information and to support collaborative engineering work in a heterogeneous system world. Standardized interfaces to the majority of the required PDM functionality allow short set-up times for establishing working partnerships. Organizational data is in scope of PLM Services specification. So, all PLM information may be extended by creator and owner information as well as alias names suitable for cross-company referencing of parts, documents and others.

The scenario is realized by providing a definition for a common neutral data model and access methods based on common Web technology, easily accessible for a large number of automotive companies. Using a Web-based transport mechanism delegates aspects of authentication and authorization (as well as encryption and other security requirements) to already existing technologies and infrastructures. PLM Services adds a mechanism to access valid data by session-related identifiers. This allows efficient interactive scenarios.

## **Communication across domain borders**

The standard is suitable to implement cross-domain scenarios interconnecting PLM, CAD, ERP, and other planning system data sources. The structural information provided by product or document information given in the PLM system provides a comprehensive access methodology to complete product definition data, including support measurements for manufacturing, process data, logistics, change management, and more.

## **Communication across system borders**

The standard is suitable to implement aggregated simultaneous views on multiple PLM systems. Sharing a common data model as defined by the specification and merging such a common view is an obvious application. But PLM Services provides not only a shared and system independent view on PLM data but defines the semantics of access functions and the necessary answers of the PLM systems to form valid PLM data. The aggregated view on multiple PLM systems is a one-system image on distributed data. Users are no longer forced to use multiple interfaces to assess data from different sources. PLM Services allows easy exchange of viewers to meet different requirements of different user groups. The need for erroneous data replication is minimized.

## Communication across technology borders

The standard is suitable to implement integration scenarios with existing STEP file-based infrastructures. Due to the normative mappings between AP214 AIM (the data model exchanged in STEP files) and the XML representation, (exchanged by the Web Services as defined in the PLM Services specification) a complete round-trip and unambiguous data transfer scenario becomes feasible. All the necessary transformation steps are defined in the standard. Therefore, a seamless transformation between an XML message with a PLM data set of information and a corresponding complete STEP part21 representation suitable for processing by an industry-strength STEP processor is defined by the PLM Services standard specification. Furthermore, the MDA nature of the specification will extend the data exchange capabilities to other implementations or representations and support the interoperability and compliance of the different data representations.

## 4.2 Using the PLM Services Informational Model

The informational model of PLM Services consists of 49 interfaces and 240 classes. It is categorized into 13 packages whose partitioning origins from the STEP AP214 standards units of functionality.

The packages are as follows:

- ▶ “PLM base” on page 72
- ▶ “Part\_identification” on page 72
- ▶ “Part\_structure” on page 74
- ▶ “Document\_and\_file\_management” on page 76
- ▶ “Shape\_definition\_and\_transformation” on page 77
- ▶ “Classification” on page 77
- ▶ “Properties” on page 78
- ▶ “Alias\_identification” on page 78
- ▶ “Authorization” on page 78
- ▶ “Configuration\_management” on page 79
- ▶ “Change\_and\_work\_management” on page 80
- ▶ “Process\_planning” on page 80
- ▶ “Multi\_language\_support” on page 80

## 4.2.1 PLM base

The package PLM base provides the basic types of all elements of the informational model:

- ▶ **PLM\_context**

The PLM\_context element holds basic descriptive data on date, time, version and application context. UID are relative to that context.

- ▶ **PLM\_core\_container**

The type PLM\_core\_container is the main container to all specific elements and PLM\_container is derived from it to hold all PLM elements (for example, those that are derived from PLM object). PLM\_core\_container provides an extension point for other application domains.

- ▶ **PLM\_object**

Every specific PLM-related type is derived from PLM\_object. PLM root object is derived from PLM object and is the base-type for all elements that are immediate children of PLM container. All other PLM\_object types are existentially dependent on PLM root objects and may not be referenced to without accompanying PLM\_root\_objects.

Finally, the data-type UID represents a primary key used by PLM\_core\_container and PLM\_object, that is, all elements of the informational model have a unique identifier.

## 4.2.2 Part\_identification

The package Part\_identification provides basic elements to describe part master data. A part, for instance, is represented by Item, a version of a part by Item\_version and a view on that version is modelled by associated Design\_discipline\_item\_definition. Moreover, ordered enumerations of type Item\_version can be created using Item\_version\_relationship. In addition to hierarchical assembly structures, the PDTnet Schema supports relationships between parts to characterize explicit alternates and substitutes for the assembly. Another relationship between part definitions exists to characterize the supplied part identification. See Figure 4-2 on page 73.

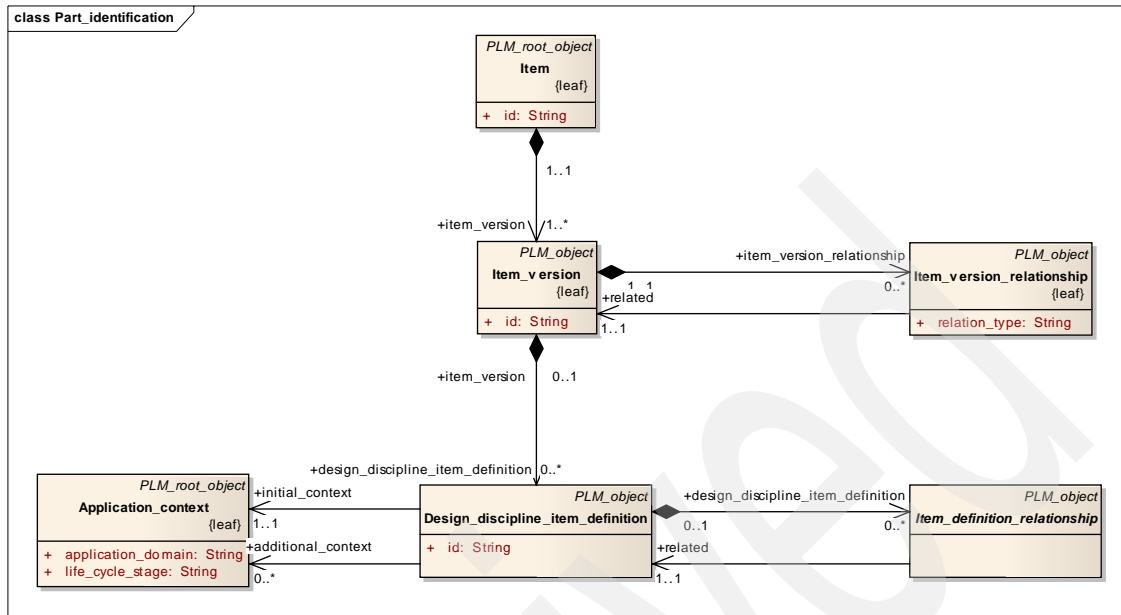


Figure 4-2 Part identification

The following relationships between item versions are supported:

- Derivation

This relationship describes where the related Item\_version is based on the parent Item\_version that is an earlier version of the same Item.

- Hierarchy

This hierarchical relationship defines where the related Item\_version is a subordinated version of the parent Item\_version.

- Sequence

This relationship defines a version sequence where the parent Item\_version is the preceding version and the related version is the following version.

- Supplied item

This relationship relates two Item\_versions that represent the same object in different organizational contexts. The Alias\_identification may also be used for defining an alias but Item\_version\_relationship shall be used if the related Item\_version corresponds to an item at the supplier's side. Additionally, it also defines a distinction of additional information such as name or organizational information is needed. See Section 4.2.8, "Alias\_identification" on page 78.

The `item_identification` maintains information common to all versions, disciplines, and life cycle views. It contains the base product number and name. The base number must not be subject to any encoding of information into a single complex parseable string.

There must be at least one version assigned to an `item_identification`. The `version_information` may represent a design revision or iteration in an item's design cycle. There may be more than one version associated with an `item_master`. It is recommended that at least one view (a design discipline item definition) be assigned to each item version.

The `view_definition` collects information relevant from the perspective of a particular application domain or life-cycle stage. There may be more than one life cycle view definition associated with a particular version of an item. This is especially important to enable different views on product assembly structures (see 4.2.3, "Part\_structure" on page 74).

### 4.2.3 Part\_structure

The package `Part_structure` refines the information of an `Item`. To describe additional aspects of this item, a collection of objects and relationships is provided. Examples of such aspects are the suppliers of the part or a reference to a CAD drawing describing the geometry of the parts.

The package `Part_structure` supports explicit hierarchical product structure facilities allowing representation of assemblies and their constituents. The basic instantiation pattern is comprised of an assembly as a hierarchy of `Item_instance` elements assigned to `Design_discipline_item_definition` instances and interrelated by specializations of `Item_definition_instance_relationship`. Furthermore, this package provides the tools to create bills of material (BOM), (for example, part lists assessed by weight, value, or other attributes important for the product's life cycle) and allows in-detail controlled views (for example, as designed, as manufactured, after substitution). See Figure 4-3 on page 75.



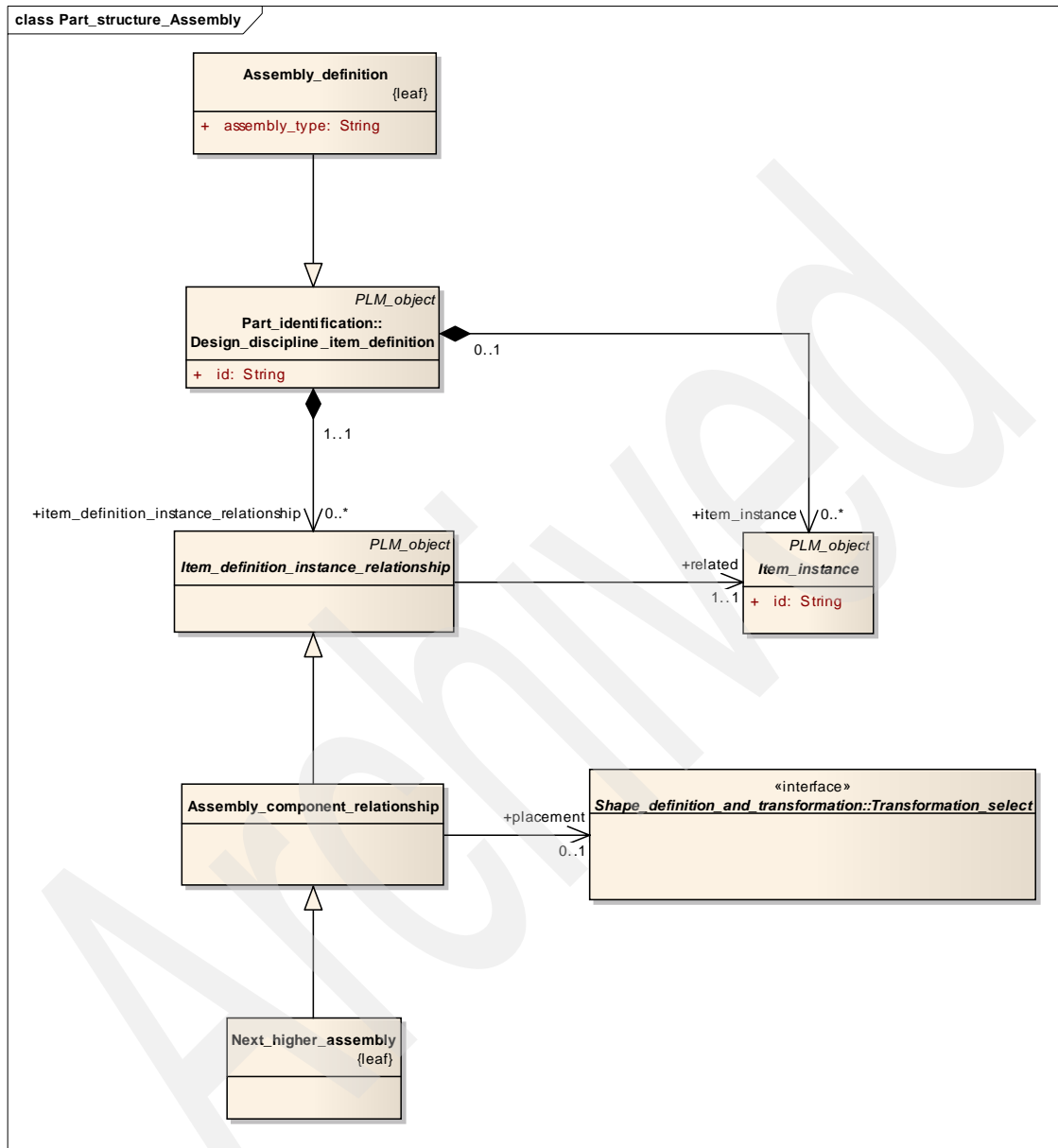


Figure 4-3 Part structure assembly

All these different view definitions are related to the same part versions. In this way, information that is common across the different assembly structures (the base version identification for all nodes that have definition in both structures) is

represented only once. Information different between the various views, such as associated management data, property information, or the product structure itself, is managed independently by using different design discipline item definitions.

The assembly structure would be constructed between the item definition instances that represent the part view definitions within the same appropriate view context (example design, for instance).

#### 4.2.4 Document\_and\_file\_management

The package Document\_and\_file\_management handles documents, the corresponding files, and related information. Document\_and\_file\_management package tracks documents that are not actively managed by the PLM system. The two main base classes are Document\_file and Document\_representation representing digital or physical documents. A Document\_representation is comprised of one or more Document\_files and can be related to other product data using Document\_assignments, for instance.

The package supports the definition of document structure by defining relationships of the Digital\_documents. The meaning of the relationship is given by the Relation\_type attribute. Where applicable the following predefined values must be used:

- ▶ Addition

This value specifies that the related document provides supplementary or collateral information with regard to the information provided by the relating document,

- ▶ Copy

This value defines a relationship where the related document representation is a copy of the relating document representation.

- ▶ Derivation

This value defines a relationship where the related document representation is derived from the relating document representation (for example, a raster image may be derived from a 3D CAD model).

- ▶ Decomposition

This value defines a relationship where the related document representation is one of potentially more sub documents of the relating document representation (for example, a decomposition of a document into clauses or into several CAD models).

- ▶ Peer

This value specifies that the related document provides required information with regard to that provided by the relating document. The peer document is essential for a complete understanding.

- ▶ Reference

This value defines a relationship where the related document is referenced from the relating, for example, a text processor representation referencing a CGM representation for inclusion as a graphical object in the text document.

- ▶ Sequence

This value defines a logical sequence where the related document representation comes after the relating document representation (for example, a sequence of clauses).

- ▶ Substitution

This value defines a relationship where the related document representation replaces the relating document representation.

- ▶ Translation

This value specifies that the related document is generated automatically from the relating document.

The relation types decomposition and substitution can be used to model assembly-type structures of document representations. This approach does not mirror the approach taken with `assembly_component_usage` for the product structure. This reflects the understanding that there is no common requirement to manage document representation structures in the same way as product structure.

## 4.2.5 Shape\_definition\_and\_transformation

The package `Shape_definition_and_transformation` defines the geometric model of a shape, the transformation, and the relationships to other items. The geometric model allows for using auxiliary elements and elements that describe product data. The relationship to other items may happen with or without transformation. The linkage between geometric structures is recommended.

## 4.2.6 Classification

The package `Classification` divides product data into different categories identified by arbitrary names. The names are defined by related classifications. In cooperation with the hierarchy class, it is possible to build hierarchical structures of additional information for items, documents, and files.

## 4.2.7 Properties

The package Properties describes all the detailed features associated to items and documents and their versions. These detail data comprise general measurable values, optionally given with units and accuracy. Specific property model classes are specified for cost, duration, mass, material, quality, recyclability, and shape properties. Measures are optionally restricted by a range or limit value and a general parameter feature may be used to extend the given model space if necessary. All properties may be modelled to be valid within a context.

## 4.2.8 Alias\_identification

The package Alias\_identification allows to associate an identifiable item and its version with an additional identifier. By this identifier, the object can be identified in another context. Therefore an Identification\_role or an Applied\_organization\_assignment has to be specified.

This facility synchronizes different identification schemes of existing in-house systems, identification schemes for development and maintenance, or to associate the numbering scheme of an OEM with the numbering scheme of a suppliers system. Therefore, this facility is the key for the support of cross-company use cases like engineering change management and supplier integration along the lifeline of a product.

## 4.2.9 Authorization

The package Authorization provides the possibility to describe the structure of an organization. The central item in this package is Person\_in\_organization. It is the specification of a Person in the context of an Organization. The role specifies the relationship between Person and Organization. Organizations and people in organizations are represented as they perform functions related to other product data and data relationships. A person must exist in the context of some organization. An organization or a person in an organization is then associated with the data or data relationship in some role indicating the function being performed. Both people and organizations may have addresses associated with them. The address is entirely optional. It is done with the address entity being related to the person (by personal\_address) or organization (by organizational\_address).

Groups of people (for example, companies, countries, and so forth) are represented by the organization entity. The identification or ID data is optional. Because this information is important in providing unique identification to the organization or company, however, it is recommended that this field always be

populated with unique data. If appropriate, an URL-like convention for the organization identifier may be useful, for example, prostep.de. The name attribute must contain the common nomenclature of the organization. The description attribute may contain a characterization of the type of the organization, or a textual explanation of its reason for existence.

Another important part of this package is Approval. An Approval is a judgement concerning the quality of certain product data that is subject of this Approval. An Approval represents a statement made by technical or management personnel whether certain requirements are met.

Approval may be represented as a simple basic approval, or it may represent a more complex approval cycle involving multiple approvers, on different dates and times, and possibly with different status values. The approval constructs actually support an approval cycle. An approval may only need one signature corresponding to the simple basic approval scenario.

The level of an approval is understood to represent the aspect for which the approved object is endorsed. This level represents a state for which the approved object requires approval. The approval status indicates the level of acceptance given to the object for the specified state.

The entity `Design_discipline_item_definition` exists to represent different views on a part version. Each view is characterized by a type and a life cycle stage. Various part view types correspond to different states that a part assumes throughout its life cycle. The decision to represent the state of a part as a level of approval on a view definition or an entirely different view definition is dependent upon individual business processes. Different views are typically characterized by different life cycle stages such as design and manufacturing. Approval levels are various states defined within a given life cycle stage view, and are typically dependent upon the approval cycles within an individual business process. The states associated with each defined state are described by the approval status values leading towards the status approved indicating achievement of the particular level, or state.

#### **4.2.10 Configuration\_management**

The package `Configuration_management` provides capabilities for configuration for interrelated functional and physical characteristics of a product defined in requirements for product design, realization, verification, operation, and support. For instance, `configuration_management` assures that the production line for diesel engines does not need ignition plugs. Therefore, a configuration contains a defined set of specifications, including their relationships.

#### **4.2.11 Change\_and\_work\_management**

The package `change_and_work_management` exchanges information about work items. Contrary to most other packages, the scope of `change_and_work_management` is not the description of a part, document, or file, but the description of an action to be done with a certain part, document, or file. The package describes what, when, how, and from whom the work has to be done.

#### **4.2.12 Process\_planning**

The package `Process_planning` is used to determine the required steps to produce an item. This item might be a simple bolt or an even more complex item like a complete car. Each step is represented by a `Process_operation_definition` that defines a single manufacturing or control operation.

#### **4.2.13 Multi\_language\_support**

The package `Multi_language_support` determines the language of text associated with an object. It is possible, therefore, to store product data information in different languages at the same time and let personnel of the respective life cycle stage choose to display this information in the desired language, if available. Usually, the chosen language is the same in every PLM context.

### **4.3 Using the PLM Services Computational Model**

The Computational Viewpoint specifies the services and operations of the PLM Services. An implementation of these operations for a specific PLM system is called a PLM Services connector.

This section illustrates the usage of the basic operations of a PLM Services connector together with code snippets of a Java™ class that uses a straight forward binding of the PLM Services.

### 4.3.1 PLM Services connectors

PLM Services connectors consists of a multilevel service instantiation hierarchy to allow decoupling of clients from different implementation aspects. The levels are represented by the following classes and interfaces:

- ▶ PLM\_resource\_adapter
- ▶ PLM\_connection\_factory
- ▶ PLM\_connection

The PLM\_resource\_adapter decouples from the specific implementation of the vendor of the connector. The PLM\_connection\_factory decouples from the location where the service is running. The PLM\_connection interface with specializations PLM\_general\_connection and PLM\_message\_connection decouples from the native API of a PLM system. Another abstraction is used to create PLM entities and computational data objects by means of the class PLM\_object\_factory.

### 4.3.2 PLM\_object\_factory class

The class PLM\_object\_factory decouples clients from the concrete implementation classes of the PLM data objects. PLM data classes are used to define the parameters and return types of PLM Services operations. For each non-abstract data class of the informational and computational viewpoint, the PLM\_object\_factory defines a create method. Each create method specifies a parameter list that contains values for required attributes and required referenced and contained data objects of the new object. If the class of the created object plays the role of a containee in a composition, the required container object is defined as an additional parameter with name parent of the create method.

### 4.3.3 PLM\_resource\_adapter abstract class

The abstract class PLM\_resource\_adapter is the entry point for a PLM Services client to get access to the services. The static class operation PLM\_resource\_adapter get\_instance returns a PLM\_resource\_adapter instance and decouples a client from the specific implementations of PLM Services connectors. There can be different connectors for different PLM systems and different connectors for different middleware technologies. With the use of PLM\_resource\_adapter, the clients can be compile-time independent from those different connector implementations. The coupling can be configured by properties and is made at runtime.

On a PLM\_resource\_adapter instance, the operation get\_connection\_factory can be used to get a PLM\_connection\_factory instance from a specific connector.

#### 4.3.4 PLM\_connection\_factory service

The PLM\_connection\_factory service can be used to create new sessions with a PLM system. PLM services supports two kinds of interaction styles between client and service:

- ▶ Remote procedure call (RPC)  
This interaction is modelled as a PLM\_general\_connection session interface.
- ▶ Message exchange  
This interaction is modelled as a PLM\_message\_connection session interface.

The PLM\_connection\_factory interface provides one create method for each session style.

#### 4.3.5 PLM\_service interface

The PLM\_service interface provides operations to manage service properties of type PLM\_property. These operations include the following methods:

- ▶ get\_properties()
- ▶ set\_properties()
- ▶ delete\_properties()

The set of supported values for instances of type PLM\_property is implementation-specific. The PLM\_property parameters may store the current version of the used PLM Services implementation in a container of type Properties. On return, a PLM\_general\_connection instance can read, set, or delete property values. The PLM\_general\_connection instance is derived from PLM\_connection interface, which is derived in turn from the PLM\_Service interface.

#### 4.3.6 PLM\_connection service

The session creation is provided by the operations of the PLM\_connection\_factory interface. The PLM\_connection interface provides the end of a session concept. All session instances realize the interface PLM\_connection.

The operation close can be used to end a session. After a call of operation close(), the service can release all resources for that session. Further operation calls on that session must throw an Invalid\_session\_operation exception.



The operation `get_download_URL` can be used to obtain an URL of a certain `Digital_file` residing in a PLM system identified by the given `file_uid`. The returned URL is appropriate for immediate download of the respective file.

The operation `get_upload_URL` can be used to obtain an URL for a certain `Digital_file` identified by the given `file_uid` that is appropriate to upload to a PLM system.

#### 4.3.7 PLM\_general\_connection interface

The `PLM_general_connection` interface represents a session with a service with an RPC style of interaction. RPC style means that the operations of the interface can be used to query or update the content of a PLM system in a synchronous manner. The query operation directly returns the requested information in its result (for example, blocking method invocation). The change operations execute their manipulations in the PLM system inside the operation calls.

The signatures of the operations of the interface `PLM_general_connection` are generic. They use the abstract types `PLM_query` to query product-data and `PLM_core_container` to transfer actual payload.

The operation `query` of the interface `PLM_general_connection` can be used to query product-data from a PLM system. The implementation of the `query` operation has to ensure that all PLM entity instantiation constraints from the informational model are fulfilled in the result objects.

The generic signature of the `query` operation allows the `query` to be used for queries from different query conformance points.

The `write` operation can be used to create new PLM entities and to update attributes and associations of existing entities in a PLM system. The implementation of the `write` operation has to decide if a PLM entity already exists, or if it has to be created by the UID of the entity.

If the client wants to create a new PLM entity in a PLM system, the client creates a local entity using the `PLM_object_factory create*` operations. The entity is transferred inside a `PLM_core_container` to the service. The service recognizes it is a new entity that has to be created on the server side by the UID of the entity. If the client wants to update some attributes of a PLM entity, the client has to use the operation `query` to get the current attributes (especially the UID) of the entity. The client can then manipulate some attributes of the entity (except the UID) and then return the entity with the operation `write` to the service. The service recognizes that the entity already exists on the server side by the UID of the entity and updates the attributes of the entity.

The distinction between new and existing entities on the server side requires that client and service use different UID ranges when creating new entities.

The operation delete interface of the PLM\_general\_connection is used to delete certain product data from a server.

The parameter uids of Type UID[] identifies the PLM\_object instances to delete. An additional set of properties of type PLM\_property is used to control the delete actions implementation specific (for example, if a server provides recursive deletion capabilities, a dedicated PLM\_property may control this feature). In certain cases (for example, an object cannot be deleted) an Object\_not\_deleted\_information object is returned for the respective PLM\_object. If the deletion of a certain PLM\_object causes another object to be deleted, an Additional\_object\_deleted\_information object is returned.

#### **4.3.8 PLM\_message\_connection interface**

The PLM\_message\_connection interface provides a service for asynchronous message exchange. Asynchronous means that the interface to build workflows defined as an exchange of messages can be executed when a single message instance is triggered. The PLM\_message\_connection service provides operations to send messages to, and to poll for messages on, a service. The send and poll operations themselves are RCP-style operations (for example, appropriate for asynchronous interaction style).

The signatures of the operations of the interface PLM\_message\_connection are generic. They use the abstract classes PLM\_message\_query for queries and PLM\_core\_container and PLM\_context (composed to PLM\_message objects) for payload data.

The query\_messages operation of the interface PLM\_message\_connection is used to query messages (for example, instances of PLM\_message) from the server.

The parameter query of type PLM\_message\_query determines the criteria messages have to fulfill in order to become selected and returned by the server. Because PLM\_message\_query is abstract, clients have to use non-abstract specializations to query messages. The Message Queries Conformance Point introduces queries to select messages by context (for example, by a certain PLM\_context identifying messages belonging to particular workflow steps or states of individual protocols) or by certain properties like date and time of their arrival at server-side.

The write\_messages operation of the interface PLM\_message\_connection is used to write messages (for example, instances of PLM\_message) to the server.

The parameter messages of Type PLM\_message defines the set of messages to write to the server. A PLM\_message consists of a PLM\_context and a PLM\_core\_container instance. The PLM\_context determines the actual position of the related message within a workflow or protocol while the PLM\_core\_container includes the actual payload, that is product-data.

The result of query\_messages is a set of PLM\_processing\_information objects describing the actual action the server performed for each product-data object included in the related PLM\_core\_container instance of the respective message.

The delete\_messages operation of the interface PLM\_message\_connection deletes messages (for example, instances of PLM\_message) from the server.

The parameter contexts of type PLM\_context identifies PLM\_message instances within the server to delete. Each PLM\_message is assigned a PLM\_context acting as its identifier and indicating the position of that message within the scope of a workflow or protocol.

The result of delete\_messages is a set of PLM\_processing\_information objects describing the actual action the server performed for each product-data object included in the related PLM\_core\_container instance of the respective message.

### **4.3.9 Query conformance points**

PLM Services groups queries into so-called conformance points. A conformance point defines a certain level of abstraction, and the set of all available conformance points builds up a hierarchy (see Figure 4-4 on page 86). Every conformance point encapsulates the semantics defined by its next lower conformance point. The dotted top of the pyramid illustrates the extensibility of this approach. Users are able to define their own conformance points based on already defined conformance points.

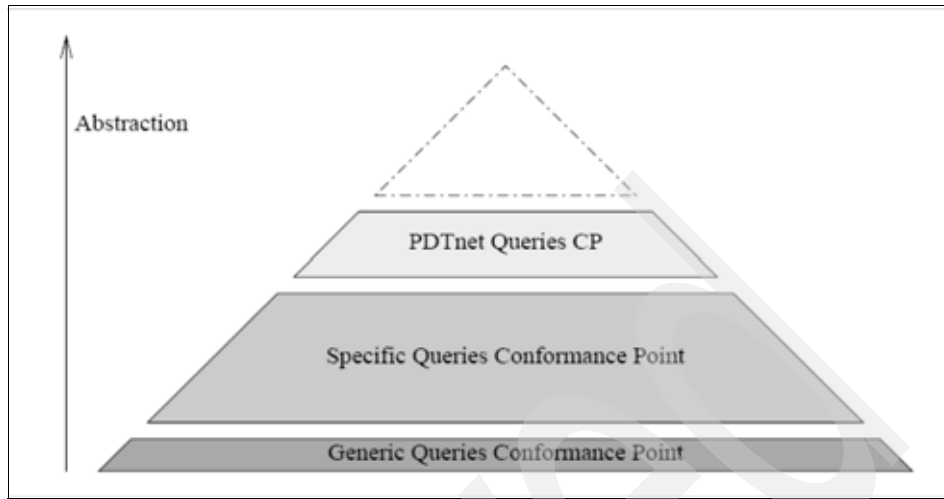


Figure 4-4 Query conformance points in layers of abstraction provided by the PLM Services query-language

The Generic Queries Conformance Point is the bottommost layer. It provides a toolkit to traverse complex product-data structures by means of `Location_path` and `Location_step` elements that are similar to the elements and semantics of the XML Path Language. A filter mechanism is provided by a wide range of predicates.

The next layer, Specific Queries Conformance Point, defines queries based on the generic queries selecting atomic product-data types like items or documents. For each atomic product-data type, one specific query is defined in terms of `Location_path` and `Location_step` instances.

On top of the Specific Queries Conformance Point, the PDTnet Queries Conformance Point is defined. Each PDTnet query reflects a whole use case as defined by the PDTnet project. Each PDTnet query is defined in terms of its next-lower conformance point. In this case, the Specific Queries Conformance Point. The specific query `Item_selection_query`, for instance, is defined by using the following specific query instances:

- ▶ `Item_query`
- ▶ `Item_version_query`
- ▶ `Design_discipline_item_definition_query`

## Conformance Points Not Part of the Hierarchy

Another important conformance point, the Utility Queries Conformance Point, defines a set of tools to concatenate queries, perform queries recursively, by condition or in a batch-like fashion. Because all conformance points, except for the Generic Queries Conformance Point, are able to make use of these tools, the utility queries are not part of the pyramid in Figure 4-4 on page 86.

Besides the aforementioned conformance points, the OMG specification defines three more conformance points that are not part of the hierarchy shown in Figure 4-4 on page 86.

- ▶ The XPath Queries Conformance Point

This conformance point consists only of one query: `X_path` (derived from `PLM_query`). This query allows clients to select product-data by specifying an XPath expression. Because PLM Services defines a platform-specific model based on XML Schema, a server can map product data received from attached PLM systems to XML documents and apply the given XPath expression to select the desired subset.

- ▶ The Proxy Queries Conformance Point

This conformance point defines classes to represent the identity of objects in complex product-data structures. That is, the proxy queries provide means to avoid transferring redundant data. Therefore, a `PLM_proxy_object` represents the identity of a full-fledged `PLM_object`, but with reduced features (for example, less detail).

- ▶ The Message Queries Conformance Point

This conformance point defines queries derived from `PLM_message_query` to select `PLM_message` objects from a server implementing `PLM_message_connection` by context or date and time properties.

## Generic Queries Conformance Point

The Generic Queries Conformance Point defines a generic query model that can be used to select arbitrary sets of PLM entities from a PLM system and to traverse from a given set of start PLM entities over arbitrary relationships to related PLM entities. The queries defined in the Generic Queries Conformance Point are low level and, because of its unlimited selection and traversal capabilities, they are probably difficult to implement for a real PLM system. In fact, this conformance point is not intended to be implemented, but to define the semantics of the queries of other conformance points by defining a semantically equivalent generic query instance for a high level query instance.

*Example 4-1 Item\_query instance and semantically equivalent Location\_path instance*

---

```
Item_query itemQuery = new Item_query();
itemQuery.setName("engine");
Location_path equivalentGenericItemQuery = new Location_path();
String_select_predicate predicate = new String_select_predicate("name", "engine");
locationPath.getRelatedObjectPredicate().add(predicate);
Location_step locationStep = new Location_step();
locationStep.setRoleName("Item");
locationStep.setRoleDeclaringTypeName("PLM_container");
equivalentGenericItemQuery.setFirstStep(locationStep);
```

---

### **Utility Queries Conformance Point**

The Utility Queries Conformance Point defines queries to enable concatenated, recursive, and batch execution of PLM\_query instances.

The concatenation of queries is realized by the association `Concatenatable_query.next_query` linking a particular PLM\_query object with a set of PLM\_query objects that becomes evaluated subsequently. If a certain query is a specialization of `Concatenated_query`, the result-set is determined by the union of the result-sets of `Concatenated_query.first_query` and the actual query that is its subtree determined by `Concatenated_query.next_query`. In this way, the result-set of the actual query becomes the initial node-set (for example, the set of start-nodes) of each `next_query`.

*Example 4-2 Usage of Concatenated\_query*

---

```
Item_query itemQuery1 = new Item_query();
itemQuery1.setName("engine");

Item_query itemQuery2 = new Item_query();
itemQuery2.setName("air-condition");

Concatenated_query concatenatedQuery = new Concatenated_query();
concatenatedQuery.setFirst_query(itemQuery1);
concatenatedQuery.setNext_query(new PLM_query[]{itemQuery2});

List<PLM_property> queryProperties = new List<PLM_property>();
// set some PLM_property

PLM_core_container resultContainer = generalConnection.query(concatenatedQuery,
queryProperties.toArray());
```

---

The type `Batch_query` combines distinct `PLM_query` instances to a batch job. A `Batch_query` instance is evaluated by considering each contained `PLM_query` instance independently and creating one combined result-set determined by the union of the result-sets of all contained queries. `Batch_query` does not support result-set propagation to next `PLM_query` instances. That is, the initial node-set of each contained `PLM_query` instance is equal to the initial node-set of `Batch_query` itself.

---

*Example 4-3 Usage of `Batch_query`*

---

```
Batch_query batchQuery = new Batch_query();
batchQuery.setQuery(new PLM_query[]{itemQuery1, itemQuery2});

List<PLM_property> queryProperties = new List<PLM_property>();
// set some PLM_property

PLM_core_container resultContainer = generalConnection.query(batchQuery,
queryProperties.toArray());
```

---

The type `Conditional_query` facilitates conditional execution of a `PLM_query` instance depending on a certain condition. If the condition evaluates to true, the `PLM_query` instances assigned to `Conditional_query.when_true` is evaluated next. Otherwise the `PLM_query` instances assigned to `Conditional_query.when_not_true` is evaluated.

---

*Example 4-4 Usage of `Conditional_query`*

---

```
Conditional_query conditionalQuery = new Conditional_query();
Boolean condition = new Boolean(name.equals("engine"));
conditionalQuery.setCondition(condition);
conditionalQuery.setWhen_true(itemQuery1);
conditionalQuery.setWhen_not_true(itemQuery2);

List<PLM_property> queryProperties = new List<PLM_property>();
// set some PLM_property

PLM_core_container resultContainer = generalConnection.query(conditionalQuery,
queryProperties.toArray());
```

---

The `Recursive_query` provides means to traverse hierarchical product-data structures. In general, executing queries against a tree of product-data elements as defined by the informational viewpoint would require a recursive traversal. This recursion is established by iteratively applying the `PLM_query` assigned to `Recursive_query`. The attribute `Recursive_query.maximum_recursion_number` controls the depth of recursion. If not set, or assigned the value 0, the associated

PLM\_query instance is applied only once. If the attribute is set to a value greater than 0, the associated PLM\_query instance is applied n times. Recursive\_query has the same semantics as the concatenation of n identical PLM\_query instances. Recursive\_query.maximum\_recursion\_number set to a value less than 0 implies infinite recursion.

---

*Example 4-5 Usage of Recursive\_query*

---

```
Recursive_query recursiveQuery = new Recursive_query();
recursiveQuery.setMaximum_recursion_number(3);
recursiveQuery.setQuery(plmQuery);

List<PLM_property> queryProperties = new List<PLM_property>();
// set some PLM_property

PLM_core_container resultContainer = generalConnection.query(recursiveQuery,
queryProperties.toArray());
```

---

### **Proxy Queries Conformance Point**

The Proxy Queries Conformance Point provides means to avoid redundant data transfers in the scope of a session. Thus, the actual size of data transferred over multiple queries can be significantly reduced. For that purpose, a special container type (the PLM\_proxy\_container, derived from the PLM\_container interface) is introduced. PLM\_proxy\_container is able to host both complete (for example, full-featured) PLM\_objects and so called proxy-objects (instances of type PLM\_proxy\_object with reduced features). Clients can ask to receive proxy-objects instead of full-featured objects if the full-featured versions of the respective instances are already cached from earlier requests.

---

*Example 4-6 Usage of PLM\_proxy\_container, PLM\_proxy\_object and PLM\_proxy\_feature*

---

```
Item_query itemQuery = new Item_query();
itemQuery.setName("engine");

Proxy_query proxyQuery = new Proxy_query();
proxyQuery.setUid("id_10019_786");
proxyQuery.setNext_query(new PLM_query[] {itemQuery});

List<PLM_property> queryProperties = new List<PLM_property>();
// set some PLM_property

PLM_proxy_container resultProxyContainer = (PLM_proxy_container)generalConnection.
    query(recursiveQuery, queryProperties.toArray());

PLM_proxy_object[] proxyObjects = resultProxyContainer.getPLM_proxy_object();
```



```
// extract proxy-feature from first proxy-object:
if (proxyObjects.length > 0) {
    PLM_proxy_feature[] proxyFeature = proxyObjects[0].getPLM_proxy_feature();
}
```

---

The Proxy Queries Conformance Point introduces the query type `Proxy_query` to select objects by their identifiers (for example, the UID). `Proxy_query` does not return full-featured instances of type `PLM_object`, but `PLM_proxy_object` instances. Because `Proxy_query` is a specialization of `Concatenatable_query`, it can have next-queries. Therefore, next-queries of type `Proxy_query` traversing the sub-tree of an initial `PLM_object` creates instances of type `PLM_containment_feature` or `PLM_reference_feature`. These represent the actual structure. That is, it represents the containment and reference relationships.

### **XPath Queries Conformance Point**

The XPath Queries Conformance Point consists of a single query called `X_path` derived from `PLM_query`. The only parameter of query type `X_path` is an XPath expression compliant to W3C's specification. The concrete version of XPath (for example, 1.0 or 2.0) depends on the actual server-implementation. Because PLM Services defines a platform-specific model based on WSDL and XML Schema, a server can represent product data received from an attached PDM system as XML documents. The server selects the requested subset from an XML document by applying the XPath expression given by `X_path.expression`.

#### *Example 4-7 Usage of the XPath Queries Conformance Point*

---

```
X_path query = new X_path();
query.setExpression("/PLM_container/Item/Item_version/Design_discipline_item_definition");
List<PLM_property> properties = new List<PLM_property>();
// set some PLM_property
PLM_core_container result = generalConnection.query(query, properties.toArray());
```

---

### **Specific Queries Conformance Point**

The Specific Queries Conformance Point consists of 70+ queries, all defined in terms of the toolkit provided by the Generic Queries Conformance Point. A specific query is characterized by the fact that it retrieves atomic product data elements like items, documents, products, persons, and so on. Because all specific queries are derived from the base type `Specific_query`, which itself is derived from `Concatenatable_query`, one can create complex queries. For instance, hierarchies of queries by using `Concatenatable_query.next_query`. As a result, the result-nodes of a specific query `a` is passed to all concatenated

queries. That is, for all queries *bi* in *a.next\_query*, the result-nodes from *a* become input-nodes for *bi*. Besides *Concatenatable\_query*, the *Utility Queries Conformance Point* (see “Utility Queries Conformance Point” on page 88) provides additional tools to create complex queries like *Batch\_query*, *Recursive\_query*, and *Conditional\_query*.

*Example 4-8 Usage of the Specific Queries Conformance Point: Building a complex query*

---

```
Item_query itemQuery = new Item_query();
itemQuery.setId("42");
Item_version_query itemVersionQuery = new Item_version_query();
itemVersionQuery.setId("A");
Design_discipline_item_definition_query ddidQuery = new
Design_discipline_item_definition_query();
ddidQuery.setId("1");
List<Concatenatable_query> nextQueries = new List< Concatenatable_query>;
nextQueries.add(itemVersionQuery);
itemQuery.setNext_query(nextQueries.toArray());
nextQueries.clear();
nextQueries.add(ddidQuery);
itemVersionQuery.setNext_query(nextQueries.toArray());
List<PLM_property> properties = new List<PLM_property>();
// set some PLM_property
PLM_core_container result = generalConnection.query(itemQuery, properties.toArray());
```

---

### **PDTnet Queries Conformance Point**

The *PDTnet Queries Conformance Point* consists of queries to select, investigate, and traverse product-data and product-structures respectively. As a result of the *PDTnet* project, the *PDTnet* queries define queries on the following fundamental product-data types:

- ▶ Document
- ▶ Item
- ▶ Product
- ▶ Work\_order
- ▶ Work\_request

These queries encapsulate atomic *PDTnet* use cases and are appropriate to be composed to larger use cases like a sequence of selection, retrieving details, and traversal starting from a specific *Item*, for instance.

Table 4-1 gives an overview of all specified PDTnet queries (see the Prostep iViP document *PDTnet Implementation Guide*) and shows the available query-categories (for example, selection, detail, and traversal) for each product-data type. For example, the set of available PDTnet queries for product-data type Item consists of the following query types:

- ▶ Item\_selection\_query
- ▶ Item\_detail\_query
- ▶ Item\_traversal\_query

All detail-queries are derived from the non-abstract base-type General\_detail\_query.

*Table 4-1 Selection, detail, and traversal-queries of the PDTnet Queries Conformance*

	Selection	Detail	Traversal
<b>General</b>		•	
<b>Document</b>	•	•	•
<b>Item</b>	•	•	•
<b>Product</b>	•	•	•
<b>Work_order</b>	•	•	
<b>Work_request</b>	•	•	•
The • symbol reflects the use cases defined by the PDTnet project.			

The semantics of each query-category is well-defined. Selection-queries act as so-called start-node queries, that is, they select an initial set of product-data elements by non-technical criteria like id, name, version\_id, and classification\_name (for the example of Item\_selection\_query). Subsequent queries like detail- and traversal-queries accept only UIDs (for example, the primary key in the scope of a session) of already selected product-data elements as input. While detail-queries select more details of certain product-data (for example, assigned documents, attached properties, classifications or approvals), traversal queries select elements within another layer of the surrounding product-structure (for example, hierarchy).

*Example 4-9 Usage example of the PDTnet Queries Conformance Point (sequence of Item\_selection\_query, Item\_detail\_query and Item\_traversal\_query)*

---

```
List<PLM_property> properties = new List<PLM_property>();
// set some PLM_property

Item_selection_query itemSelectionQuery = new Item_selectionquery();
itemSelectionQuery.setId("42");
itemSelectionQuery.setVersion_id("A");

PLM_core_container result = generalConnection.query(itemSelectionQuery,
properties.toArray());

Item_detail_query itemDetailQuery = new Item_detail_query();
// get UID of first Design_discipline_item_definition:
String ddidUid = result.getItem()[0].getItem_version()[0].
    getDesign_discipline_item_definition()[0].getUid();
itemDetailQuery.setUid(ddidUid);
itemDetailQuery.setAdd_properties(true);

result = generalConnection.query(itemDetailQuery, properties.toArray());

Item_traversal_query itemTraversalQuery = new Item_traversal_query();
itemTraversalQuery.setUid(ddidUid);
itemTraversalQuery.setInverse_direction(false);

result = generalConnection.query(itemTraversalQuery, properties.toArray());
```

---

## 4.4 How to use PLM Services for the product structure traversal

The Top-down traversal of a part or document defines its underlying structure to include the documents associated to a part. This covers the versions, classification information, org-data (creator, updater, creation date, approval, approval date, alias IDs, and so forth), document files and the transformation information.

**Note:** See Chapter 7, “Implementing PLM Services 2.0-based solutions on IBM middleware” on page 177 for more information.

All this data may be retrieved from the PLM server in several steps, depending on the combination of specific queries, or the use of the PDTnet queries (traversal and detail queries). A retrieval of a complete (multilevel) structure is possible using the appropriate recursive query.

Example 4-10 demonstrates how to put the versions of a part master together with some org data, using specific queries:

*Example 4-10 Proxy query*

---

```
<query xsi:type="ns2:Proxy_query"
xmlns:ns2="http://www.omg.org/PLMServices1.0/Services" xmlns="">
  <UId>i--4711/UId>
  <Next_query xsi:type="ns2:Item_version_query"/>
    <Next_query xsi:type="ns2:Associated_approval_query">
      <Relating_type_name>Item_version</Relating_type_name>
    </Next_query>
    <Next_query xsi:type="ns2:Alias_identification_query">
      <Relating_type_name>Item_version</Relating_type_name>
    </Next_query>
    <Next_query xsi:type="ns2:Associated_person_organization_query">
      <Relating_type_name>Item_version</Relating_type_name>
    </Next_query>
    <Next_query xsi:type="ns2:Associated_date_time_query">
      <Relating_type_name>Item_version</Relating_type_name>
    </Next_query>
    <Next_query xsi:type="ns2:Design_discipline_item_definition_query"/>
  </Next_query>
</query>
```

---

The complete use case (as shown in Figure 4-5) is comprised of login to the PLM systems if necessary, identifying the top level product or document structure element, and traversing down the structure.

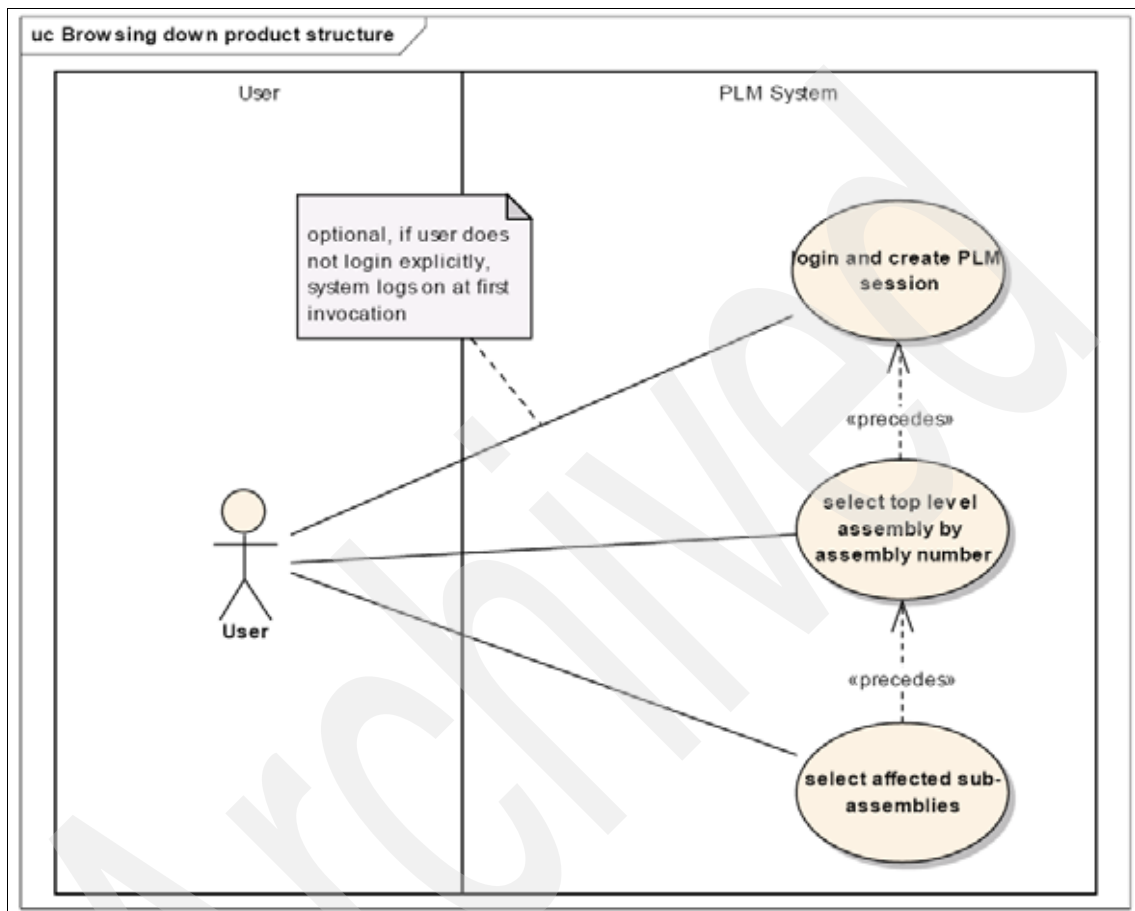


Figure 4-5 Use Case diagram for browsing down the product structure

## 4.5 How to use PLM Services in the ECM scenario to synchronize the SAP BOM

The PLM Services Informational model, as provided in 4.2, “Using the PLM Services Informational Model” on page 71, describes the entities subject to engineering change. The standard specification covers the other aspects of performing the engineering change like workflow and messaging, which are detailed in this chapter.

In this scenario, the access to PLM data is embedded in a workflow comprised of the necessary steps to manage an engineering change (for example, propose the change, comment on the change and approve the change). Every step is executed in a predefined manner to allow the revision of exchanged messages and to interpret their content in the most unambiguous manner possible. The PLM Services V2.0 supports means of asynchronously exchanging messages. The messages itself work on the informational model elements described above.

Refer to the VDA Recommendation 4965-1 for more details concerning the usage and behavior of these messages.

### 4.5.1 ECM participant proposal for a change

Embed all ECM steps needed during the initiation of a change request through a participant (supplier) that is not involved in subsequent processing of the request, excluding the coordinator (OEM). This corresponds to the Interaction Scenario 1 (IS1) of the VDA recommendation for Engineering Change Management Part 1. For example, to the following ECM messages, in this order:

1. From the Participant: Request\_initial\_ECR  
ECR\_id and ECR\_header (for example, Work\_request) of the participant plus optionally detail and comment information (activities of kind 'work definition', 'specific detail', 'comment').
2. From the Coordinator: Respond\_initial\_ECR  
Counterpart ECR\_id of the coordinator (Work\_request, Work\_request\_relationship).
3. From the Coordinator: Notify\_initial\_ECR\_accepted (or Notify\_initial\_ECR\_rejected ' end of the use case)  
Relationship between the initial ECR\_id of the coordinator to a regular ECR\_id of the coordinator (Work\_request, Work\_request\_relationship) (or acceptance of kind 'reject' to the initial ECR\_id (Activity or kind 'acceptance')).

4. From the Coordinator: Notify\_ECR\_accepted (or Notify\_ECR\_rejected ' end of the use case)  
Acceptance of kind 'accepted' (or 'rejected') to the regular ECR\_id of the coordinator.
5. From the Coordinator: Notify\_ECR\_decided  
Acceptance of kind 'accepted' (or 'rejected') to the regular ECR\_id of the coordinator (at the end of the ECR process).

#### 4.5.2 ECM participant comments

Embed all ECM steps needed during the technical and commercial evaluation and comment of an ECR by a participant (supplier) (for the parts that affect him or for which he is responsible for) that was not involved in the preceding technical analysis and detailing phase. Corresponds to the Interaction Scenario 2 (IS2) of the VDA recommendation for Engineering Change Management Part 1. For example, to the following ECM messages, in this order:

1. If the change was initiated by the Participant, the same process as IS1 has to be engaged first. Otherwise the Coordinator informs the Participant about the ECR through Notify\_ECR\_created.
2. From the Coordinator (OEM): Request\_ECR\_comments  
ECR\_id, ECR\_header (for example, Work\_request) and detail information of the coordinator (Activities of kind 'work definition', 'specific detail') plus optionally comment information (Activities of kind 'comment').
3. From the Participant: Respond\_ECR\_comments  
ECR\_id, ECR\_header (for example, Work\_request) and comment information of the coordinator plus optionally detail information (Activities of kind 'specific detail') if the comments refer to a subset of the ECR (to identify which subset is concerned).
4. (Optional) From the Coordinator: Request\_ECR\_confirmation\_comments  
ECR\_id, ECR\_header (for example, Work\_request) and all detail and comment information of the coordinator.
5. (Optional) From the Participant: Respond\_ECR\_confirmation\_comments  
ECR\_id of the coordinator and final comment from the participant to confirm his own comments or to extend them.
6. (Optional) From the Coordinator: Notify\_ECR\_decided  
Acceptance of kind 'approved' or 'rejected' to the regular ECR\_id of the coordinator (at the end of the ECR process).



### 4.5.3 ECM participant approval

Embed all ECM steps needed when the supplier (Coordinator) has full responsibility for development of certain sets of parts of the OEM and these have to be changed on the initiative of the partner (OEM, the participant). Corresponds to the Interaction Scenario 3 (IS3) of the VDA recommendation for Engineering Change Management Part 1. For example, to the following ECM messages, in this order:

Optionally:

1. From the Coordinator: Request\_ECR\_completeness\_check  
ECR\_id, ECR\_header (for example, Work\_request) and all detail and comment information of the coordinator.
2. From the Participant: Notify\_ECR\_id\_assigned  
Counterpart ECR\_id of the Participant (Work\_request, Work\_request\_relationship) and optionally an acceptance (if the participant rejects the ECR).
3. From the Participant: Respond\_ECR\_completeness\_check  
ECR\_id of the coordinator and an acceptance of kind 'is\_complete' or 'is\_not\_complete' with (optionally) completed comments and details.
4. From the Coordinator: Request\_ECR\_acceptance  
ECR\_id, ECR\_header (for example, Work\_request) and all detail and comment information of the coordinator.
5. From the Participant: Notify\_ECR\_id\_assigned (only if not already performed during step 2)  
Counterpart ECR\_id of the Participant (Work\_request, Work\_request\_relationship) and optionally an acceptance (if the participant rejects the ECR).
6. From the Participant: Respond\_ECR\_acceptance  
Acceptance of kind 'accepted' or 'rejected' to the regular ECR\_id of the coordinator
7. From the Coordinator: Notify\_ECR\_decided  
Acceptance of kind 'approved' or 'rejected' to the regular ECR\_id of the coordinator (at the end of the ECR process). The ECR can decide the change even if it has been rejected by the participant.

#### 4.5.4 ECM participant detailing and comments

Embed all ECM steps needed when a joint engineering change process is needed (the OEM is the Coordinator, the supplier the Participant). Corresponds to the Interaction Scenario 4 (IS4) of [16], for example, to the following ECM messages, in this order:

1. Either the Coordinator (Notify\_ECR\_created) or the Participant (Request\_initial\_ECR, same process as IS1) requests the creation of the ECR.
2. From the Coordinator: Request\_ECR\_details  
ECR\_id, ECR\_header (for example, Work\_request) and optionally already available detail information of the coordinator (Activities of kind 'work definition', 'specific detail').
3. From the Participant: Respond\_ECR\_details  
ECR\_id, ECR\_header (for example, Work\_request) and detail information of the participant as his result to the technical analysis.
4. From the Coordinator: Request\_ECR\_comments  
ECR\_id, ECR\_header (for example, Work\_request) and detail information of the coordinator (Activities of kind 'work definition', 'specific detail') plus optionally comment information (Activities of kind 'comment').
5. From the Participant: Respond\_ECR\_comments  
ECR\_id, ECR\_header (for example, Work\_request) and comment information of the coordinator plus optionally detail information (Activities of kind 'specific detail') if the comments refer to a subset of the ECR (to identify which subset is concerned).
6. From the Coordinator: Request\_ECR\_confirmation\_comments  
ECR\_id, ECR\_header (for example, Work\_request) and all detail and comment information of the coordinator.
7. From the Participant: Respond\_ECR\_confirmation\_comments  
ECR\_id of the coordinator and final comment from the participant to confirm his own comments or to extend them.
8. From the Coordinator: Notify\_ECR\_decided  
Acceptance of kind 'approved' or 'rejected' to the regular ECR\_id of the coordinator (at the end of the ECR process). The ECR can decide the change even if it has been rejected by the participant.

## 4.5.5 Usage examples

The examples presented in this section describes the usage of the following operations in context of ECR (Engineering Change Request) messages (see VDA Recommendation 4965-1 for Engineering Change Management Part 1):

- ▶ write\_messages()
- ▶ query\_messages()
- ▶ delete\_messages()

ECR\_context and ECR\_message\_by\_property\_query are defined in InformationalECR.xsd and ComputationalECR.xsd. Both XML schema files extend the OMG PLM Services 2.0 by ECR features.

*Example 4-11 Usage of the operation write\_messages() of the PLM\_message\_connection*

---

```
PLM_message plmMessage = ecrMessage.getPlmMessage();
plmMessage.setPLM_core_container(plmContainer);

((ECR_context)plmMessage.getPLM_context()).setMessage_name("Request_initial_ECR");
((ECR_context)plmMessage.getPLM_context()).setOrganization_id(organizationId);
((ECR_context)plmMessage.getPLM_context()).setOrganization_id_type("DUNS_number");
((ECR_context)plmMessage.getPLM_context()).setInteraction_id(interactionId);

//...

PLM_processing_information[] resultList = messageConnection.write_messages(
    new PLM_message[] {plmMessage});
```

---

*Example 4-12 Usage of the operation query\_messages() of the PLM\_message\_connection*

---

```
PLM_message_query messageQuery = new ECR_message_by_property_query();

messageQuery.setMessage_name("Respond_initial_ECR");
messageQuery.setOrganization_id(organisationId);
messageQuery.setOrganization_id_type("DUNS_number");
messageQuery.setInteraction_id(interactionId);

//...

PLM_message[] messages = messageConnection.query_messages(messageQuery);
```

---

*Example 4-13 Usage of the operation delete\_messages() of the PLM\_message\_connection*

---

```
PLM_context plmContext = new ECR_context();

((ECR_context)plmMessage.getPLM_context()).setMessage_name("Respond_initial_ECR");
((ECR_context)plmContext).setOrganization_id(organizationId);
((ECR_context)plmContext).setOrganization_id_type("DUNS_number");
((ECR_context)plmContext).setInteraction_id(interactionId);

//...

PLM_processing_information[] resultList = messageConnection.delete_messages(
    new PLM_context[] {plmContext});
```

---

# Implementing end-to-end Business Process Management

This chapter discusses and positions Business Process Management (BPM) in the context of the business problems of the fictitious Trucks Inc. (see Chapter 6, “Achieving business flexibility with a service-oriented architecture” on page 121). BPM is a strategic business decision that a company needs to address. Trucks Inc. is not familiar with end-to-end BPM. Some documentation exists for specific procedures, but these are paper-based and have no visibility of process execution time or cost. In addition, they are never updated and no one in the organization has formal responsibility for ensuring process quality and integrity. The point-to-point flows of data between applications are actually islands of bigger business processes that are not controlled by an IT component. Trucks Inc. has been educated on the benefits of BPM and has made investment in BPM tools and methods as part of their strategic restructuring activities.

One of the first topics of discussion with Trucks Inc. was to ensure they had a clear definition of a process. Processes are used to bridge the gap between the demands of the business to deliver the product, service, or ultimate deliverable and the technology that is used to support that delivery. A process can be described as a set of linked activities that take an input and transform it to create an output. Ideally, the transformation that occurs in the process must add value to

the input and create an output that is more useful and effective to the recipient either upstream or downstream. The BPM model can be considered as a multidimensional representation of reality capturing a moment in time. The model has purpose, perspective, audience, content, level of detail and phases. It is used to summarize information and convey messages. The process model includes a detailed specification of all activities within the process as well as key performance indicators.

This definition also emphasizes the linkage between activities and the transformation that takes place within the process. The characteristics of a process are as follows:

- ▶ **Definability**  
It must have clearly defined input and output boundaries.
- ▶ **Order**  
It must consist of activities that are ordered according to their position in time and space.
- ▶ **Customer**  
There must be a recipient of the process' outcome.
- ▶ **Value-adding**  
The transformation taking place within the process must add value to the recipient, either upstream or downstream.
- ▶ **Embeddedness**  
A process cannot exist in itself. It must be embedded in an organizational structure.
- ▶ **Cross-functionality**  
A process regularly can span several functions. It is not required to do so.
- ▶ **Process ownership**  
A process must be owned. A person must be responsible for the performance and continuous improvement of the process.

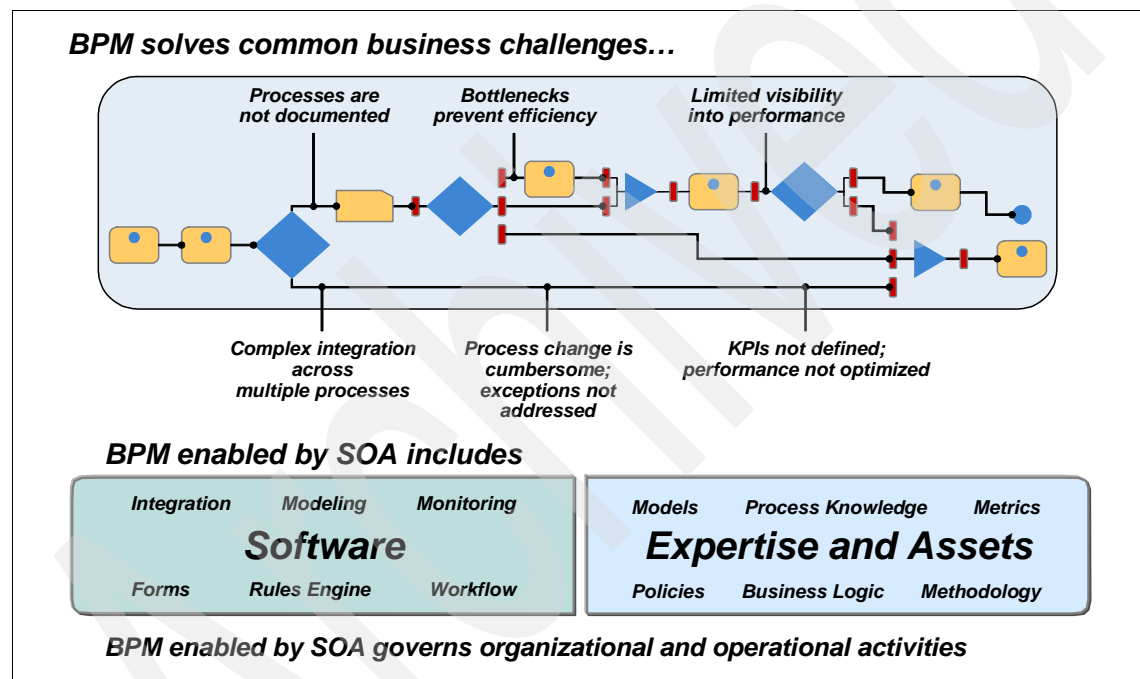
In addition, processes are defined at multiple levels to differentiate between strategic direction, business execution, and technical management. Opinion within the BPM community over how many levels of process exist (As well as, the names and intent of each level) varies but one example is as follows:

- ▶ **Level 1: Enterprise View**  
A business described in terms of its major functions or categories of activity. Boundaries are typically drawn based on the use of personnel, resources, and accountability.

- ▶ Level 2: Business Function View  
A component of a Level 1 view described in terms of its major functions or categories of activity.
- ▶ Level 3: Process  
A major business function or category of activity, a component in a Level 2 view.
- ▶ Level 4: Sub-process  
Actions needed to accomplish a Level 3 sub-process. Level 4 defines the conceptual approach to accomplishing the sub-process and defines the best practice for completing the sub-process.
- ▶ Level 5: Activity  
Describes activity groups, activities, and work steps comprising one or more actions included in a sub-process. This is the most-detailed level of process models and includes the capture of services and integrations.
- ▶ Level 6: Work Steps  
Describes details of work steps comprising each work step. Work steps are defined in sufficient detail to permit validation against system requirements.
- ▶ Level 7: Step Scripts  
Work step-related detailed decision tables, test scripts, configuration settings and detailed work step narratives. End-user materials, help windows, desk materials, and other documentation in support of education and training.

## 5.1 Business Process Management

BPM is a discipline combining software capabilities and business expertise to accelerate process improvement and facilitate business innovation. BPM, enabled by service oriented architecture (SOA), provides a flexible architectural style in support of efficient process change and rapid process deployment. For success in BPM projects, it is critical not only to use SOA-enabled software, but also to possess the expertise to deliver and fulfill the promise of BPM. BPM has evolved to address the challenges of the modern business, recognizing that business methodology, organizational efficiency, and software are needed to address the process needs of today's business.



*Figure 5-1 BPM addresses challenges related to the capture, analysis adoption and execution of business processes by leveraging world leading BPM applications, methods, standards and highly skilled professionals*



Ultimately, the benefit to Trucks Inc. of adopting BPM are as follows:

- Process insight and optimization

The first step in many engagements is to monitor what is happening. Understanding what is happening inside the business facilitates the ability to enhance the most important and impactful parts of an organization.

- Accelerated process improvement

It is not just about improvement and optimization. It is about how fast you can identify the parts of the business that drives change and how fast you can implement and deploy those changes to make the improvement happen.

- Flexible design for future change

It is critical not only to make the changes once, but to be prepared for the future changes that every organization faces. This is referenced in Chapter 6, “Achieving business flexibility with a service-oriented architecture” on page 121 where the business flexibility challenge is addressed in detail.

Before we address the adoption of BPM in Trucks Inc., let us review the condition of their business infrastructure. The first observation is that it is heterogeneous and complex. This complexity is increased due to the company merger that resulted in the formation of Trucks Inc. It is characterized by a spectrum of business applications for point problems, multiple technologies, and specialized skill sets required to keep the business up and running. Another observation is that it is difficult to obtain contextually relevant business information residing in disparate IT systems. Given this business infrastructure, it is almost possible for the corporate IT to keep up with changing business conditions.

Indeed, it is difficult for IT to keep pace. Worse, a chasm has developed between lines of business and IT in terms of IT's ability to create or change applications under these conditions. This problem is compounded due to the organizational merger that created Trucks Inc.. As external economic factors, customer switching ability, regulatory pressure, and product and service development commingle and drive the need for greater process flexibility. IT is often trapped in operations and existing application maintenance and unable to respond in a timely manner. It is unreasonable for organizations to delay progress while waiting for developers to change business processes embedded in core applications. That said, there is an understandable degree of ownership that IT must maintain relative to the availability and auditability of core applications and infrastructure elements. Thus, there must be a means for IT and lines of business to collaborate effectively to accomplish business goals. There also must be a way for each entity to manage and change its environment without impacting the other. This multirole and multifaceted capability set, as well as the flexibility to enable business users to create or change processes and maintain visibility into them, is at the heart of BPM.

### 5.1.1 Adopting process standards

To aid the adoption of BPM methods, Trucks Inc. reviews the structure and content of industry standard processes. There are a number of sources of standard processes that may be of use to Trucks Inc. For execution of the engineering change process as a technical level, they may look to the Verband der Automobilindustrie (VDA) organization. VDA provides support for ECM processes with respect to the communication of change data between customers and suppliers using standards such as ISO10303-214 (STEP AP214) and PLM Services from the OMG. STEP AP214 was defined in Chapter 3, “Product Lifecycle Management Standards overview” on page 31. From a business process perspective, Trucks Inc. may consider the Value Chain Operations Reference (VCOR), Supply Chain Operation Reference (SCOR), or Design Chain Operations Reference (DCOR) models, although these are typically focused around the core product development domain. To expand the horizon of BPM adoption though, it would be recommended that Trucks Inc. use an industry-specific process classification framework (PCF). The original generic PCF was created by the American Productivity and Quality Council (APQC) organization. For more information about APQC, see the following Web site:

<http://www.APQC.org>

This generic PCF was created by APQC with support from partners including IBM. Subsequent industry-specific models (including automotive) have been created by the APQC partners and are available as open source standards. These PCF's provide a hierarchical structure of processes, commonality of terms for tasks, decisions, and roles, as well as detailed process models to the equivalent of the Level 5 definition mentioned earlier in this chapter.

In many cases, patterns are used in the definition of the processes. These patterns are repeatable groupings of process tasks and decisions that are re-used many times during the whole end-to-end process framework. An example of a process patterns would be the review cycle. Reviews happen through the product development cycle, from the time someone identifies a market opportunity for a new truck to the point where the hazardous materials are recycled. The participant, data, and location of the review may vary, but the core steps are the same.

1. The need for a review is identified
2. The data to be reviewed is collated and distributed to the review team
3. The review team make an assessment on the next steps (including if they are able to define the next steps themselves or need outside influence)
4. The next steps are actioned.

An example of this review pattern is shown in Figure 5-2 on page 109.

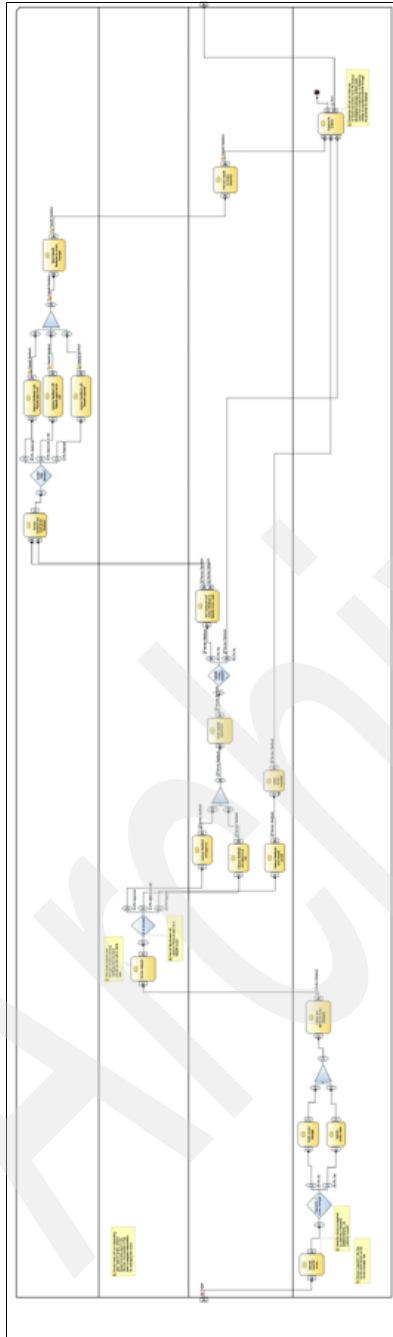


Figure 5-2 Review pattern example

In addition to the processes themselves, the PCFs include a comprehensive suite of key performance indicators (KPIs). By adopting these KPIs, if Trucks Inc.. embraced the PCF and subscribes to the APQC organization, they is able to benchmark their effectiveness efficiently compared to other members of the standards organization. As part of the adoption of BOM, Trucks Inc. is legally obliged to demonstrate capability in certain processes for homologation and regulatory compliance. By using an industry PCF tailored to suit Trucks Inc..'s specific process needs, the challenge of illustrating compliance is greatly simplified.

### **5.1.2 Demand for process flexibility and innovation with control**

Notably, no two organizations, even those in the same vertical market, are exactly alike. While they may possess business processes that are fairly consistent at a macro level within certain markets (managing customer bills for instance) every company or organization adds its own nuances and tailors its version of those processes to reflect its own business requirements. For regulated markets, business processes must also be auditable, and once again there are varying degrees of regulation impacting organizations across markets. In parallel though, organizations must innovate within their business practices, whether that means finding ways to improve service or cutting cycles out of product development to reach the market more quickly with high quality products and services. Chapter 6, “Achieving business flexibility with a service-oriented architecture” on page 121, discusses how this flexibility in the processes is realized through the technical services that enable them.

### **5.1.3 A wide range of requirements to address**

As a result of this mix of individual business best practices from the two former companies, industry standard processes such as those derived from the APQC, regulatory requirements for safety and environmental considerations, and the drive to innovate, BPM solutions must address a wide variety of process types, constituents, participants, and performance needs. Needs range across the following situations:

- ▶ Processes that require multiple applications to be tied together so that data can be transacted with appropriate controls among them
- ▶ Applications requiring documents to be persisted and declared as records in lockstep with (or as a result of) business actions
- ▶ The ability to handle exceptions and drive resolution from within structured processes
- ▶ Integrating shared areas in which team members can work together on projects and related materials

### 5.1.4 Addressing complexity: processes and participants

All of these needs exist within Trucks Inc. and they must be linked together to address more complex operations. They must also be monitored and audited based on industry and business regulations. BPM solutions must enable organizations to capture and easily change the critical elements of processes:

- ▶ The routes information must take among people and systems
- ▶ The roles that participate in the processes
- ▶ The rules and policies that govern these interconnections

The variety of roles that participate in these processes must also be addressed. This includes line of business process owners, IT professionals providing integration and tailored features, managers and executives that only see dashboard representations of performance across processes, and business analysts that may model and simulate processes. BPM must offer tailored experiences supporting all of the above circumstances and participants.

### 5.1.5 Maintaining a business user focus

Despite the potential for complexity, BPM must deliver the appropriate information with which to make decisions in a simple and straightforward fashion. This begins with the ability to model, simulate, and modify processes. For processes that require people to make decisions or for situations in which people may be required to weigh in on an automated process, delivering the information in context of business use is paramount. For BPM solutions, this means addressing the appropriate business roles, obtaining the effective information required to drive a decision, and enabling the user to interact with it before moving on. In some situations, the ability to present the right application or information dynamically (for example, without the process knowing ahead of time which particular application or service to call upon) to a business user is a requirement. Further, when a variety of information within and external to a process is required for a decision, portal and, ultimately, Web 2.0 technology is employed to deliver a unified and contextually relevant view to the business user.

### 5.1.6 BPM: Tactical versus strategic approaches

In starting the BPM activities in Trucks Inc., the Board has decided that they will implement BPM to address specific pain points in a single business area of the newly merged company. Even in those cases, however, a BPM solution cannot be an island unto itself. It must integrate with other applications, it must be flexible enough to allow frequent changes to processes, and it must provide the right context for the people working with it. At the same time, however, BPM

cannot sacrifice process integrity, and it must easily integrate with information management facilities, be they analytical or content related, to ensure that users are presented with what they need in appropriate context to make decisions.

At the point when Trucks Inc. tackles the larger strategic objectives of a full BPM implementation, all of the above requirements apply, but a chosen BPM solution set or portfolio must also provide interoperability among its component parts, providing leverage for users, managers, developers and other participants. Providers must also offer a broad array of capabilities addressing the span of processes in organizations, be they collaborative or transactional, structured or dynamic. Increasingly, the ability to optimize processes both from within the confines of the process (through business activity monitoring or analytics about process performance and infrastructure performance) and with respect to related applications or events outside the process is a requirement.

### **5.1.7 Dynamic processes, BPM and SOA**

The BPM relationship with SOA must be exploited both to enhance dynamic requirements within processes and to reflect the ability of infrastructure and integrated applications to be easily modified with changes to organizational, market, and business requirements. Imagine creating a financial account opening process flexible enough to determine in real time which of several offers is appropriate as a new customer provides information about his status and as external systems help determine his creditworthiness that can be changed simply by re-modeling the process to take advantage of new services and to follow changed business rules. And as processes become more complex along with the dynamics of integration, the ability for those processes to either react to changes in information and content or easily obtain information for automated or human decisions becomes ever more critical.

### **5.1.8 BPM and Enterprise Architecture (EA)**

In the adoption of BPM at Trucks Inc., BPM must be considered from an architectural standpoint, particularly given the organizations' ever increasing need to move toward SOA and the related requirement for a supporting information infrastructure. EA discipline suggests that there are separate but related business, information and technical architecture components. Thus, a BPM solution must comply with technical infrastructure and platform requirements while also providing business flexibility and links to information management standards.

Thus, organizations must consider capabilities and providers offering the following features:

- ▶ **Multiple BPM Entry Points**

A broad set of interoperable, though separable, BPM capabilities open enough to fit current and future architecture, platform, application, process and information demands is crucial. Separability is key because not every situation demands every facet of BPM technology. Some organizations, for example, determine which processes to optimize (and how to do so) after monitoring them for a time. Others have identified processes to innovate given changing business requirements.

- ▶ **Ability to Address Multiple Business Scenarios**

BPM technology that can address a broad span of processes, from collaborative to transactional, while the appropriate contextual focus is delivered to every type of user (business, IT, managerial, and so forth) involved in any stage of the process is important. Processes may be initially specified according to a primary characteristic (for example, online account opening) but can require different BPM capabilities for secondary requirements (for example, customer exceptions requiring documentation to be captured for regulatory compliance). Providers must offer this range of capability and provide appropriate interoperability within it.

- ▶ **Flexibility and the ability to change processes policies and rules**

Flexibility is a major benefit of BPM technology, and solutions must reflect that throughout their platforms. This means process modelling that integrates with development tools and standards while providing and reacting to information for process simulation and optimization. It also entails the ability to link dynamically and choreograph external services and other processes to address more complex business situations. The ability to include and enact collaborative and team-oriented capabilities to resolve exceptions or to tackle more casual process issues must be addressed.

### **5.1.9 Addressing specific business needs with BPM**

To drive a tactical advantage of adopting BPM as a pragmatic solution to some of their key business challenges, Trucks Inc. must ensure their BPM solutions offer a full set of integrated capabilities ranging from modelling through monitoring while addressing the key processes identified by the organization for optimization (or even for simple automation). On the point of optimization, the ability to take advantage of analytics derived directly from a process, as well as those external analytics related to a process (for example, regional sales data that could affect supply chain processes), are becoming more critical, as is the ability to establish and measure process KPIs on an ongoing basis. The ability to remediate a process or drive policies that can address dynamic business requirements

specified through analytics will become a hallmark of the next generation of BPM. Tactical BPM approaches must also perform a delicate balance of addressing processes for which they are optimized, while providing enough leeway and interoperability to broaden their process scope or address exceptions.

### **5.1.10 Expertise**

It is critical to gain a jump-start on process optimization through the availability of industry expertise, be it in the form of pre-established, standards-based roles and KPIs, industry benchmarks, or vertically-oriented services. The ability to call upon predefined data models or templates reflecting appropriate personas, security, and capabilities not only speeds implementation but adds to usability and ensures commonality and ease of compliance with industry expectations. Given the imperative to empower business users within the organization through BPM, the ability to address business needs quickly and easily by delivering familiar and accepted user environments is paramount.

### **5.1.11 Empowering LOB and IT to work together, with BPM enabled by SOA**

While IT is equipped to solve complex technical problems, IT does not know how a process is best optimized, or which key performance indicators must be tracked to track and tune business performance. Until recently, lines of business, while aware of key business processes, have not been empowered to change business processes as conditions change, or in anticipation of change.

The use of innovative BPM software by Trucks Inc., however, is pointless unless they adopt a proven methodology through which their BPM content is created and deployed to drive business-led change.

## **5.2 Implementing Business Process Management**

One of the challenges in realizing BPM is the need to balance the top-down approach to addressing business level challenges with the bottom-up approach of developing services. To address this challenge, separating the Trucks Inc. processes into their constituent components, and a similar separation of the applications, is required, with agreement reached by both technical and business architects in the PLM domain.



Component Business Modelling (CBM) is a proven method that allows organizations to identify opportunities for improvement and innovation by regrouping their activities into a manageable number of discreet, modular, and reusable components. CBM may be used in driving top down development activities. This enables flexibility and provides for a clarified focus on the core capabilities needed to run the business and drive strategy.

Key aspects of CBM areas are as follows:

- ▶ A business component is a part of an enterprise that has the potential to operate semi-independently (for example, it has a defined and declared set of inputs and outputs and can run with limited additional outside influence). A business component is a logical view of part of an enterprise that includes the resources, people, technology, and know-how necessary to deliver value. It has a high degree of autonomy, is managed separately, and is linked to the rest of the organization through business services and integrated information systems.
- ▶ A business component map is a tabular view of the business components. The map delivers a top-down view on the business.
- ▶ A business component model is a term for the business competencies, business components, business services, and their relationships that describe an enterprise or industry.

Standard Component Business Model™ templates for automotive heavy trucks industry have already been created and form the basis of the CBM assessment with Trucks Inc. These templates are used with the customer as a starting point for discussion, which can then be tailored specifically to the way the customer works. Because the CBM further divides into a suite of industry processes and sub-processes (Figure 5-3 on page 116), as well as a suite of KPIs, it is possible to use this model to identify the parts of the business processes that are targets for improvement.

	Business Administration	Financial Management	Product / Process	Production	Supply Chain	Marketing & Sales	Service & Aftersales
Direct	Corporate/LOB Strategy & Planning	Financial Planning & Forecasting	Portfolio Strategy & Planning	Production Strategy	Supply Chain Strategy & Planning	Customer Relationship Strategy	Post Sales Strategy
	Organization & Process Policies	Capital Appropriation Planning	R&D/ New Products	Master Production Planning	Demand Planning	Sales & Promotion Planning	Service Contracts
	Alliance Strategies		Design Rules & Policies	Production Rules & Policies	Supplier Relationship Planning	Brand Management	Supplier Recovery
Control	Human Capital Management	Risk management & Internal Audit	Program Management	Production Scheduling	Supply Chain Performance Monitoring	Relationship Monitoring	Failure Analysis
	Legal & Regulatory	Treasury	Configuration Management	Production Monitoring	Supplier Management	Demand Forecast & Analysis	Quality Management
	Business Performance		Design Validation				
	Intellectual Property	Tax Management	Change Management	Quality Control	Logistics Management	Dealer Management	Parts Management
Execute	Knowledge & Learning	Accounting & General Ledger	Mechanical/ Electrical Design	Plant Operations	Inventory Management	Finance/ Lease Management	Claim Entry & Administration
	Building/ Facilities & Equipment		In-vehicle System Design		Transportation Management	Order/ Configuration Management	Service
	IT Systems & Operations	Cost Management	Process Design Tool Design & Build	Maintenance Management	Procurement	Customer Relationship Management	Body Builders

Figure 5-3 Component Business Model suite of industry processes and sub-processes

Implementing BPM is much more than simply identifying opportunities for process improvement. As with the implementation of any new strategic direction, consideration must be given not only to the processes or new technology but also to the people who are going to execute the process. When adopting BPM, Trucks Inc. has to look at the organizational change management implications of what they intend to do, these include the following implications:

- Organizational structuring

The adoption of new processes and technology may affect the way the current organization is structured. As part of BPM, the organizational effectiveness in its current state and its predicted efficiency for a desired condition is assessed and a recommendation is made for vital organization restructuring.

- Education and Training

BPM adoption creates the need for additional training and education to the organization: education being the instruction on WHY a given role is performing a task, and training being exactly HOW to do it.

In moving from the As-Is to the To-Be BPM state, most companies go through a four stage implementation approach:

- ▶ Proof of Concept

Capturing the intent of the new process and testing the new process and new technologies for a short period to capture an indication of how much better, cheaper, or faster the new process and tools are. This PoC phase is usually delivered in a stand-alone environment on test data.

- ▶ Pilot

Testing the new tools and processes a little further, significantly loading the tools and processes with volumes of data and more frequent execution. Usually executed on a pre-production IT environment with copies of live databases.

- ▶ Implementation

Full scale deployment of the new tools and processes to the user community on live data.

- ▶ Support

User support to ensure seamless adoption of the new tools and processes and to gather feedback from the wider user community.

To put these four steps in SOA terms, we consider the implementation of BPM as the following phases

- ▶ Model phase

Capture the current state static process model. This as-is model gives a view of how much the current process costs and how long it takes to execute. Customer-specific, dynamic to-be process states are created based on the APQC Process Classification Framework industry processes, and again apply representative time and cost figures. The comparison of the as-is to the to-be simulations results gives the return on investment and business case to justify expenditure on BPM by the customer.

In addition to a streamlined and effective business process, in the to-be model the KPIs are captured, which is used later in the cycle to monitor how well the process is running. The hand off of the business to the IT architect at the end of the model phase is a Business Process Execution Language (BPEL) model.

- ▶ Assemble phase

The IT architects implement the BPEL export from the model phase. Developers are then able to fine-tune the execution of the process, implement policies and rules based on the integration platform, and test the implementation thoroughly before deployment to the production environment.

- ▶ **Deploy phase**  
Process Execution and Automation (workflow and choreography), including human task management, are important elements of a business process. Execution occurs with the deployment of the results of the Assemble activities on a process engine preferably running on an ESB and secure application server and can include integration with Content Management. In all business processes, information is either created or consumed as work progresses. Content is the primary object to be manipulated in a business process. Therefore, participants in a process need the ability to create new content, as well as the ability to access and use existing content. Having the right information at hand at the right time is critical to process success.
- ▶ **Monitor phase**  
Business Activity Monitoring is the ability to monitor process performance and detect events that may influence performance. Analyzing process efficiency and effectiveness and aligning process improvement with enterprise goals and objectives involves using software technology such as WebSphere Business Monitor to listen for critical business events, correlate event data, and update KPIs captured during the Model phase.

### 5.2.1 BPM tooling

In the creation and deployment of BPM content, Trucks Inc. has been recommended the following suite of WebSphere products:

- ▶ **IBM WebSphere Business Modeler**  
This product provides process modeling, simulation, and analysis capabilities to help business users visualize, understand, and document business processes for continuous improvement. It enables business users to design, model, and document vital business processes in addition to modeling resources, roles, organization, information, and business metrics.
- ▶ **IBM WebSphere Integration Developer**  
This product is used in the development, assembly, and testing of applications and integrations resulting from business process optimization prior to deployment in the product environment. It is a common tool for building SOA-based integration solutions across WebSphere Process Server, WebSphere ESB, and WebSphere Adapters.
- ▶ **IBM WebSphere Process Server**  
This product manages the execution of the processes, integrations, and applications defined in WID through automated and human tasks.

► IBM WebSphere Business Monitor

This product provides end-to-end visibility of business performance through the following features:

- Customizable dashboards
- Early detection and alerts to emerging business situations
- Complete feedback mechanism with WebSphere Business Modeler for process improvement
- Near real-time information displayed on customized dashboards

These features provide the ability to make timely changes and advanced analytics for decision-making based on historical and trending data. In addition to processing events received from event sources, WebSphere Business Monitor can pull additional business reference information into the Business Monitor server. When integrated with WebSphere Integration Developer and WebSphere Process Server, WebSphere Business Monitor can track the activity of the full range of components running in WebSphere Enterprise Service Bus and WebSphere Process Server.

► IBM WebSphere Business Services Fabric

This product enables business processes to dynamically change and adapt according to changing circumstances (for example, the use of alternate suppliers at various parts in the development cycle or being able to change the process according to a set of business rules). See Figure 5-4 on page 120.

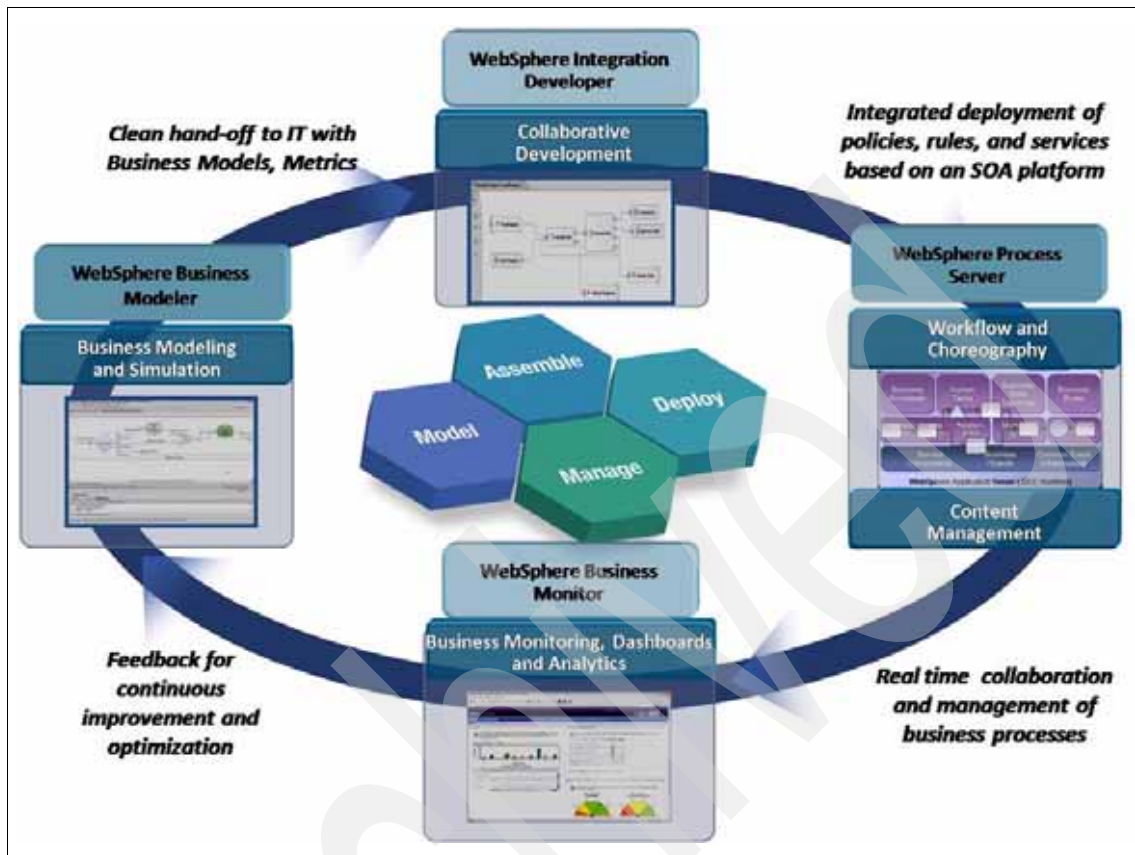


Figure 5-4 A graphical representation of the linkage between the four key phases of a BPM for SOA implementations as well as the correlating tools

# Achieving business flexibility with a service-oriented architecture

This chapter presents a fictitious company known as Trucks Inc. The company is fictitious, but the story about how service-oriented architecture (SOA) enables rapid change and transformation is real. This chapter discusses how Service Design focuses on aligning a company's business design and IT solution design with proven methods and tooling. You will read and understand how SOA governance helps to define the roles and responsibilities in an organization.

This Redbooks publication seeks to resolve the following basic PLM pains:

- ▶ Typical tasks and activities of PLM processes take a long period of time requiring process integrity for long running transactions
- ▶ The PLM data used and changed requires specific applications and editors for processing. A heterogeneous system landscape is inevitable
- ▶ The protection of intellectual property is of critical importance, as is external collaboration with design partners
- ▶ The product data is needed during the product development processes by different participants in different formats and granularity
- ▶ Product data is managed in complex large structures with a high level of semantics

## 6.1 Achieving business flexibility with a service-oriented architecture

Trucks Inc. is at the beginning of an enterprise transformation initiative. Trucks Inc. understands the key problems that keep them from reaching their goals. The company is convinced that SOA can enable rapid change and transformation.

There are four approaches to implementing SOA, and each SOA adoption approach can build on each other, leading to full enterprise-level adoption:

- ▶ SOA-related Technology Adoption focuses on an SOA-related technology project, such as Web services or BPEL implementation.
- ▶ SOA Preliminary Deployment or Pilot Project Adoption is used to transform a single business process, such as engineering change management.
- ▶ Business Solutions Adoption addresses a company's specific business problem usually within a process or business component.
- ▶ Enterprise Adoption or Transformation is a large scale adoption that integrates business and IT at the enterprise level.

Trucks Inc. decided that they will employ the second approach to SOA adoption and decided to select the Product and Process Management, or PLM business unit (see Figure 6-1), to pilot the adoption of SOA, along with the supporting infrastructure. This project is known as the Trucks Inc. SOA for PLM project.

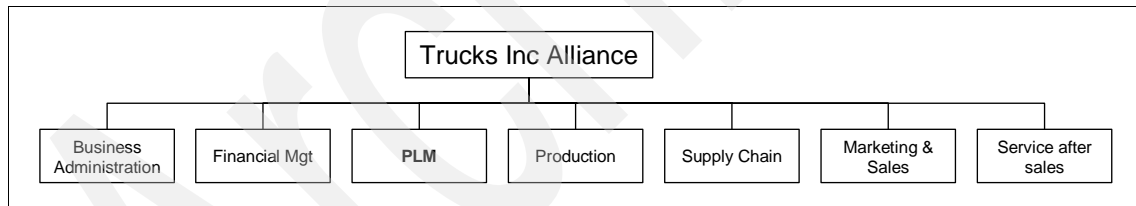


Figure 6-1 Trucks Inc. business unit selected to pilot SOA adoption

As shown in the previous chapters, in order to get ready for the transformation of this business unit, Trucks Inc. decided to prepare in the following manner:

- ▶ Adopt OMG PLM Services as the common neutral integration language
- ▶ Deploy a BPM culture as opposed to the traditional PLM-application centered approach

The solution requirements for the SOA for PLM project are categorized into focus areas that are designed to work together to achieve the business objectives.



### **6.1.1 Service Design**

The Service Design focus area aligns the modeling of business design and IT solution design through a set of roles, methods, and artifacts to render a set of explicit business processes for optimization and services for composition and integration. To align its business objectives properly with its IT goals, the PLM for SOA project team must begin by analyzing the business architecture through one of the key services methodologies from IBM, perform data architecture to create the System Data Model, and perform service analysis and design.

### **6.1.2 Service Creation**

The Service Creation focus area involves the creation of flexible, service-based business components. Trucks Inc.'s Product and Process business unit identifies high-value existing IT assets and enable these assets for reuse. The solution exposes existing backend systems, such as PTC Windchill, DS Enovia LCA, UGS TeamCenter, MSC SimManager, SAP, and any existing PLM applications as services to be more easily consumed for reuse.

### **6.1.3 Service Connectivity**

The Service Connectivity focus area links people, processes, and information within the business and the extended enterprise through a gateway or Enterprise Service Bus. The solution enables the SOA for PLM processes to access multiple channels, access back-end applications, federate organizational units, and provide business-driven service availability.

### **6.1.4 Process**

The Process focus area is a set of business-related activities that are invoked in a specific sequence to achieve a business goal. Business processes consist of tasks, which are comprised of human interactions, an automated workflow, information services, business rule interactions, subprocesses, and the invocation of functions and services. Examples of business processes at Trucks Inc.'s Product and Process business unit are: engineering change management (ECM), Release to manufacturing, Enterprise BOM management, and so forth.

## 6.1.5 Interaction and Collaboration Services

The Interaction and Collaboration Services focus area is a portal-based solution designed to improve people productivity. For example, Trucks Inc. executives track business portfolio and product roadmaps using portal-based business dashboards. Similarly, interaction and collaboration services allows Trucks Inc. project managers to control and track efficient communication, deliverables, events, risks, agreements, and plans. Operational processes required for product development, such as requirement management, supplier relationship management, product configuration, and quality management, can also be improved with the use of specific portals. A supplier portal being a well-known example in the industrial sector.

## 6.1.6 Information Services

The Information Services focus area offers information access to complex, heterogeneous data sources within Trucks Inc. as reusable services. Trucks Inc. needs take advantage of and manage existing information. The solution can accomplish this through enabling relevant information to be shared throughout the organization and allowing real-time capabilities to access and manage both structured and unstructured information. The solution must also provide an interface to both automated and manual data cleansing functions to support post-validation corrections to information.

## 6.1.7 BPM

The BPM focus area is a discipline combining software capabilities and business expertise to accelerate process improvement and facilitate business innovation. BPM enabled by SOA provides a flexible architectural style in support of efficient process change and rapid process deployment. For success in BPM projects, it is critical to use not only SOA-enabled software, but also the expertise to deliver and fulfill the promise of BPM. The combination of software and expertise is what IBM is calls *Higher Value BPM*. BPM is covered in Chapter 5, “Implementing end-to-end Business Process Management” on page 103.

### 6.1.8 SOA Governance

The SOA Governance focus area enables a more flexible business at Trucks Inc. This flexibility needs to be managed in order to control the potential quick evolution of business services and processes.

SOA Governance embodies the management of relationships between services and other parties along with ensuring services' compliance with the laws, policies, standards, and procedures under which an organization operates. The solution must consist of a formal process for assigning decision rights, implementing policies that are derived from business rules, and instituting measures around the service processes and life cycle.

Furthermore, the solution must streamline the following processes:

- ▶ Service registration
- ▶ Service versioning
- ▶ Service ownership
- ▶ Service discovery
- ▶ Service access
- ▶ Service and composite applications deployment
- ▶ Service security

### 6.1.9 SOA Security and Management

The SOA Security and Management focus area includes security services, discovery, and monitoring of SOA resources for SOA environments. The Trucks Inc. team designs and deploys SOA security and management solutions in support of the SOA for PLM Project deployment

In the next sections, we study how Trucks Inc. implements the Service Design (6.2, "Service Design" on page 126) and SOA governance (6.3, "SOA Governance" on page 145) focus areas.

## 6.2 Service Design

Service Design focuses on aligning Trucks Inc. business design and IT solution design through the use of proven methods and tooling.

IBM methods such as CBM and Service-Oriented Modeling and Architecture (SOMA) provide the conceptual framework to define the what, how, and why of modeling to align the business and IT design. IBM tools can be used in support of the design methods to model traceability and to create the design artifacts throughout the life cycle. The SOA Service Design Scenario can be applied to each of the base SOA scenarios. The fundamental constructs of the SOA Design Scenario model (Figure 6-2) are flows, services, and components.

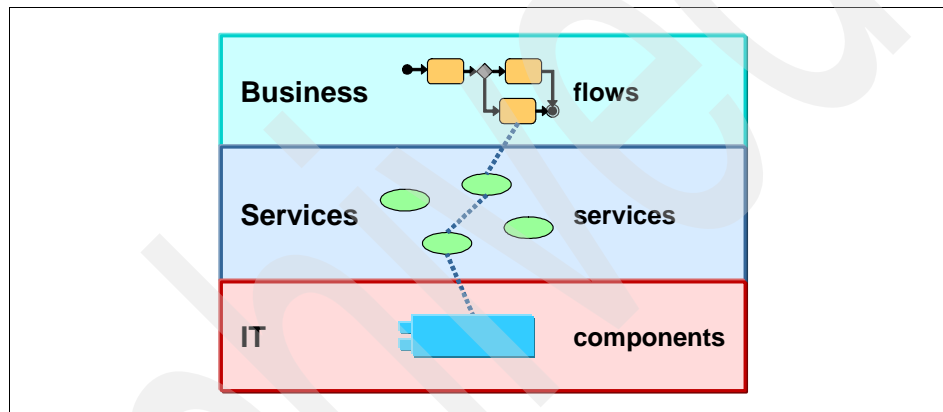


Figure 6-2 Services provide the alignment between the Business and IT

This section covers key elements of the SOMA for Service Design method:

- ▶ Service Identification
- ▶ Service Specification
- ▶ Service Realization

The SOMA Method (Figure 6-3 on page 127) is an analysis and design method that is used for the design and construction of SOA to enable target business processes. SOMA accomplishes this through the identification, specification, and realization of services, components, and flows.

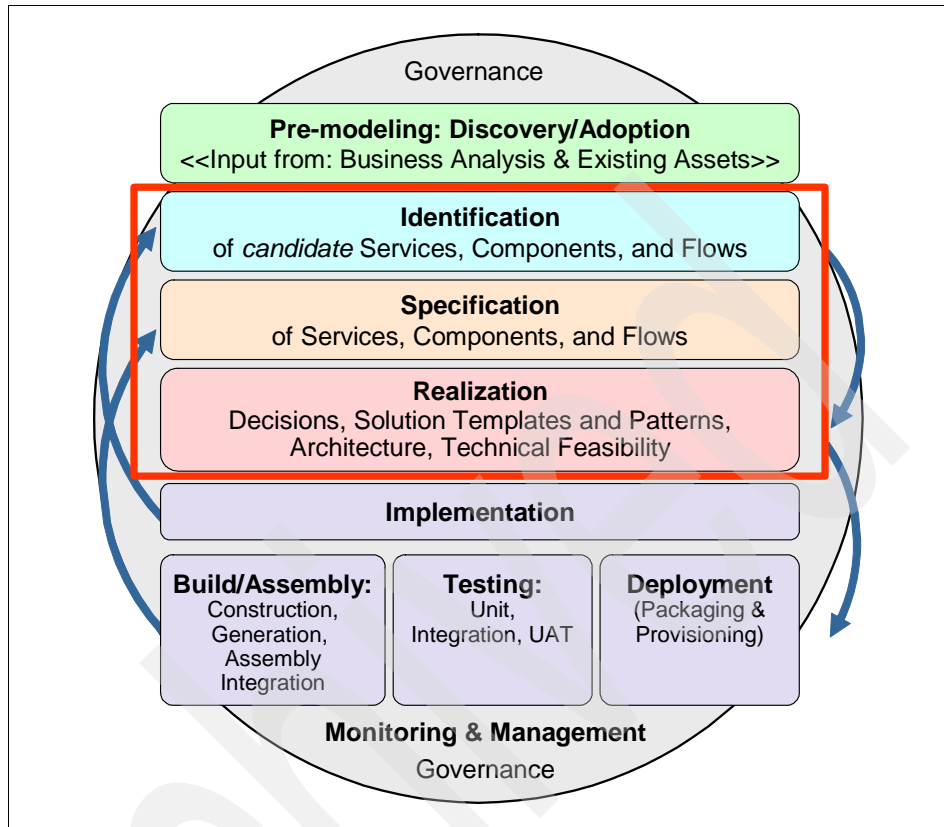


Figure 6-3 Service-Oriented Modeling and Architecture (SOMA) Method

The SOMA Method provides prescriptive guidance for SOA design (see Figure 6-4) and is the foundation for SOA solution design patterns.

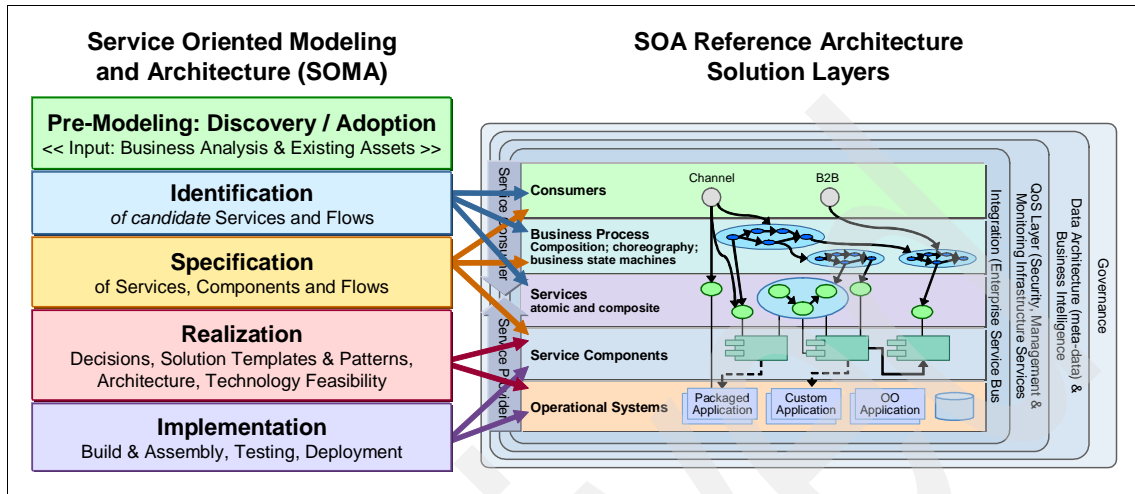


Figure 6-4 SOMA guidance for the SOA Reference Architecture layered solution

## 6.2.1 Service Identification

The goal of Service Identification is to create an initial set of candidate services and their associated operations that are meaningful to the business. Service Identification is performed primarily by software architects and often includes participation from business analysts in a supporting role.

During Service Identification, the service model work product is created and handed off to the software architect who is responsible for the service specification. Service identification is synonymous with producing the analysis level of the service model, whereas the service specification is the design level.

Key inputs to Service Identification include:

- Business analysis and modeling

Business analysis and modeling is used to define the business architecture. CBM is often used for business analysis to help customers understand their business and capabilities, and identify capability gaps. Other methods can be used as well for business analysis.

- Service registry and repository

Existing services and information about them are typically stored in a service registry and repository. Because Trucks Inc. SOA for PLM Project is the first SOA adoption, there are no existing services. Refer to 6.1.8, “SOA Governance” on page 125 to see the examples of the content of a service registry and repository for Trucks Inc. Product and Process business unit.

Let us take a closer look at the three complementary techniques that are used to identify candidate services:

- ▶ Goal-service modeling
- ▶ Domain decomposition
- ▶ Existing asset analysis

### **Goal-service modeling**

The key objective of goal-service modeling is to demonstrate traceability and alignment of services with business goals. The goal-service model is a middle-out approach, which is used to validate the completeness of the list of candidate services that are identified through the domain separation and existing asset analysis techniques when the corresponding outputs are available.

When developing the goal-service model, you typically work closely with business executives, business analysts, and subject matter experts to identify the goals of the business within the scope and phase of the project. For each goal and subgoal, you identify KPIs and metrics to evaluate business performance.

The focus of Service Identification, in the Product and Process business unit business components, are the goals that identify services to support the business component. Figure 6-5 on page 130 shows the IBM Component Business Model™ map for the automotive industry. Trucks Inc. might take this CBM map as one of the inputs of the Service Identification study.

	Business Administration	Financial Management	Product/Process	Production	Supply Chain	Marketing & Sales	Service & Aftersales
Direct	Corporate/LOB Strategy & Planning	Financial Planning & Forecasting	Portfolio Strategy & Planning	Production Strategy	Supply Chain Strategy & Planning	Customer Relationship Strategy	Post Sales Strategy
	Organization & Process Policies	Capital Appropriation Planning	R&D/ New Products	Master Production Planning	Demand Planning	Sales & Promotion Planning	Service Contracts
	Alliance Strategies		Design Rules & Policies	Production Rules & Policies	Supplier Relationship Planning	Brand Management	Supplier Recovery
Control	Human Capital Management	Risk management & Internal Audit	Program Management	Production Scheduling	Supply Chain Performance Monitoring	Relationship Monitoring	Failure Analysis
	Legal & Regulatory	Treasury	Configuration Management	Production Monitoring	Supplier Management	Demand Forecast & Analysis	Quality Management
	Business Performance		Design Validation				
	Intellectual Property	Tax Management	Change Management	Quality Control	Logistics Management	Dealer Management	Parts Management
Execute	Knowledge & Learning	Accounting & General Ledger	Mechanical/ Electrical Design	Plant Operations	Inventory Management	Finance/ Lease Management	Claim Entry & Administration
	Building/ Facilities & Equipment		In-vehicle System Design		Transportation Management	Order/ Configuration Management	Service
	IT Systems & Operations	Cost Management	Process Design Tool Design & Build	Maintenance Management	Procurement	Customer Relationship Management	Body Builders

Figure 6-5 Automotive industry Component Business Model

Table 6-1 provides a summary of one business goal and the supporting KPI to illustrate the goal-service model.

Table 6-1 Business goals and KPIs for the goal-service model

Goal	ROI of Goal	KPI / Metric	Service
1.1 Reduce cost and time to market by reducing the number of physical prototypes to required validate the design	1000000\$	1.1.1 Reduce the time to perform simulation analysis in the product development life cycle	PerformSimulation

## Domain decomposition

With domain decomposition, we work from the top-down by separating the business domain into major functional areas and subsystems. At the next level, we further decompose the functional areas into processes, subprocesses, and high-level business use cases.



Domain decomposition uses and enhances a subset of the Domain Analysis and Domain Engineering approaches, including the following approaches:

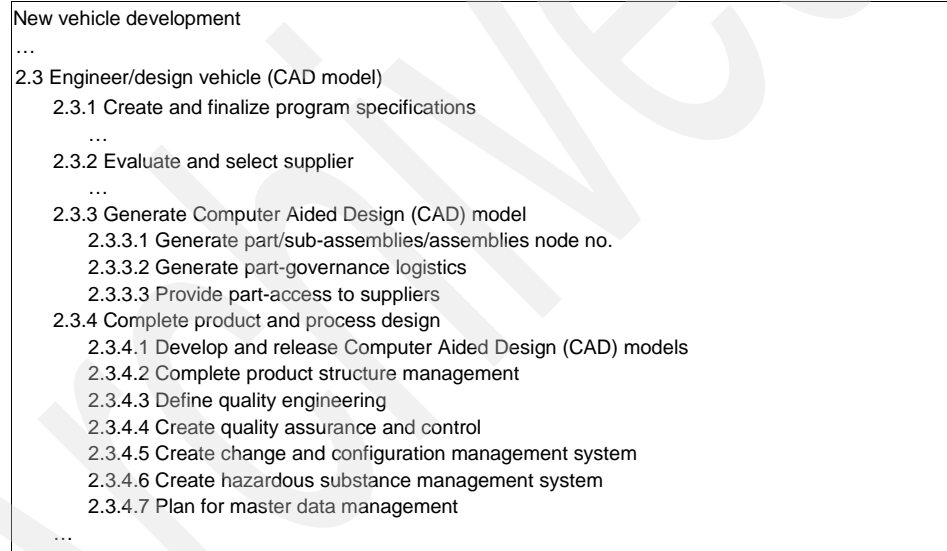
- Functional area analysis

Decompose domains into functional areas to provide business boundaries for the design of IT subsystems and their corresponding service components that realize services. If a CBM is not provided, then Domain Analysis is conducted as an initial stage of the SOMA study.

- Process decomposition

Conduct business process modeling to decompose processes into sub-processes and tasks. Three levels of decomposition is typically adequate for an initial list of candidate services.

Figure 6-6 shows an sample process decomposition re-used from the IBM Process Classification Framework.



*Figure 6-6 Sample Process decomposition extracted from the IBM Process Classification Framework for Auto*

- Variation-oriented analysis

Look across the processes, rules, policies, and structure (data) to identify candidates for commonality. Next, separate out the variations of processing, rules, and structure. The Variation-oriented analysis ensures that identified services are re-usable in various contexts.

## Existing asset analysis

The main objective of the existing asset analysis is to maximize the reuse of existing application transactions, modules from existing systems, and packaged applications. When performing an existing asset analysis, we work bottom-up to identify candidate services. There might be some new services identified, and in other cases, the technique confirms identification from a previous technique.

Trucks Inc.'s Product and Process business unit owns many COTS PLM applications as well as applications developed in-house. COTS PLM applications such as UGS TeamCenter, PTC Windchill, Dassault System Matrix One, and MSC SimManager can expose Web services that can be consumed by other components. The existing asset analysis is not simply about exposing the existing application interfaces as a Web service. Consideration is needed to determine if existing application interface allows for good service design.

This is all the difference between a re-usable business service and an application-level service, which in general is specific to the application's standards (such as a specific application data format) and therefore is not likely to comply to the OMG PLM Services 2.0 standard chosen by Trucks Inc..

As shown in Figure 6-7 on page 133, there are several options for exposing existing applications:

- ▶ **Wrap existing function**  
Leave the function as-is, but use tools or middleware to expose the existing function as a service. For example, expose Trucks Inc. existing applications as SOAP Web services (also known as direct exposure).
- ▶ **Wrap and replace existing function with a service**  
Wrap a function as above, but use the resulting service specification to redevelop the service at a later date. Then, replace the original service and redirect clients to the new implementation
- ▶ **Use an adapter more amenable to service invocation**  
In some cases, it is not possible to wrap a function and expose it as a service. However, it might be possible to wrap the function in something more able to integrate, such as a message queuing interface or the Java Connector Architecture (JCA), which allows new services to access the function in-place (also known as indirect exposure). In general, JCA adapters provide more robust Quality of Services such as security, performance, transaction, fail-over that are directly managed by the middleware. IBM SAP JCA Adapter is an example of an adapter supporting these quality of services.

- Integrate the function into the service

In some cases, it is possible for the new service to access the existing function in-place, simply using the function as a logical component within the implementation of the service. For example, the service implementation may use a message-oriented protocol such as Java Message Service to access directly existing functions exposed by the application.

In any of the possible scenarios, OMG PLM Services interfaces can be exposed to the other service consuming applications.

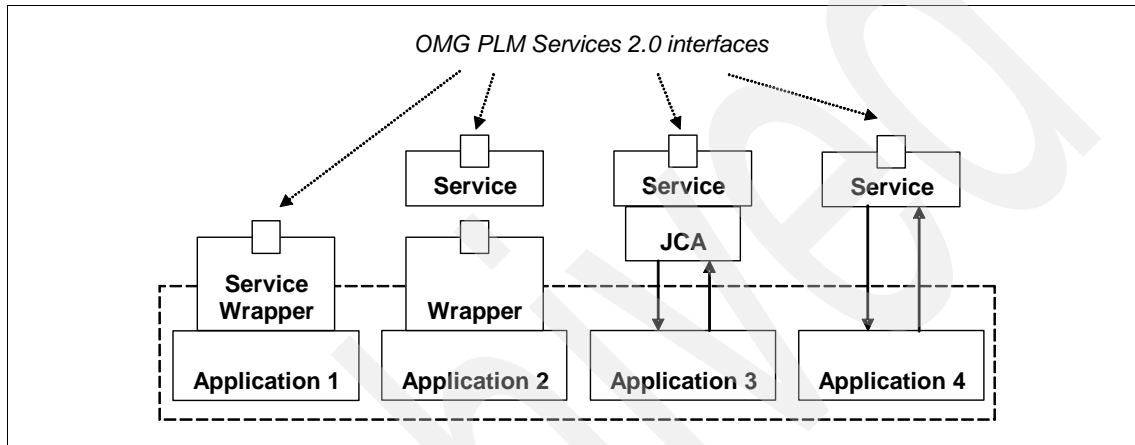


Figure 6-7 Options for exposing existing PLM applications as services

## 6.2.2 Service Specification

The service specification defines the dependencies, composition, exposure decisions, messages, quality of service constraints, and decisions regarding the management of state of a service. The objective of the service specification task is to elaborate the service model.

The service specification includes the following subtasks:

- Apply service litmus test to make exposure decisions.
- Identify service dependencies.
- Identify services composition and flow.
- Identify non-functional requirements.
- Specify service messages.
- Document state management decisions.

## Apply service litmus test to make exposure decisions

At this stage of the SOMA Method, candidate services have been selected and documented in the (categorized) Service Portfolio. We now need to determine which ones must be exposed as services.

Although, in theory, any candidate service could be exposed by exporting its interface as a service description, not every candidate service must be. It may not be economically and practically feasible to do so as non-functional requirements may be compromised. In particular, the naïve decision to expose all methods from all classes results in an overwhelming and often unmanageable number of services leading to the *Service Proliferation Syndrome*. This creates huge performance and service management problems, not to mention the possible loss of Trucks Inc.'s intellectual capital. We must remember that there is a cost associated with every service we choose to expose: the funding, governance, the underlying infrastructure (its security, performance, management) of the service, and the components that implements them must be taken into account.

Therefore, some criteria are needed to help decide whether to expose a service and, most importantly, whether to fund the creation of the service component that provides the functionality of the service as well as the maintenance, monitoring, security, performance and other service level agreements of the service. For example, Trucks Inc. may not want to expose a service triggering a CPU and disk space consuming simulation.

This decision making process is referred to as the *Service Exposure Decisions*.

A set of criteria in the form of the *Service Litmus Test* can be used to filter the collections of candidate services.

## Identify service dependencies

Detailed review of the service can expose service dependencies on other services or applications that are used to realize the functionality of the service.

There are two types of dependencies that need to be considered:

- Functional dependencies

Functional dependencies are those between services that rely on each other to deliver the services that they deliver. For example in Figure 6-8 on page 135, the Manage Configuration composite service has a functional dependency on the Validate parts, build combinations, and Develop BOM services.

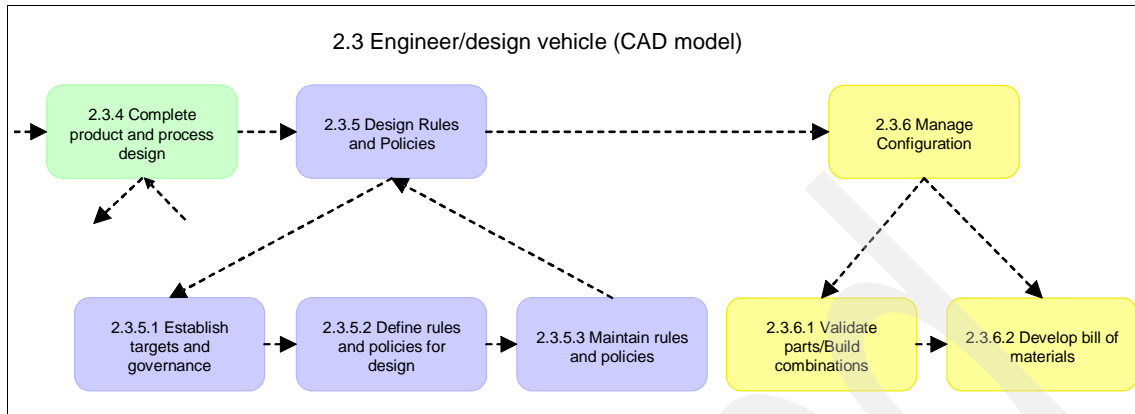


Figure 6-8 Composite services in Engineer/design vehicle process (IBM Auto Process Classification Framework)

► Processing dependencies

Processing dependencies are those between services that are choreographed together to make up a business process. For example, the Design review process is dependent on the services shown in Figure 6-9.

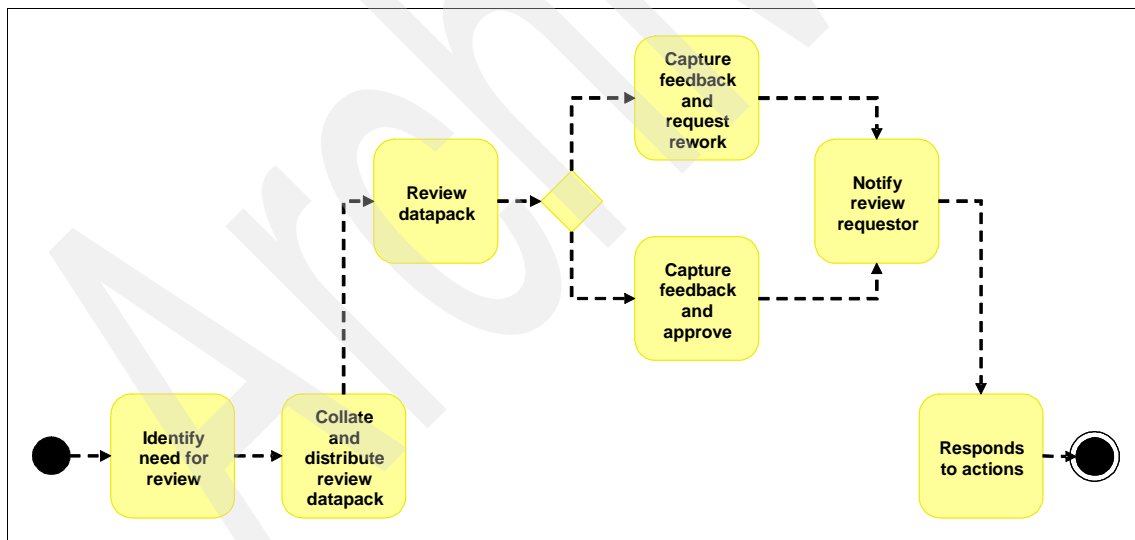


Figure 6-9 Design Review process

## **Identify services composition and flow**

A review of functional areas and business processes helps elaborate the composition of services and their flow. The service flow specification describes the choreography between services. For example, the design review composite service is a long-running, interruptible process macro-flow. The capture feedback and request rework process is a short-running, non-interruptible process (micro-flow). Other examples of micro- and macro-flow are presented in Part 3, “Overview of working examples” on page 175.

## **Identify non-functional requirements**

Service models must consider the non-functional requirements that are used to specify quality of service (QoS). In the PLM domain, non-functional requirements can be important to consider up-front in the identification and the design of services.

For example, a Bill of Material Synchronization service likely introduces a non-trivial transaction handling management with the enterprise ERP that is worth documenting during the Service Identification phase. A BOM Synchronization service implementation with SAP is illustrated in Part 3, “Overview of working examples” on page 175.

## **Specify service messages**

Data flow within the service model is most often represented in terms of messages that flow between the services. During the service specification process, there are cases where the data model is not completed. Consider cases where there is not enough detail about the services to be realized at this point in time. With that said, there still needs to be consideration given for the data and messages for the service inputs and outputs.

In some situations, the polymorphic structure of the OMG PLM Services 2.0 model can allow for an incremental approach when designing service messages. Let's take the example of a service designer who knows that there is a need to perform a query but does not know exactly which exact query type. The designer may specify `Specific_query`, or `Relationship_query` at this stage of the Service Identification. Later on, the designer may go deeper in the message specification and select `Geometric_model_relationship_query` (see Figure 6-10 on page 137).

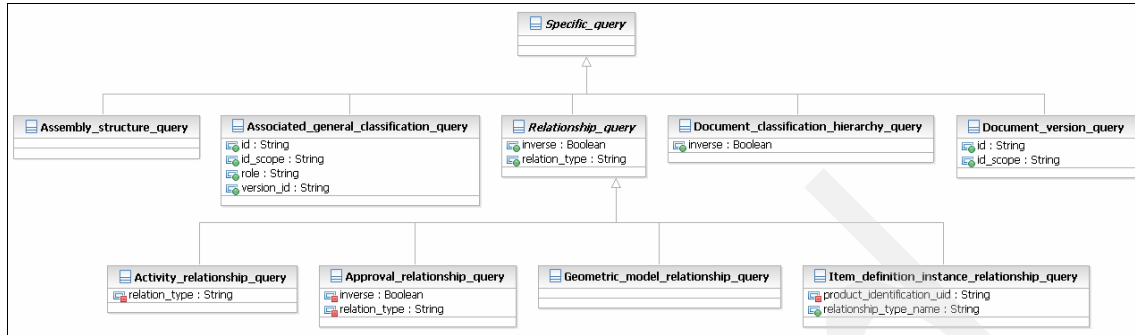


Figure 6-10 Sample query classes from the OMG PLM Services 2.0 computational model

## Document state management decisions

In certain cases, the composition of services requires management of state to be documented, such as stateful, stateless, stateful with cached state, and so forth. For example, there are cases where state management for portions of the process might be controlled by composite services or other elements.

OMG PLM Services 2.0 contains several state-oriented objects such as the activities related to a specific work order, shown in Figure 6-11.

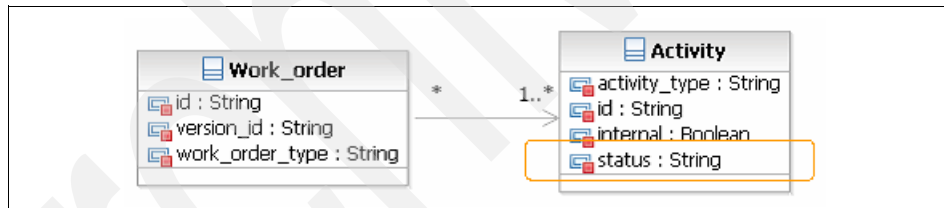


Figure 6-11 An Activity is typically a state-oriented entity

### 6.2.3 Service component specification

The final part of the Service Specification task is component specification. It is often difficult to perform this task in isolation, thus the Realization tasks are often performed in parallel and are iterative. Service component specification includes the following subtasks:

- ▶ Identify component attributes
- ▶ Identify events and messages
- ▶ Identify component internal flow
- ▶ Create component class diagrams
- ▶ Variation-oriented design

All the service component specifications are packaged in one or several Service component models, making these models a major design artifact that contain most of the service specifications captured in the previous stages of the SOMA method. In general, service component specifications are described as UML Components. In the following samples, we use IBM Rational® Software Architect to define UML Component diagrams.

**About Rational Software Architect:** IBM Rational Software Architect is an integrated design and development tool that uses model-driven development with the UML for creating well-architected applications and services. With Rational Software Architect you can unify all aspects of software design and development:

- ▶ Develop applications more productively than ever
- ▶ Exploit the latest in modeling language technology
- ▶ Review and control the structure of your Java applications
- ▶ Use an open and extensible modeling platform
- ▶ Simplify your design and development tool solution
- ▶ Integrate with other facets of the life cycle

Figure 6-12 on page 139 shows how Trucks Inc. organizes its various UML Service Component models. One enterprise-wide service component model contains the other models that are maintained at business unit level (note that only three business unit are represented here).

The UML models can refer to other UML models (such as industry standard model) as OMG PLM Services 2.0 and OAGIS 9.1 UML models. By doing so, the SOA architect in charge of specifying the service components is able to re-use most of the elements of those standards in its UML tool. In the following examples, we show how to re-use the OMG PLM Services 2.0 UML model.



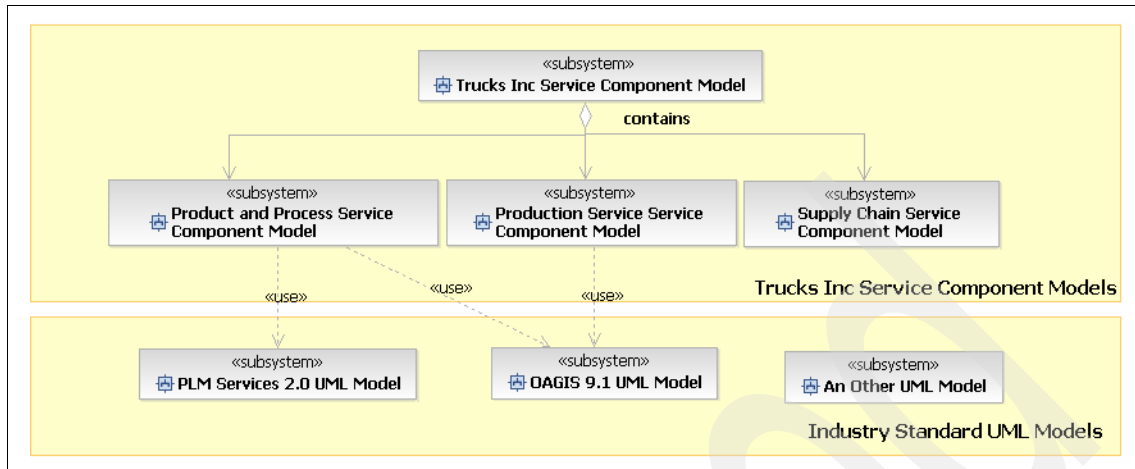


Figure 6-12 UML Service Component Models organization at Trucks Inc.

Let us now describe how Trucks Inc. may model its Enterprise PDM application as a service component in UML, using IBM Rational Software Architect, the UML 2.0 Profile for Software Service and OMG PLM Service 2.0 UML model.

**About UML profiles:** UML profiles add a simple extension mechanism to the domain-independent Unified Modeling Language. They allow definition of domain-specific entities and rules. The UML 2.0 profile for software services is dedicated to Service component specification. It allows an architect to define a service component models and then generate WSDL service definitions using a model driven development technic. See the following Web page for more information on UML profiles:

[http://www.ibm.com/developerworks/rational/library/05/419\\_soa/](http://www.ibm.com/developerworks/rational/library/05/419_soa/)

Our basic architectural assumption is that the Enterprise PDM application is integrated using a JCA Adapter (Figure 6-13 on page 140) following the indirect exposure pattern explained in “Existing asset analysis” on page 132.

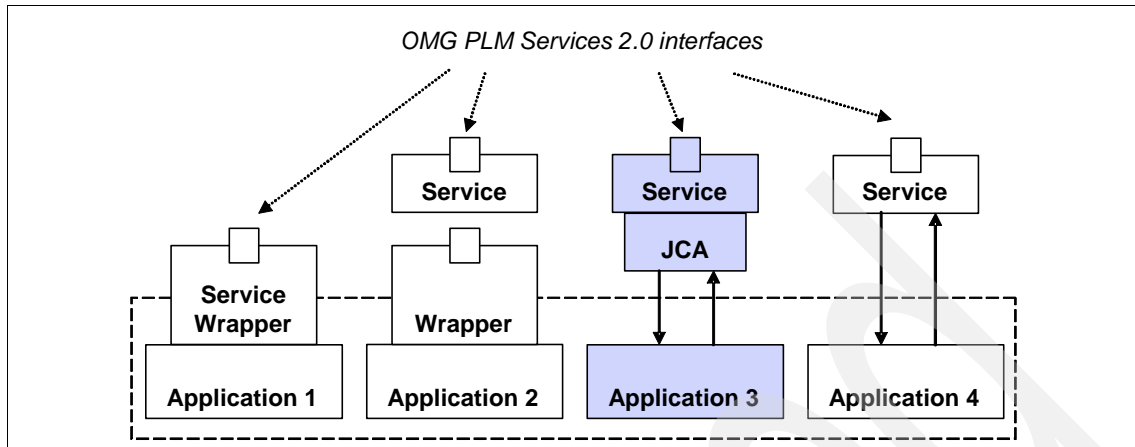


Figure 6-13 Trucks Inc. Enterprise PDM is integrated as a service component using the indirect exposure pattern, with a JCA adapter

Figure 6-14 on page 142 shows four major elements to be considered in this integration scenario:

- Enterprise PDM component

This UML component represents the Enterprise PDM application. This might be an existing application or a COTS PLM application such as PTC Windchill, Siemens UGS TeamCenter for example.

The Enterprise PDM UML component is associated with the Enterprise PDM Specific API UML interface with an interface realization relationship.

- Enterprise PDM specific API

This UML Interface represents the Enterprise PDM-specific API. This API may be exposed as Web services, or may use protocols such as RMI/IIOP. In order to take full advantage of JCA adapters features such as security handling, transaction handling, connection pooling and clustering, use a robust transport protocol such as RMI/IIOP. In general, COTS PDM applications expose several kind of API's depending on the usage:

- API's exposed as Web services

SOA clients (such as portal applications) typically use the API's exposed as Web services.

- API's using a robust transport protocol

Robust SOA components that can be re-used in many business scenarios with high level QoS requirements use the API's using a robust transport protocol. For example, SAP exposes an API called Java Connector API (JCO).

► Enterprise PDM Adapter

This UML Component represents the JCA adapter that integrates the Enterprise PDM. It is stereotyped as a service provider (this stereotype is coming from the UML 2.0 Profile for Software Services). An SOA architect experienced in service component UML modeling immediately recognizes that this component is a service provider and that one or more of its interfaces are re-usable by other service components.

Commercial JCA adapters such as IBM SAP JCA Adapter come with a feature called Enterprise Metadata Discovery (EMD) that allows JCA Adapter (at development-time) to introspect the various operations the PDM application can expose. The designer typically selects a subset of these operations and lets the EMD tooling generate service-based interfaces corresponding to these operations. This scenario is described in detail in Part 3, “Overview of working examples” on page 175.

Generally, The service interfaces exposed by the adapter are not using HTTP as transport protocol binding. Instead, robust transport protocols supported by the middleware platform are used. For example, IBM WebSphere is using the Service Component Architecture that supports various service binding transport protocols such as RMI/IIOP, JMS, and HTTP.

The Enterprise PDM Adapter UML component is associated with the Enterprise PDM Specific Service UML interface with an interface realization relationship. It is associated with the Enterprise PDM UML component with a usage relationship.

► Enterprise PDM Specific Service Interface

This UML Interface contains the subset of the operations selected for our service component during EMD. This interface is stereotyped as a service specification (this stereotype is also coming from the UML 2.0 Profile for Software Services). Service components using this adapter typically use this interface.

At this stage, however, our adapter is not exposing more than the selected application's specific operations as services. Using this adapter as an enterprise component would present a number of drawbacks. Any component consuming the services exposed by this adapter would have to map its own data representation to the Enterprise PDM's specific data representation.

The next step is to map this adapter's interface into an interface compatible with OMG PLM Services 2.0, Trucks Inc.'s PLM neutral language.

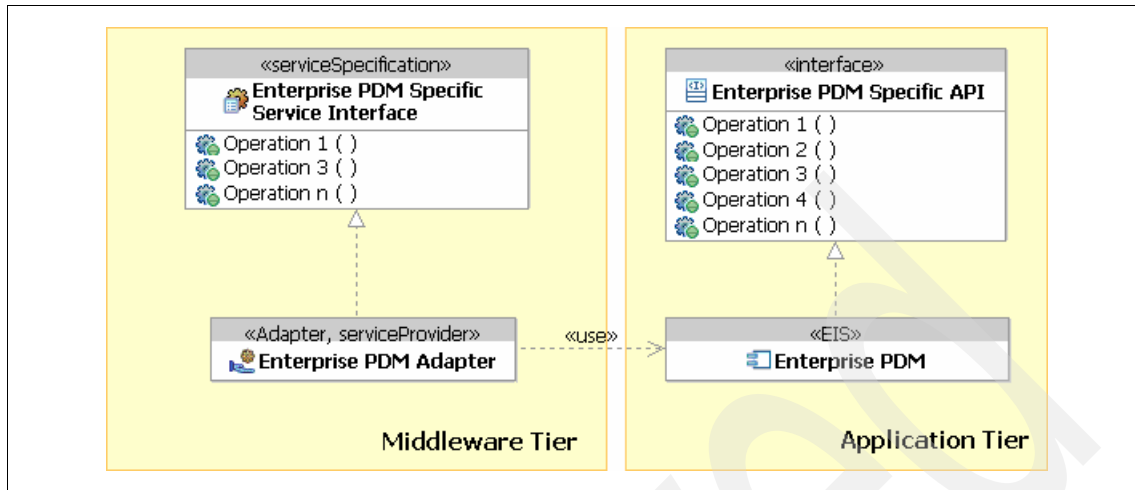


Figure 6-14 Trucks Inc. Enterprise PDM and its JCA adapter re-using a subset of the operations available in the Enterprise PDM Specific API

Let us take the simple case where there is only a requirement to integrate synchronously with the Enterprise PDM. In this situation, the OMG PLM Services 2.0 PLM\_general\_connection interface is the service interface that our service component must expose.

At this stage, two major components need to be introduced: the maps in charge of converting OMG PLM Services 2.0 data into the Enterprise PDM data and vice-versa. Due to the complex nature of PLM data structures, these maps are likely the most time consuming components in terms of development. Another alternative for Trucks Inc. is to buy commercial adapters specialized for the OMG PLM Services standard. For example, OpenPDM from ProSTEP contains a large number of adapters compliant with most of the COTS PLM Applications. In the situation where Trucks Inc. needs to specify a service component wrapping an existing application, however, the development of those maps is required.

These two maps are described in Figure 6-15 on page 143. We explicitly show which interfaces are required and which are provided by setting the appropriate UML appearance parameters in Rational Software Architect. The PLM\_general\_connection interface is re-used from the OMG PLM Services 2.0 UML model.

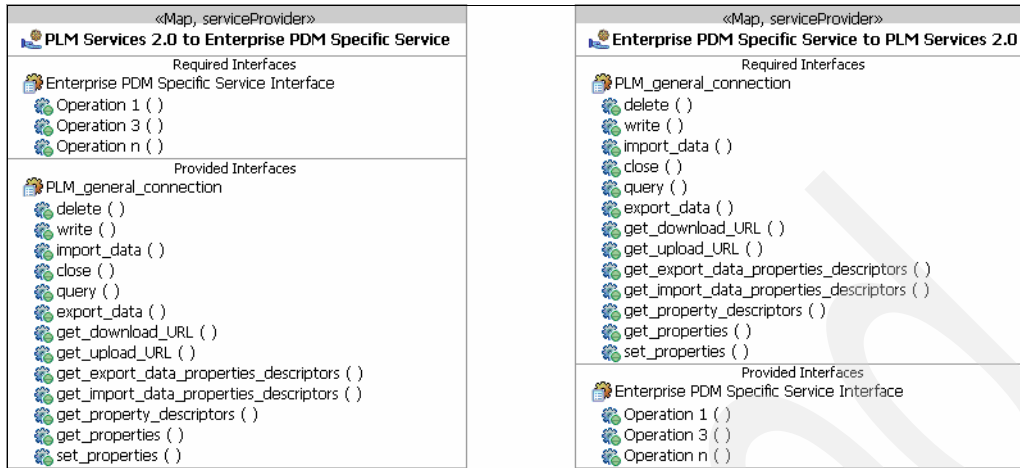


Figure 6-15 Modeling the maps converting OMG PLM Services 2.0 data to the Enterprise PDM data and vice versa

Figure 6-16 on page 144 shows the Enterprise PDM Services service component with its two maps, outbound and inbound.

**Outbound and inbound defined:** Outbound is the direction from the middleware tier to the application tier. Inbound is the direction from the application tier to the middleware tier

We displayed the UML Component structure compartment in order to show the two maps that are inner components belonging to the Enterprise PDM Services component. Note the little handles describing which interfaces are required and provided in a compact way.

The UML component diagram in Figure 6-17 on page 145 shows an overview from the Enterprise PDM Services component down to the Enterprise PDM application through the adapter. The Trucks Inc.'s SOA Architect would likely use this diagram to explain his end-to-end design in one diagram.

From now, from an SOA integration perspective, the Trucks Inc.'s Enterprise PDM application is only represented by the Enterprise PDM Services service component. The service consuming components are not aware of the underlying interaction method and of the application-specific API. This design has many advantages. One main advantage is that Trucks Inc. can replace the Enterprise PDM application with a new one with a minimal impact on its SOA-based components.

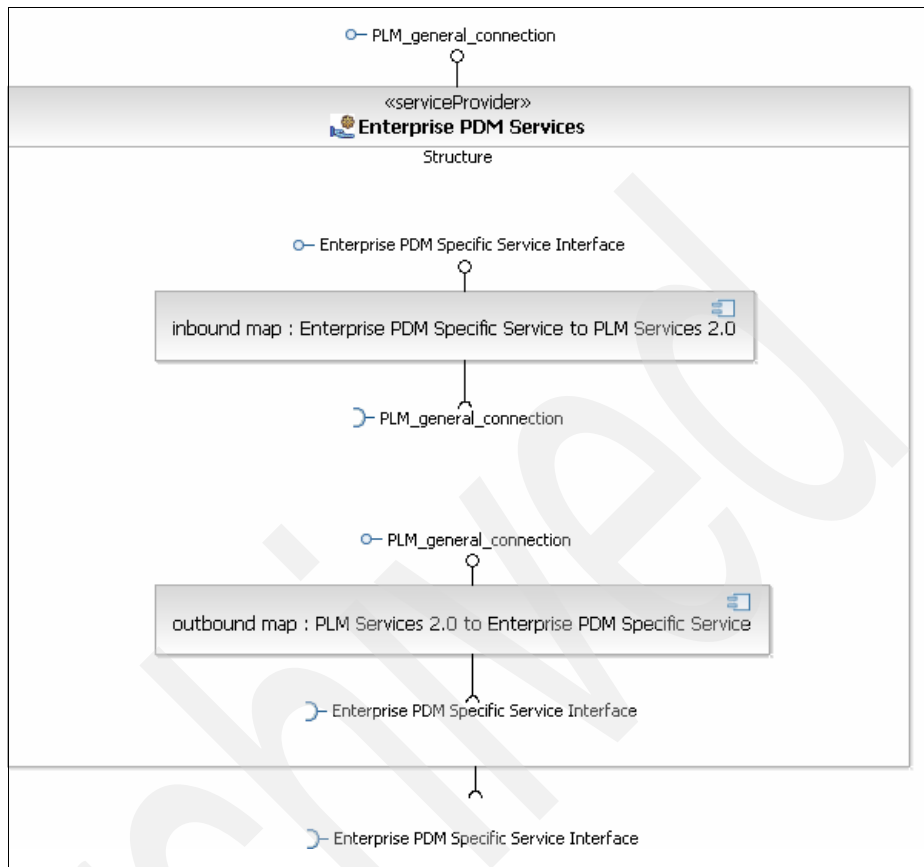


Figure 6-16 The Enterprise PDM Services component with its two outbound and inbound maps

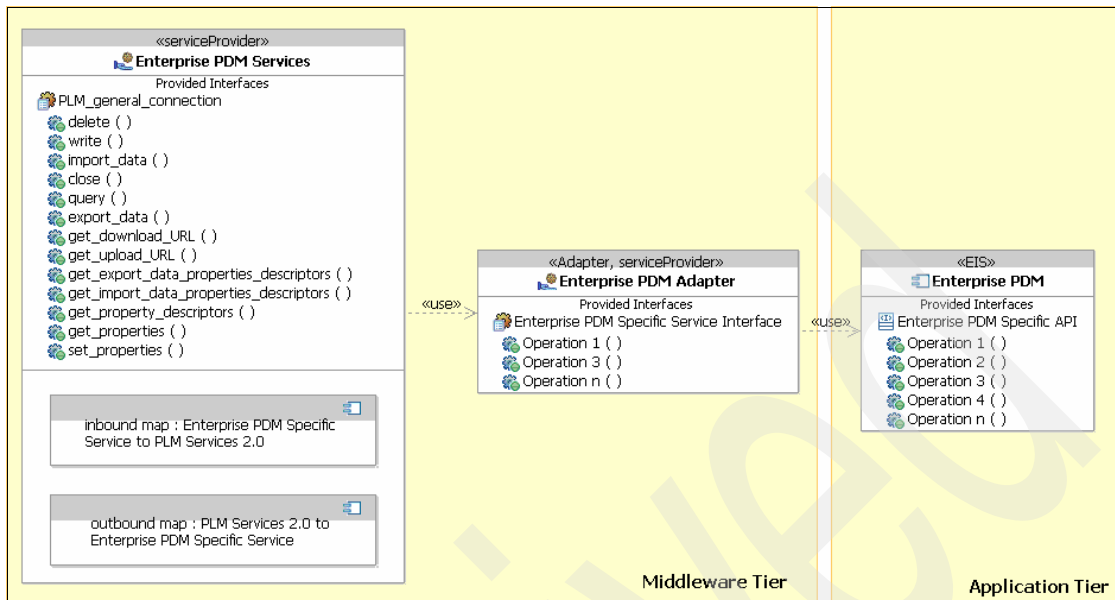


Figure 6-17 Overview from the Enterprise PDM Services component down to the Enterprise PDM application

## 6.3 SOA Governance

Trucks Inc. started in the service identification process with enthusiasm and efficiency, and the first SOA components are implemented with success. SOA now becomes a corporate strategy and several projects are launched in various Trucks Inc. business units.

After several months, however, Trucks Inc. is now plagued with certain problems surrounding their SOA implementation. They are not achieving their business agility goals:

- ▶ They are struggling to identify new service candidates and prioritize them.
- ▶ They are having significant issues with service reuse (or lack thereof).
- ▶ Adoption of corporate standards is haphazard at best.
- ▶ With the first services in place, governance of changes and versions for services is crude and unrefined.
- ▶ There is no way to enforce operational policies consistently across the SOA runtime environment.

### 6.3.1 SOA Governance project requirements

SOA governance helps with the proper definition of roles and responsibilities within an organizational structure. Trucks Inc. plans to use the SOA Governance scenario to help them govern the creation of new services, reuse existing services, and comply to standards and best practices.

This section describes the organization structure of Trucks Inc., and lists their requirements for SOA governance.

Figure 6-18 on page 147 shows the IT organizational structure of Trucks Inc. The groups and roles shown in Figure 6-18 on page 147 are defined in Table 6-2 on page 147 and Table 6-3 on page 148.

The Trucks Inc. organization is headed by an IT Executive Steering Committee (ITESC) led by the CIO. The domain owners within the ITESC own and are accountable for the business functionality within their proper business domain. These domain owners report to the CIO, but they also have direct reporting responsibilities within their business domain. The ITESC also lists key technical roles. These technical roles, along with the domain owners, strive to achieve a confluence between business and IT.

The next layer of the organizational model shows the business relationship directors and the SOA Board. The business relationship directors provide insight to the domain owners on priorities for that particular business domain. They also work with the SOA Board to give insight on what business functionality is needed. The SOA Board is in charge of defining and updating the strategy and direction for the SOA initiatives, the high level direction of the architecture, and properly directing the service portfolio with the guidance of the business relationship directors.

The SOA Core Team takes direction from the SOA Board to help implement the required services. The role of the SOA core team is as follows:

- ▶ Helps define the various processes across the service life cycle
- ▶ Helps define the standards and best practices around those process
- ▶ Ensures those processes are properly communicated, enforced, and evolved

Like the other layers of the CoE organization, the core team also has representation from both the business and IT perspectives.



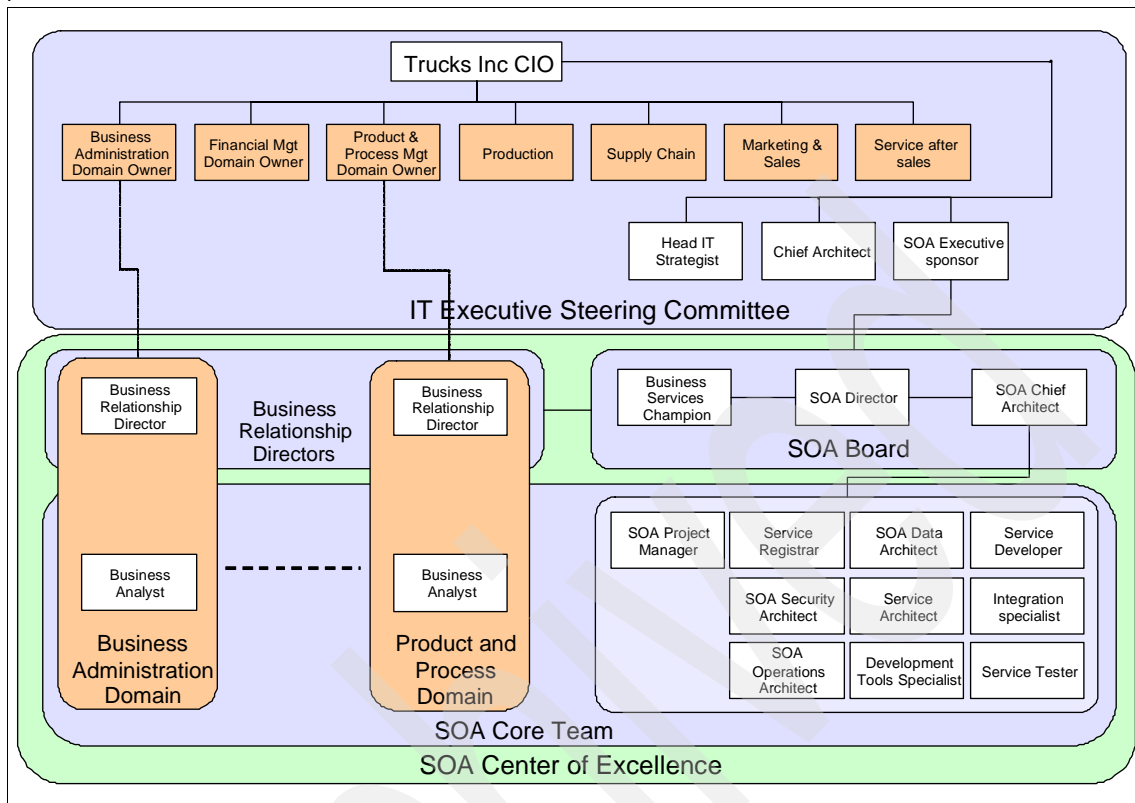


Figure 6-18 Trucks Inc. IT organizational structure

Table 6-2 Groups within the Trucks Inc. organization

Group	Description
SOA Board	Oversees SOA Governance process, provides recommendations to IT Executive Steering Committee.
IT Executive Steering Committee	Ultimate decision makers for decisions regarding service and IT related matters.
SOA Core Team	Group of business and SOA IT enablement expertise that oversees the implementation of SOA related activities.
SOA Center of Excellence (CoE)	The larger SOA Core team, SOA CoE and business domain owners that decide on the business objectives, imperatives and decisions that drive SOA enablement.

Table 6-3 SOA Governance-related roles within the Trucks Inc. organization

Group	Description
SOA Chief Architect	Oversees all SOA architectural decisions.
SOA Security Architect	Oversees all security considerations for SOA.
SOA Project Manager (PM)	Manages all SOA related projects.
SOA Operations Architect	Orchestrates SOA related operations activities.
SOA Data Architect	In charge of data related management for SOA.
Service Tester	Tests services for SOA compliance.
SOA Integration Developer	Integrates services into SOA ecosystem.
SOA Director	Leads SOA Board and manages all SOA initiatives.
Business Relationship Director	Per Line of Business (LOB), manages SOA related business matters.
Business Service Champion	Champions business related SOA requirements. Drives business priorities into the service portfolio with the help of the Business Relationship Directors.
Business Analyst	Performs business analysis for SOA services. Involved in the Service Identification and specification phase of the SOMA method.
Service Registrar	Manages the service metadata and ensures that the advancement of services through the life cycle is consistent with the standards defined.
IT Architect	Aligns SOA and Enterprise Architecture needs.
Service Architect	Architects individual services.
Service Developer	Develops individual services.
IT Portfolio Manager	Manages the portfolio of IT relates services.
Integration Specialist	Implements and integrated services into IT infrastructure.
Development Tools Specialist	Ensures integration with business back-end as well as the proper functioning of the development environment.
Configuration Librarian	Generates CIs in change management database.
Release Specialist	Creates and certifies release packages.
Security and Operations Specialist	Determine significance of events from operations.

Group	Description
Compliance Auditor	Determine if events represent a compliance problem (intrusion, fraud or availability impacting financial reporting).
Availability Analyst	Determine outcome for monitoring events (real service outage, or something handled in IT capacity plan).
Change Manager	Decides and records decisions about how to handle an incident, change request or availability (service outage) report.

### 6.3.2 SOA Governance requirements at Trucks Inc.

The Trucks Inc. ITESC have defined the following requirements to address the challenges of SOA governance:

- Govern how new services are created

Trucks Inc. continues to introduce new services, particularly with the development of Product and Services business unit. Challenges exist on how to extend the current IT governance model to embrace these new services, and to understand the roles involved.

- Reuse existing services at every opportunity

The Trucks Inc. ITESC is concerned that services with similar functionality exist across the enterprise. In many cases a pre-existing service could have been reused but differences in terminology, disputes about entitlement, and problems in enhancing services have prevented this.

- Enforce standards and best practices across Trucks Inc.

The Trucks Inc. ITESC would like to see a greater use of standards across the Trucks Inc. organization to reduce errors that arise throughout the service life cycle. The Trucks Inc. ITESC would also like to see best practices compliance to ensure consistent development, deployment, and operations of services.

### 6.3.3 Applying the SOA realization patterns at Trucks Inc.

Trucks Inc. intends to use the realization patterns of the SOA Governance scenario to solve their governance problems. Implementing these realization patterns secures value from the following realization patterns:

- ▶ Governing new service creation and determine priorities and funding for the establishment of services.
- ▶ Identify who owns services and drive better requirements against services to harbor reuse by establishing service level agreements.
- ▶ Establish consistent use of business terms across services and data sources.
- ▶ Ensure services are adhering to standards and best practices, thereby reducing risks across deployment and management.
- ▶ Ensure a consistent service change management capability and a more refined, explicit versioning process for services.
- ▶ Enforce a starter set of WS-Policy-based domains across the runtime environment.

By not implementing the realizations in this scenario, Trucks Inc. takes the following risks:

- ▶ Increased project risks due to inconsistent approaches.
- ▶ Increased cost due to non-interoperable services.
- ▶ Lengthened time to market (slower speed to market) due to longer project time frame.
- ▶ Increased business risks due to redundant services.
- ▶ Lack of data integrity across services and persistent business data.
- ▶ Audit exposures.
- ▶ Spiral increase in maintenance cost (due to multiple copies of similar or same services).
- ▶ Development and deployment of redundant and inefficient services.
- ▶ Redundancy and possible inconsistency of systems functions to support same business processes or requirements.

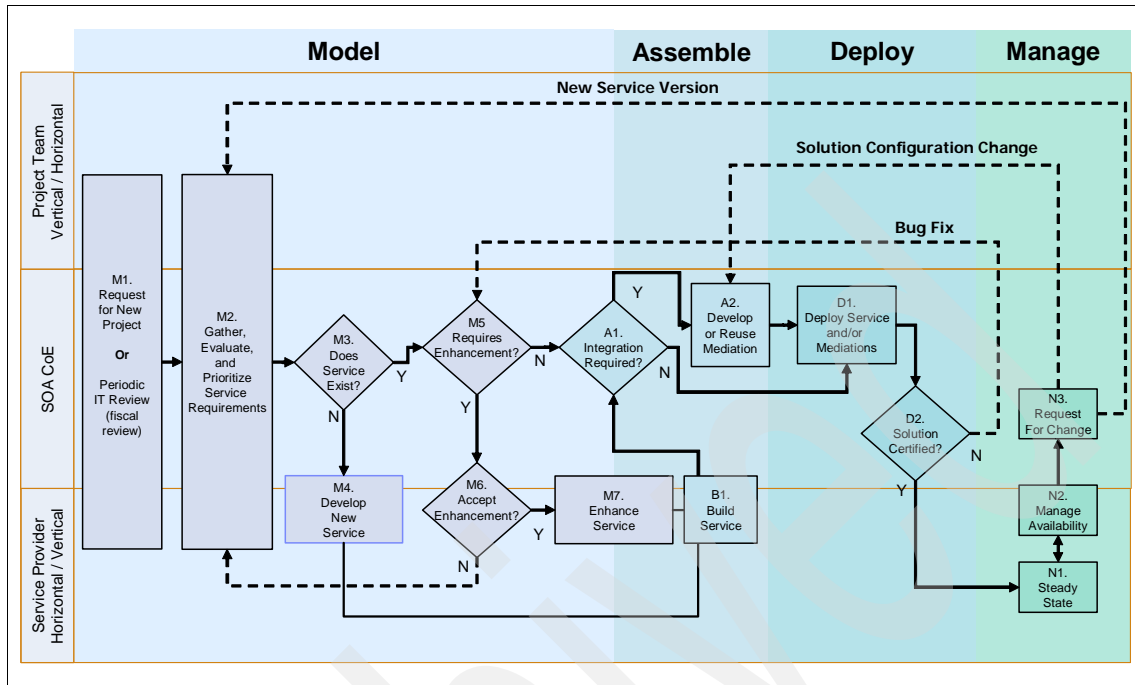


Figure 6-19 Service life-cycle process

The following sections discuss how Trucks Inc. uses the service life cycle process (Figure 6-19) with the following realization patterns:

- ▶ Regulation of New Service Creation
- ▶ Getting More Reuse of Services
- ▶ Enforcing Standards and Best Practices

### 6.3.4 Regulation of New Service Creation

The introduction of new services into the Trucks Inc. SOA environment presents a number of governance challenges:

- ▶ Extending the existing IT governance process that includes the current IT planning process that reviews project candidates and decides which projects to pursue.
- ▶ Understanding the role ownership in various service domains plays when choosing appropriate services and who is accountable for those services.
- ▶ Placing the necessary rigor around the service creation process ensuring the method is followed and harvesting consistent results.

- ▶ Understanding the importance of the funding process, which outlines common scoring techniques used to evaluate various service candidates, and properly prioritize these candidates in a manner consistent with the goals of the business.
- ▶ Understanding the importance of tooling to help cement the concept of business domains, ensuring these domains become more than columns on a slide.
- ▶ Defining a comprehensive approach that takes quality of service criteria defined in the specification phase and ensure these criteria get enforced appropriately.

### Proposed solution

Trucks Inc. use the New Service Creation sub-process shown in Figure 6-20. This is a lower-level process of the service life cycle process. The codes in each activity in Figure 6-20 (M1.A, M2.0, and so forth) relate to the codes shown in the service life cycle process shown in Figure 6-19 on page 151.

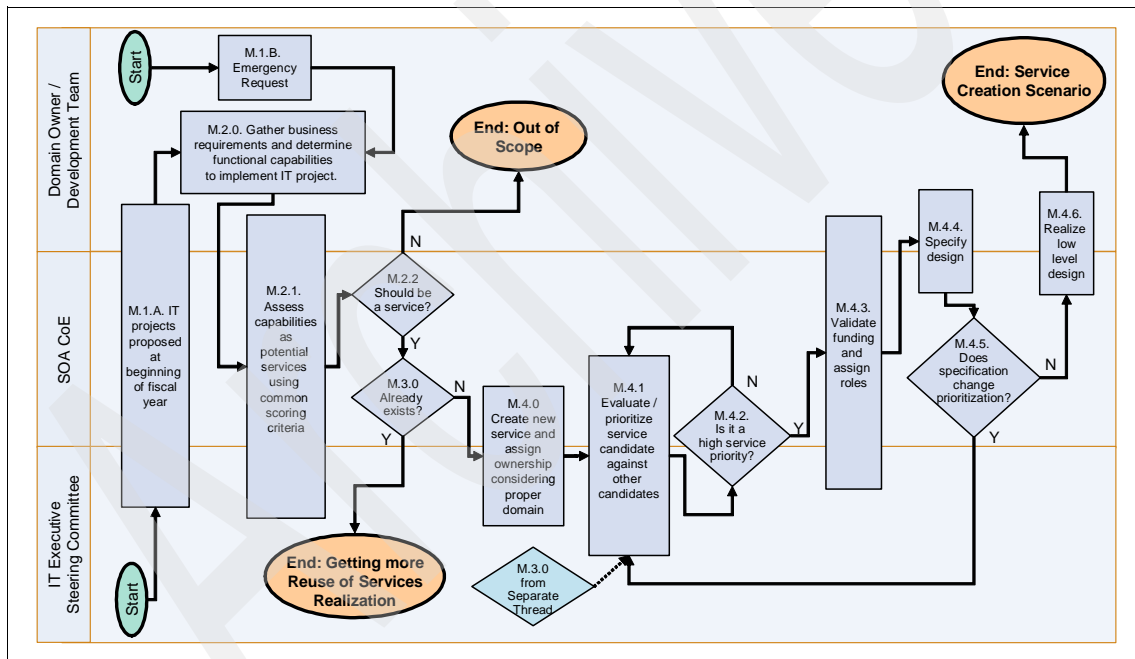


Figure 6-20 Regulation of New Service Creation sub-process

Trucks Inc. uses this process to determine if the new service creation requirements define an SOA service, and if they do, whether an existing service must be reused (in which case the realization pattern discussed in Section 6.3.5, “Getting more Reuse of Services” on page 153 must be applied) or a new service

must be created, ultimately leading to the implementation of a new service (see the Service Creation SOA Scenario focus area described in 6.1, “Achieving business flexibility with a service-oriented architecture” on page 122).

### 6.3.5 Getting more Reuse of Services

Trucks Inc. is eager to implement a structure for the reuse of existing services. Trucks Inc.’s business services champion explains that service reuse drives down development and operations costs of services. Some of the key characteristics of service reuse are as follows:

- ▶ Service reuse requires a centrally governed and managed process for reuse adoption.
- ▶ Service reuse is a cultural challenge as much as it is a technical one. It requires motivation and incentive to drive success.
- ▶ Service reuse requires a holistic approach to information and metadata management. For example, the use of a business glossary ensures consistent use of terminology and promotes clarity for service usage and consumption.

The business services champion explains that there are several aspects of service development, deployment, and runtime management to be governed to get more reuse of services. He identifies three aspects that need to be governed:

- ▶ Governing enterprise-wide use of consistent business terms  
It is hard to identify if an existing service meets the requirements for reuse without governing enterprise-wide use of consistent business terms. For example the business terms *Life-cycle-stage* and *Product family* have slightly different meanings in different departments of Trucks Inc.
- ▶ Governing service enhancement  
A service when first developed may not take into account a varied set of requirements across multiple use cases (spanning multiple business processes). Therefore, without further enhancement (and potentially re-implementing key aspects of a service in some cases) it does not satisfy requirements for reuse.
- ▶ Governing service entitlement  
Service entitlement addresses who can use a service, how often they can use it, and what qualities of service can be expected by consumers of the service. With each new consumer of a service there is the potential for impacting other users. For example, another user may generate a high workload affecting not just that business process, but all other business processes sharing this service.

**Proposed solution**

Trucks Inc. uses the Getting More Reuse out of Services realization pattern to address the governance aspects of consistent business terms, service enhancement, and service entitlement.

**Governing enterprise-wide use of consistent business terms**

Trucks Inc. uses the sub-process shown in Figure 6-21 to illustrate the steps in governing enterprise-wide use of consistent business terms. This is a lower level process of the service life cycle process. The codes in each activity in Figure 6-21 (M2.0.A, M2.0B, and so forth) relate to the codes shown in the service life cycle process shown in Figure 6-19 on page 151.

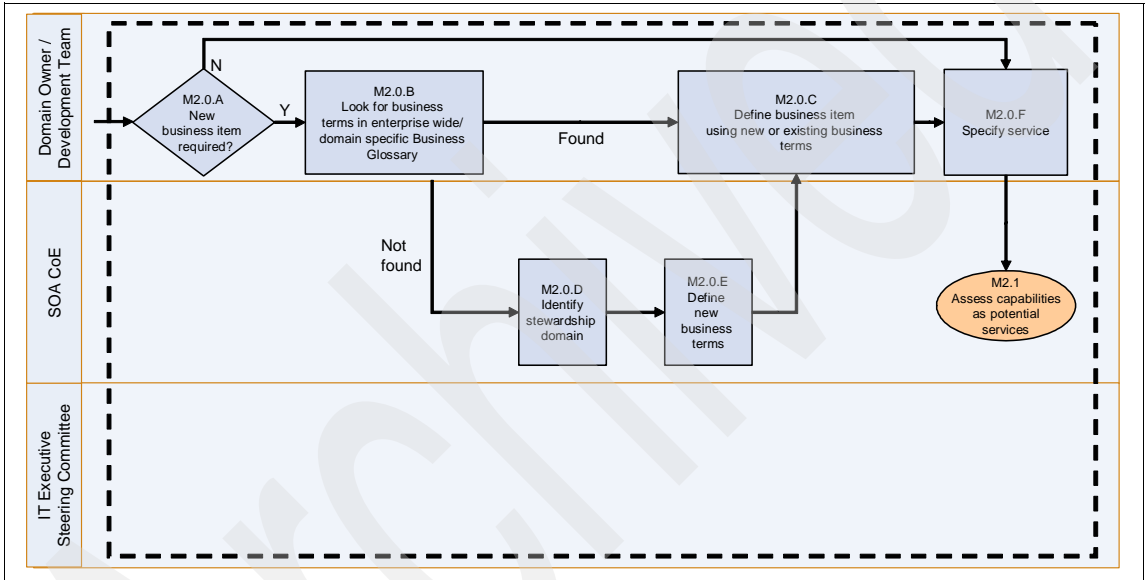


Figure 6-21 Governing enterprise-wide use of consistent business terms sub-process



## Governing service enhancement

Trucks Inc. uses the sub-process shown in Figure 6-22 to illustrate the steps in governing service enhancement. This is a lower level process of the service life cycle process. The codes in each activity in Figure 6-22 (A1, A1.A and so forth) relate to the codes shown in the service life cycle process shown in Figure 6-19 on page 151.

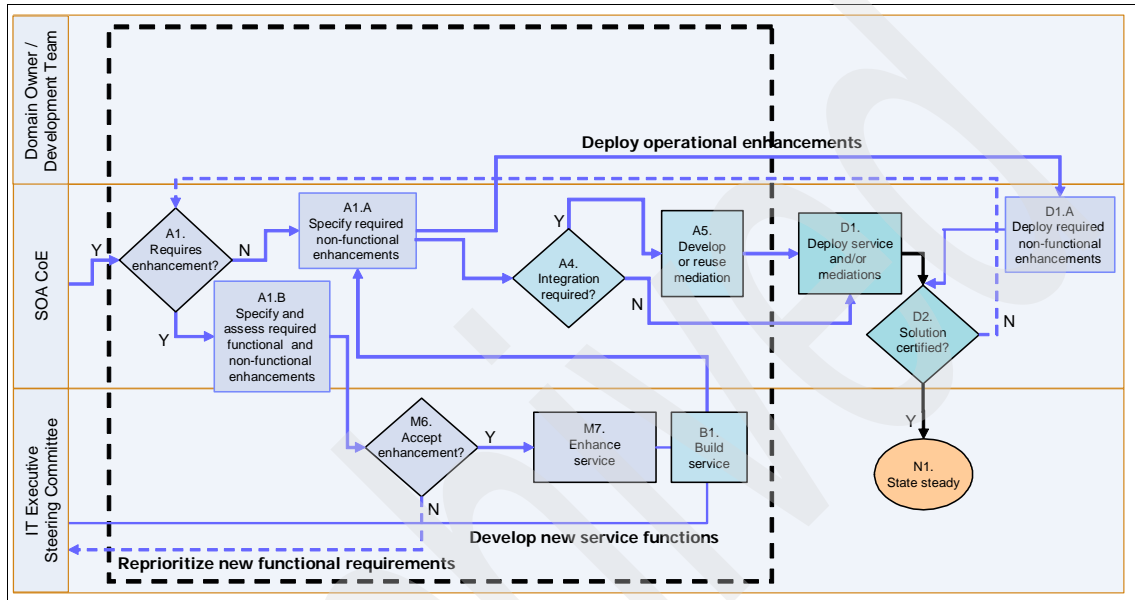


Figure 6-22 Governing service enhancement sub-process

## Governing service entitlement

Trucks Inc. uses the sub-process shown in Figure 6-23 to illustrate the steps in governing service enhancement. This is a lower level process of the service life cycle process. The codes in each activity in Figure 6-23 (A1, A1.A and so forth) relate to the codes shown in the service life cycle process shown in Figure 6-19 on page 151.

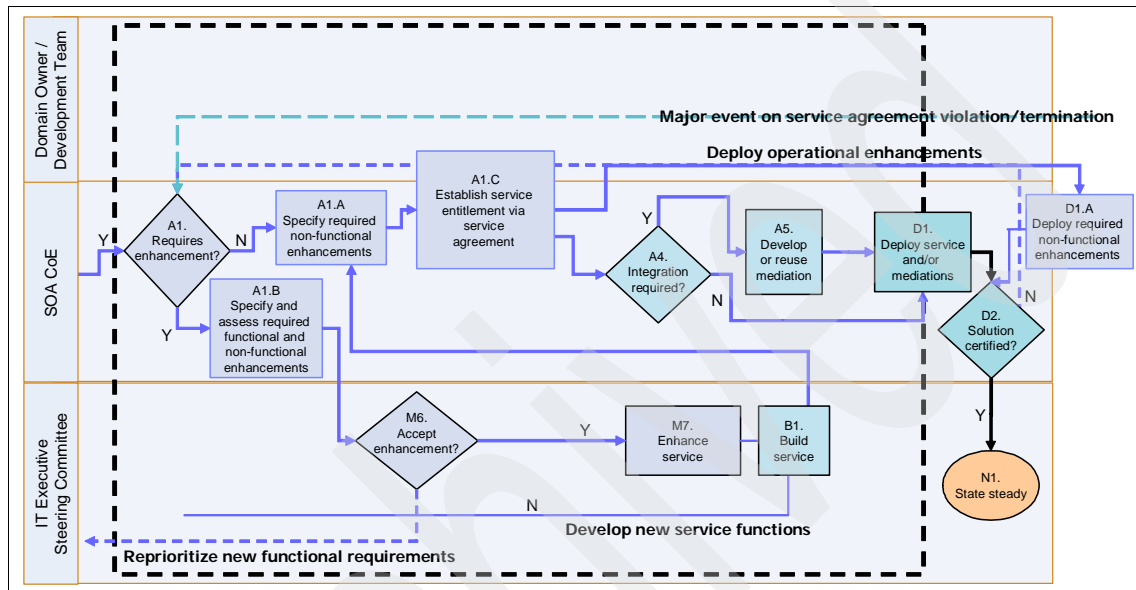


Figure 6-23 Governing service entitlement sub-process

## 6.3.6 Enforcing Standards and Best Practices

The lead business services champion and the SOA director are faced with the following business challenges within Trucks Inc.:

- ▶ How to build, preserve, and apply intellectual capital and working knowledge
- ▶ How to institutionalize governance best practices and standards compliance enterprise-wide
- ▶ How to maintain consistency for IT projects, infrastructure, and so forth
- ▶ How to ensure that standards are in line with business objectives
- ▶ How to comply with industry and government regulations such as RoHS or WEEE<sup>1</sup>

<sup>1</sup> RoHS = Restriction on the use of certain Hazardous Substances; WEEE = Waste from Electrical and Electronic Equipment

- ▶ How to use and coexist with IT best practices such as Information Technology Infrastructure Library (ITIL®)
- ▶ How to ensure that services use enterprise definitions for data models, standard protocols, and so forth (for example, how to ensure that everybody is using the PLM Services 2.0 Web Service standard?)
- ▶ How to satisfy needs for auditing
- ▶ How to achieve compliance in a heavily outsourced environment

To solve these issues, the lead business services champion and the SOA director would like to see Trucks Inc. adopt better processes for best practice compliance and the enforcement of standards.

- ▶ Standards enforcement:
  - Reduces errors throughout the life cycle of a service
  - Provides pre-certification of services for various compliance criteria
  - Drives towards a template driven approach for standards compliance
- ▶ Best practices compliance:
  - Ensures consistent development, deployment and operations of services
  - Organizations with well-defined life cycle phases understand each others' artifacts and deliverables
  - Higher level of expectation managed as organizations process artifacts consistently across the life cycle of services

### **Proposed solution**

Lead Business Services Champion describes a governance framework to update the service life cycle to add compliance points and hooks into existing IT life cycle processes (ITIL) and OMG PLM Services 2.0 compliance. This is shown in Figure 6-24 on page 158.

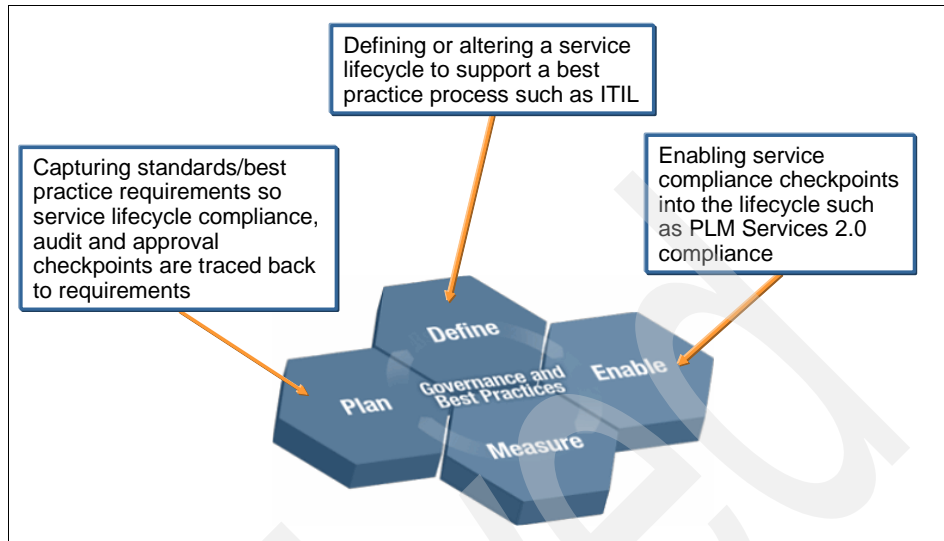


Figure 6-24 Converting governance requirements to proactive best practice processes and compliance and audit checkpoints

The impact on the end-to-end life cycle can be seen in Figure 6-25:

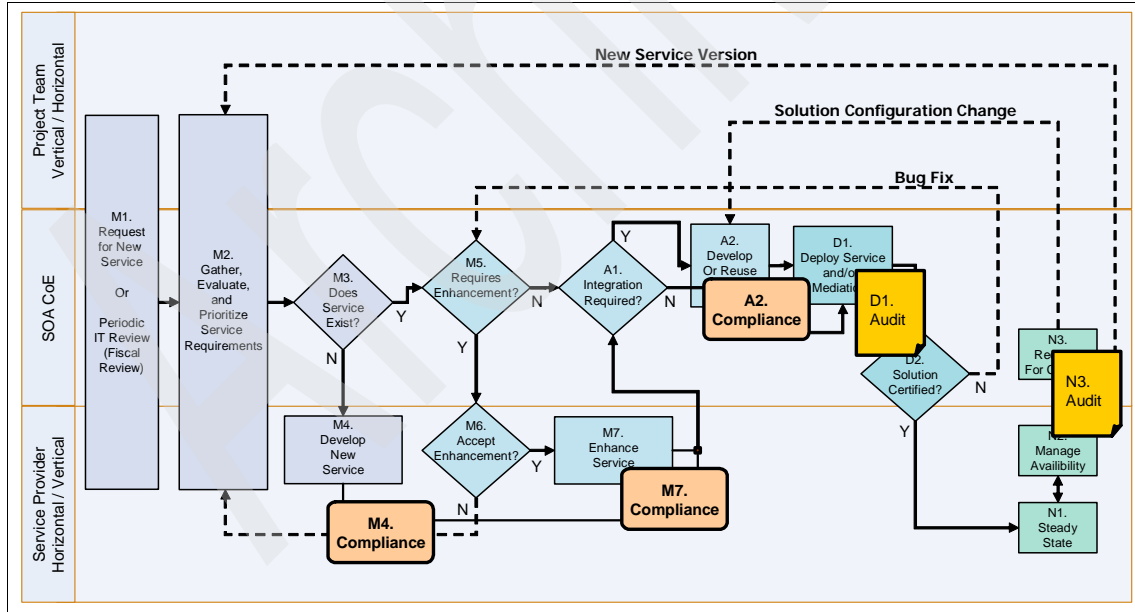


Figure 6-25 High-level view of standards enforcement points

## 6.3.7 Using IBM WebSphere Service Registry and Repository

WebSphere Service Registry and Repository (WSRR) is the tool that Trucks Inc. selected to manage its SOA governance.

### Overview of the WSRR role in SOA

WebSphere Service Registry and Repository (WSRR) is the master metadata repository for service descriptions. This uses a broad definition of service, including:

- ▶ Traditional Web services implementing WSDL interfaces with SOAP or HTTP bindings.
- ▶ A broad range of SOA services that can be described using WSDL, XSD, and WS-Policy decorations, but might use a range of protocols and be implemented according to a variety of programming models.

As the integration point for service metadata, WSRR establishes a central point for finding and managing service metadata acquired from a number of sources, including service application deployments and other service metadata and endpoint registries and repositories, such as UDDI. WSRR is where service metadata scattered across Trucks Inc. is brought together to provide a single, comprehensive description of a service.

When that happens, visibility is controlled, versions are managed, proposed changes are analyzed and communicated, usage is monitored, and other parts of the SOA foundation can access service metadata with the confidence that they have found the copy of record.

In this context, WSRR handles the metadata management aspects of operational services and provides the system of record for these metadata artifacts. The place where anybody looking for a catalogue of all services deployed in or used by the enterprise goes first.

WSRR provides registry functions that support publication of metadata about the capabilities, requirements, and semantics of services that allow service consumers to find services or to analyze their relationships. And it provides repository functions to store, manage, and assign a version number to service metadata. It also supports governance of service definitions. This provides the following capabilities:

- ▶ Control access to service metadata
- ▶ Model life cycle of service artifacts
- ▶ Manage promotion of services through phases of their life cycle in various deployment environments
- ▶ Perform impact analysis and communicate changes to the governed service metadata

A business profile is a complete re-usable configuration of WSRR. Later, we describe how Trucks Inc. is leveraging the Governance profile, a business profile providing basic capability to implement governance policies.

## **Business goals for WSRR**

Trucks Inc. is moving to service-oriented architectures (SOA) to give it increased business agility. Central to this move is the need for a service registry to help manage and govern the services and their business processes.

WSRR addresses this need and the following business goals through service metadata management:

- ▶ **Business process vitality**

Trucks Inc. needs to change quickly to address market needs. WSRR provides the following features to support this goal:

- Rapid selection of different business service providers.

These are published in the registry and allow applications to be routed to them.

- Policy changes to modify behavior.

Policies are published in the registry together with policy references provided by and required by individual services.

- ▶ **Business governance**

This ensures that the business processes are legal, auditable, and mapped correctly to the IT services. WSRR provides the following features to support this goal:

- Policy enforcement and change auditing

This provides basic life cycle state representations of business services and metadata in the registry, where any changes can be validated against policies and notified to interested parties.

- Security

Security to control who can change business service metadata and policies. This provides role-based access control for service metadata changes or life cycle transitions.

► IT solution: Time to value

As Trucks Inc. bring business changes to the market quickly and at a known or lower development cost. WSRR provides the following features to support this goal:

– Development of new services.

The registry ensures reuse of interfaces and existing approved services promoting interoperability and faster integration.

– Reconfiguration and assembly of existing services.

WSRR builds on existing interfaces, protocols, policies, and frameworks. This promotes the reuse and adoption of standard policies.

► IT governance

To ensure that the changed systems perform as intended and no other systems are impacted, WSRR provides service life cycle management. The registry stores information about life cycle status and other metadata that is used to filter visibility of endpoints according to usage and context. This metadata includes information about the development, assembly, testing, deployment, operation, and version number.

► Value-based quality of service

This ensures that you pay for the quality of service you need, and not what you have installed. The Policy-based service selection registry manages runtime mediations using policies and metadata.

► Business management

This helps you to monitor what is needed to effectively manage your business and IT solutions. The Policy-based logging registry stores metadata that defines and controls logging and process monitoring.

► Migration

This allows the successful movement from traditional architecture to SOA. Discovery and management tools work alongside the registry to help populate it with standards-based mappings from traditional applications.

## WSRR and the SOA service life cycle

The SOA life-cycle as described in Figure 6-26 is summarized in Table 6-4.

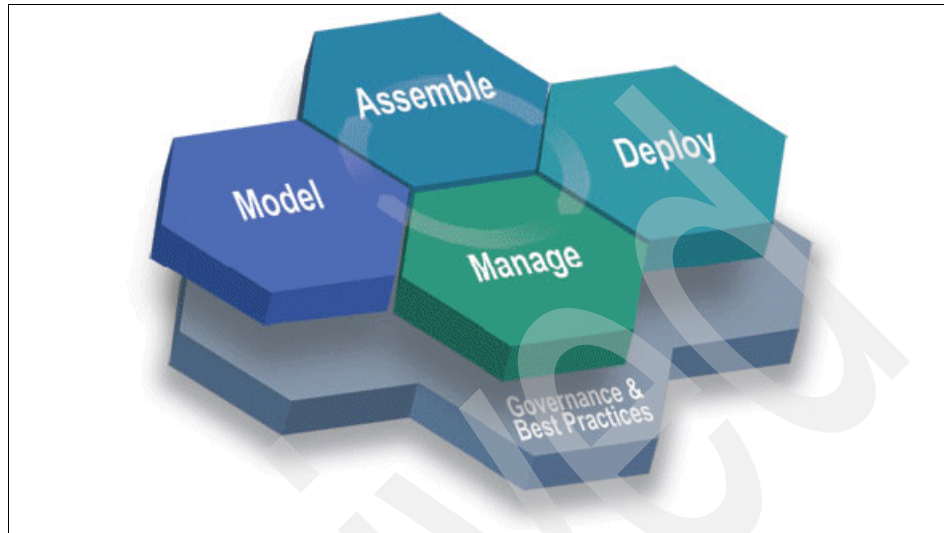


Figure 6-26 WSRR and the SOA service life-cycle

Table 6-4 SOA Life-cycle phases and how WSRR supports each of these phases

Phase	Description	WSRR support
Model	<ul style="list-style-type: none"><li>► Gather requirements</li><li>► Model and simulate</li><li>► Design</li></ul>	During the model and assemble phases of the service life cycle WSRR is used to locate the copies of record of candidate service interaction metadata or intermediaries as well as policies governing the interactions. WSRR can also be used to publish and govern service metadata about emerging, to-be-deployed services.
Assemble	<ul style="list-style-type: none"><li>► Discover</li><li>► Construct and test</li><li>► Compose</li></ul>	
Deploy	<ul style="list-style-type: none"><li>► Integrate people, process, information</li></ul>	WSRR provides the system of record for metadata describing service interaction endpoints. It is populated with metadata as part of SOA solution deployment or through the discovery of existing endpoints.
Manage	<ul style="list-style-type: none"><li>► Manage applications and services</li><li>► Manage identity and compliance</li><li>► Monitor business metrics</li></ul>	Operational management and resilience in the SOA is enhanced by sharing the service metadata that exists in WSRR with operational data stores.



## WSRR and SOA Governance

WSRR plays a key role in the end-to-end governance underpinnings of the SOA life cycle. WSRR supports governance of service metadata throughout the life cycle of a service from its initial publication in a development space through deployment to service management. WSRR is used in the life cycle phases of the governance cycle as described in Figure 6-27 and Table 6-5 on page 164:

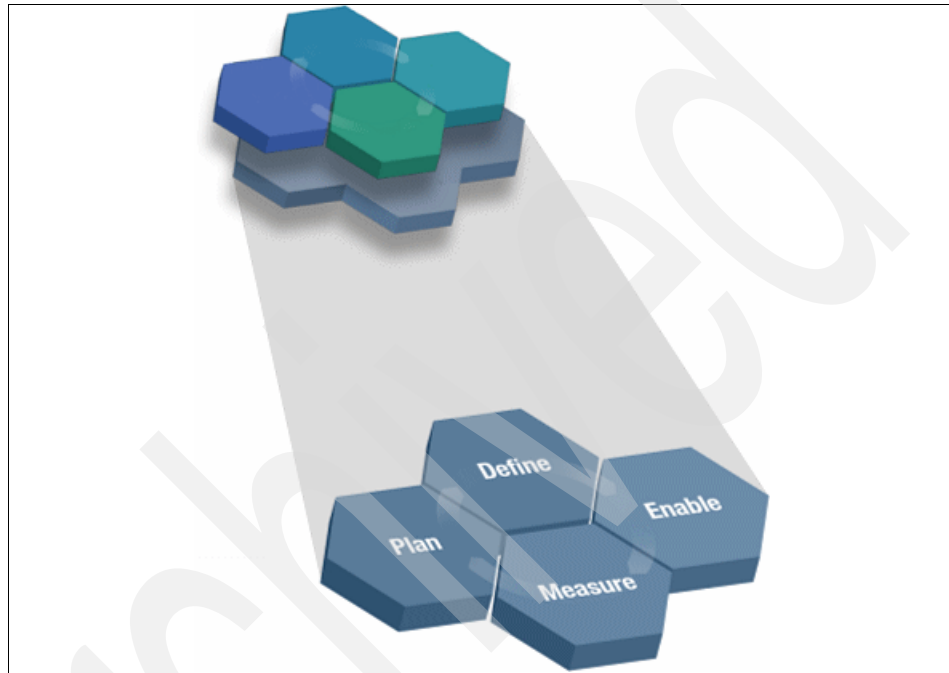


Figure 6-27 WSRR and SOA Governance phases

Table 6-5 SOA Governance phases

Governance Phase	Description
Enable	This is the most important phase, where governance policies and processes are implemented, for example, to implement service life cycle models or metadata validation rules, or to define access control for service metadata.
Measure	Governance requirements are monitored and measured and WSRR information allows for the monitoring process.
Plan	Governance needs are defined, for example, to review existing governance capabilities realized in WSRR or to define the basic life cycle model for governed entities.
Define	The Governance approach is specified, for example, to capture details of the life cycle models for service artifacts or SLAs governing service interactions.

### Using the WSRR Governance profile at Trucks Inc.

In WebSphere Service Registry and Repository (WSRR), a complete configuration of WSRR is called a *business profile*.

Business profiles can be used to capture the way SOA governance will be applied. Service-oriented modeling and architecture (SOMA) and SOA Governance and Management Method (SGMM) can help understand how to apply these profiles for the specific needs of a company. SOMA is described previously in 6.2, “Service Design” on page 126. In Table 6-6, the WSRR Business Profile lists our business profile elements and their descriptions:

Table 6-6 WSRR Business Profile

Business Profile element	Description
Business Models	To define user-defined models supporting concepts. Later, we give a sample view of Trucks Inc.’s business model leveraging the WSRR Governance profile.
Access control	To support service metadata visibility and life cycle decision rights.
Web UI Views	To display the concepts in a way that is suitable for their intended use.
Roles and perspectives	Additional roles and perspectives to support those roles.
Taxonomies	To support Trucks Inc. user-defined models, business domains and technical domains of relevance.
Life cycles	Appropriate to the services life cycle and its governance as aligned with the methods being used (SOMA, SGMM).
Configuration of user defined validators and notifiers	To implement governance policies.

Trucks Inc. decided to start configuring WSRR based upon the Governance profile, a business profile shipped with WSRR.

### ***WSRR Governance profile overview***

This section provides an overview of the governance profile's business model. Later, we provide samples applied to Trucks Inc. based upon this model. We will now review the main components of the WSRR Governance profile: Composite Application Representation, Service ownership and funding, Service consumption and contracts, and Business to technical mapping.

► Composite application representation (Figure 6-28)

One of the key goals in SOA (especially in the assemble phase of the service life cycle) is to be able to respond rapidly to business challenges and produce new or changed business applications by reconfiguring existing services. To support this, the WSRR governance profile identifies specific relationships between the business concepts used to represent applications, processes, and services. Because these representations can be mapped to existing deployed services, assembly can theoretically be undertaken without a slow development test deploy cycle.

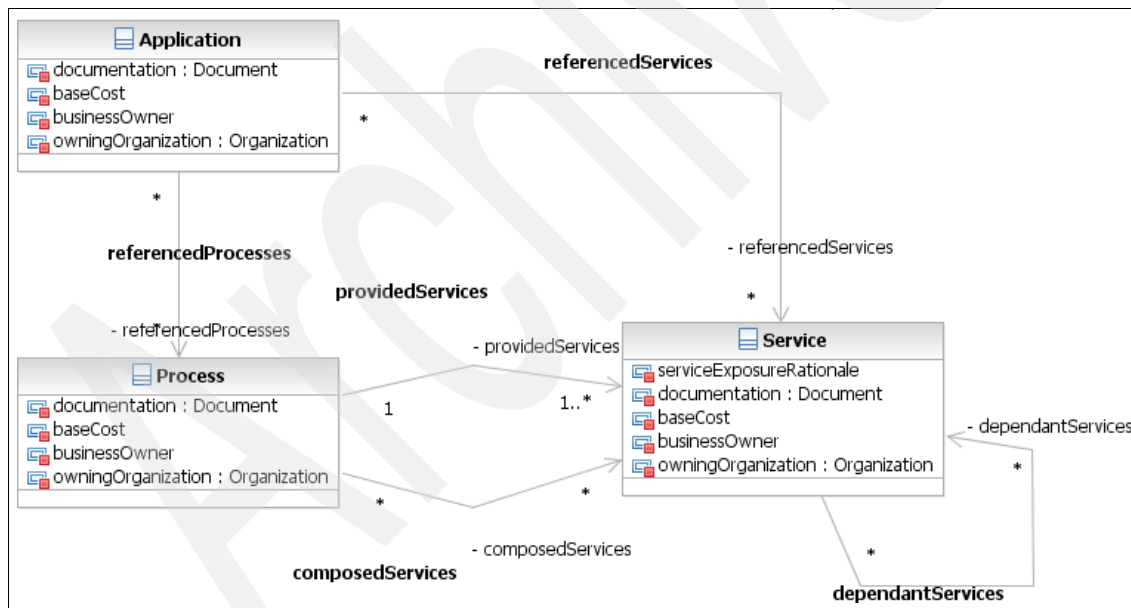


Figure 6-28 Composite application representation

► Service ownership and funding (Figure 6-29)

One of the most important SOA governance requirements is to ensure that all SOA artifacts (such as applications, processes and services) are assigned an owner. The owner has total responsibility for the artifact including the following issues:

- Responsibility for reconciling requirements
- Responsibility for agreeing contracts and SLAs
- Responsibility for delivering to agreed SLAs
- Responsibility for service life cycle versioning and retirement

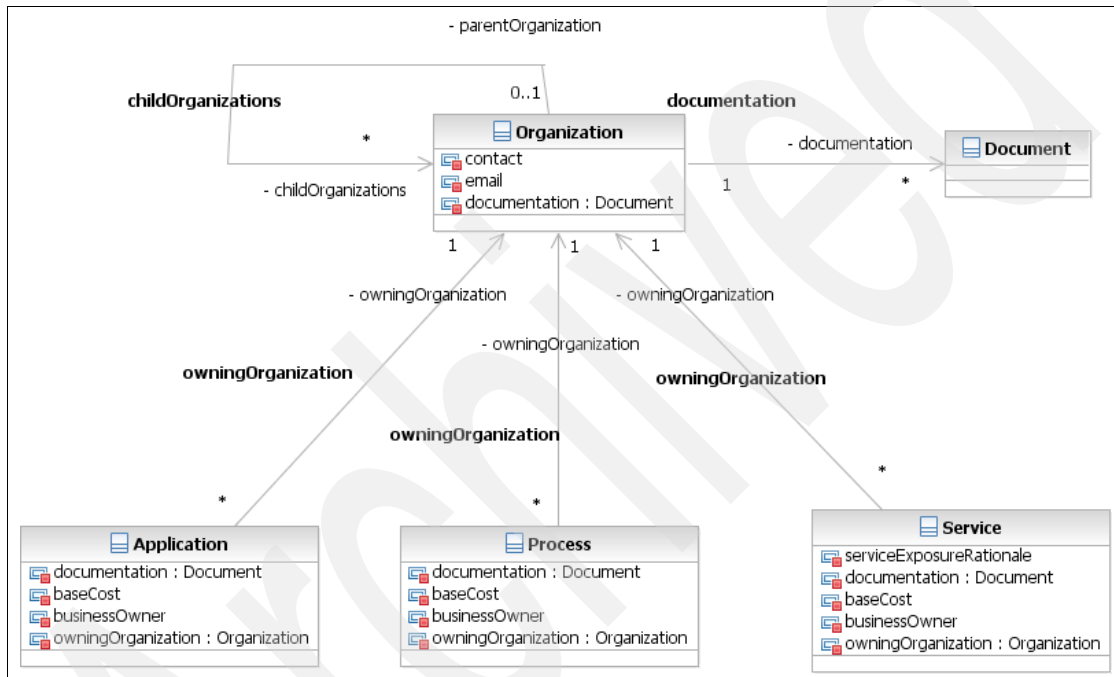


Figure 6-29 Service ownership and funding

► Service consumption and contracts (Figure 6-30)

When services are shared between lines of business, it becomes important to ensure that a clear understanding (or contract) is established between the provider and the consumer for the use of those services.

This is represented in the Governance Profile by a contract. The structure of a contract is shown in the diagram Figure 6-30.

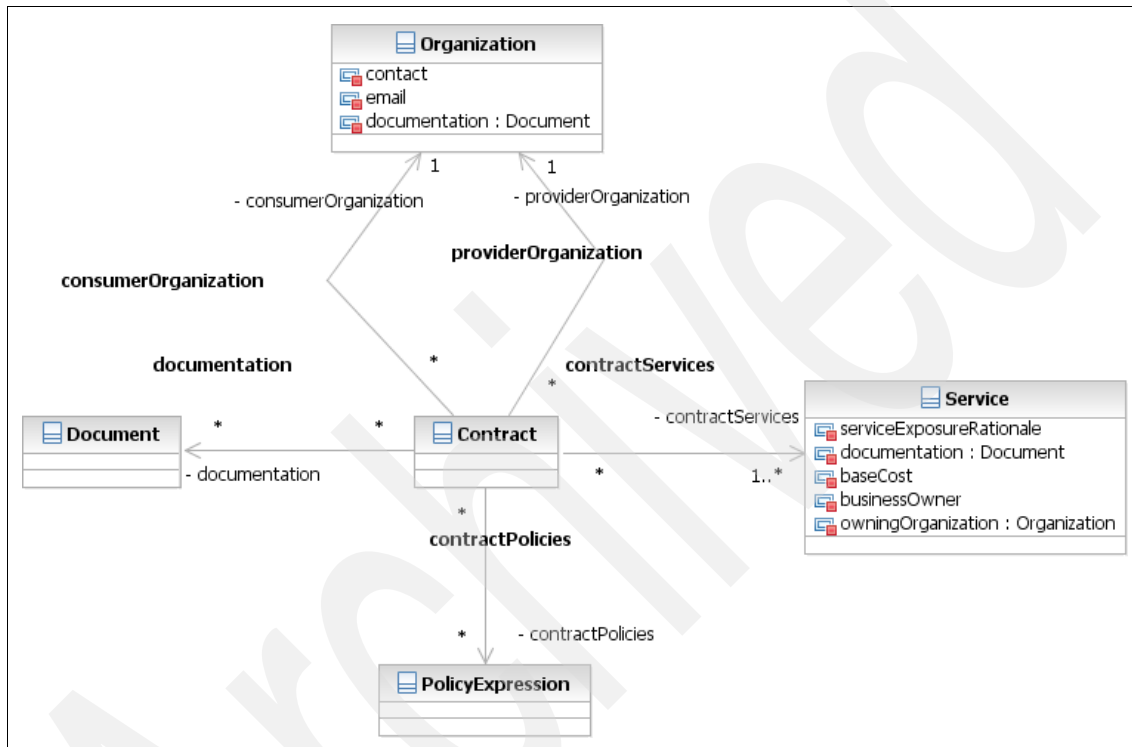


Figure 6-30 Service consumption and contracts

► Business to technical mapping (Figure 6-31 on page 168)

WSRR provides strongly typed representation of technical SOA artifacts defined in documents such as WSDL and XML schemas. To help in the mapping from business to technical perspectives, certain key elements from these files are brought out as views with different names.

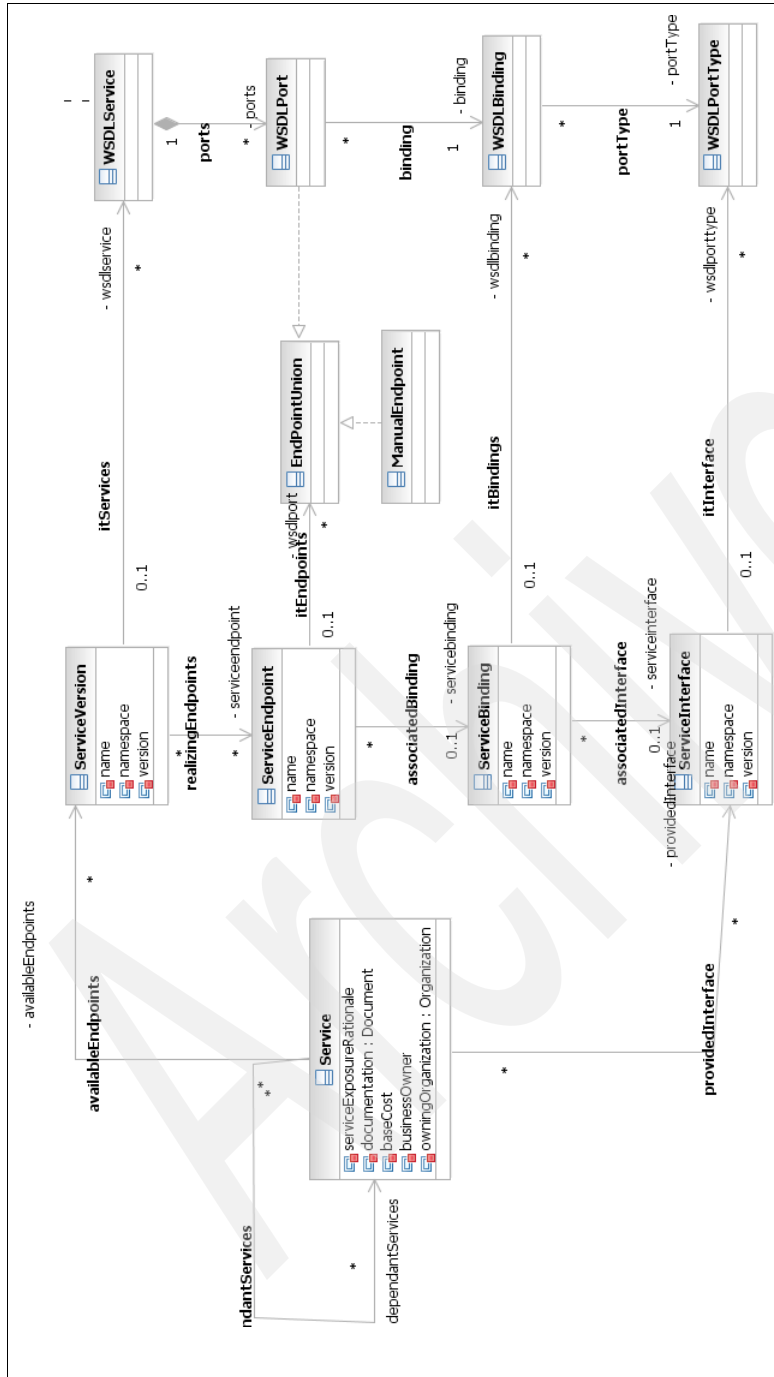


Figure 6-31 Business to technical mapping

► Business model concepts

Figure 6-32 presents a sample of concepts applied to Trucks Inc. within WSRR.

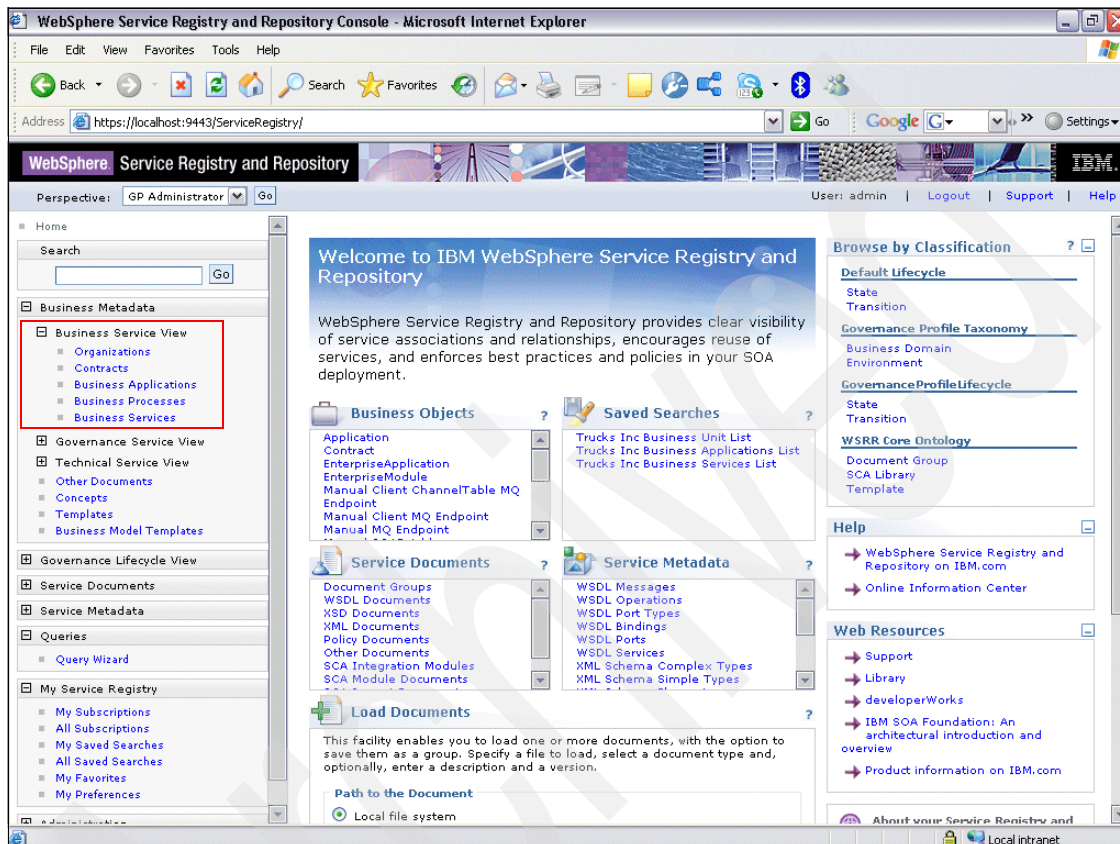


Figure 6-32 WSRR Governance profile business concepts

The key concepts relating to a business model are as follows:

– Business application

The business application concept represents high-level business applications that fulfil a direct business need. Applications provide a root for mapping the business applications to the IT implementations.

At Trucks Inc., typical business applications are Enterprise PDM, Mechanical CAD, Requirements Management, Portfolio Management, Simulation Management, and so forth.

Figure 6-33 shows some of the business applications configured by Trucks Inc. in WSRR:

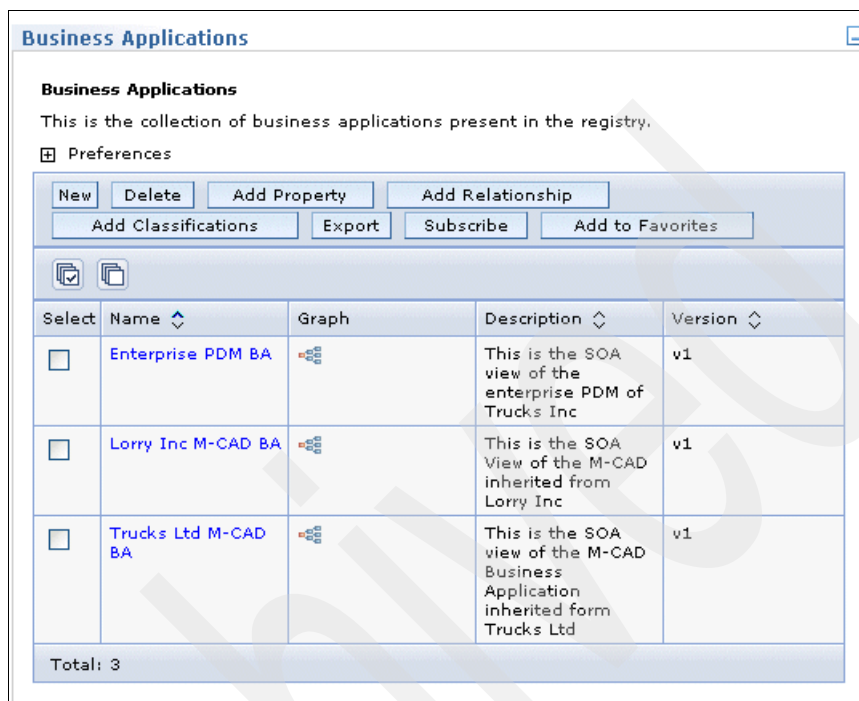


Figure 6-33 Trucks Inc. Business applications in WSRR

#### – Process

Processes represent common (shared by multiple applications) business processes. Processes have dependencies on other processes and services and are usually implemented by assembly of those dependant services through a tool such as WebSphere Integration Developer (WID) and realized through WebSphere Process Server (WPS).

Some typical business processes for a company like Trucks Inc. are described Figure 6-6 on page 131.

#### – Service

The services concept represent a common (shared) service at a business level. This is effectively the cataloged item in SOA. WSDL interfaces and endpoints would be referenced from here.



Figure 6-34 shows some Business Services used by Trucks Inc. and governed from WSRR.

Business Services

Business Services

This is the collection of business services present in the registry.

Preferences

NewDeleteAdd PropertyAdd RelationshipAdd ClassificationsExportSubscribe

Add to Favorites

Select	Name	Graph	Description	Version
<input type="checkbox"/>	Browse Abstract Product Structures BS		This Business Service Covers OMG PLM Services 2.0 Browsing of Abstract Product Structures Use Case	v2
<input type="checkbox"/>	Browse APS Alternative Solutions BS		This Business Service Covers OMG PLM Services 2.0 Browsing of Alternative Solutions within an Abstract Product Structure Data Use Case	v1
<input type="checkbox"/>	Browse Down Product Structure Data BS		This Business Service Covers OMG PLM Services 2.0 Browsing Down Product Structure Data Use Case	v5
<input type="checkbox"/>	Browse Up Product Structure Data BS		This Business Service Covers OMG PLM Services 2.0 Browsing Up Product Structure Data Use Case	v13
<input type="checkbox"/>	Change Notification BS		This Business Service Covers OMG PLM Services 2.0 Change Notification Use Case	v1

Figure 6-34 A view of the Business Services used by Trucks Inc. in WSRR

Let us take the example of the Business Service Export Assembly Data BS. Its main parameters are entered in WSRR in the window shown in Figure 6-35.

From this window, the user can perform the following functions:

- Attach a document describing what this business service is used for
- Set the organization that is in charge of this business service

**Trucks Inc Business Services > Export Assembly Data BS**

Details of the Export Assembly Data BS Business Service.

Details | Impact Analysis | Governance

**Entity Properties**

Name  
Export Assembly Data BS

Version  
v13

Description  
This Business Service Covers OMG PLM Se

Business owner  
Lionel Momméja

Service exposure rationale  
This is a core business service

Cost per Transaction  
1\$

Apply OK Reset Cancel

**Entity Metadata**

- Graphical View
- Properties
- Custom Relationships
- Classifications

**Entity Policies**

- Policies

**Entity Relationships**

- Documentation
- Owning Organization
- Governed Service Interface
- Realizing Governed Services
- Service dependencies

**Relationships to Entity**

- Dependant Services
- Dependant Processes
- Dependant Applications

Figure 6-35 Details of the Export Assembly Data Business Service

If you were to click **Graphical View**, you would see a window similar to Figure 6-36.

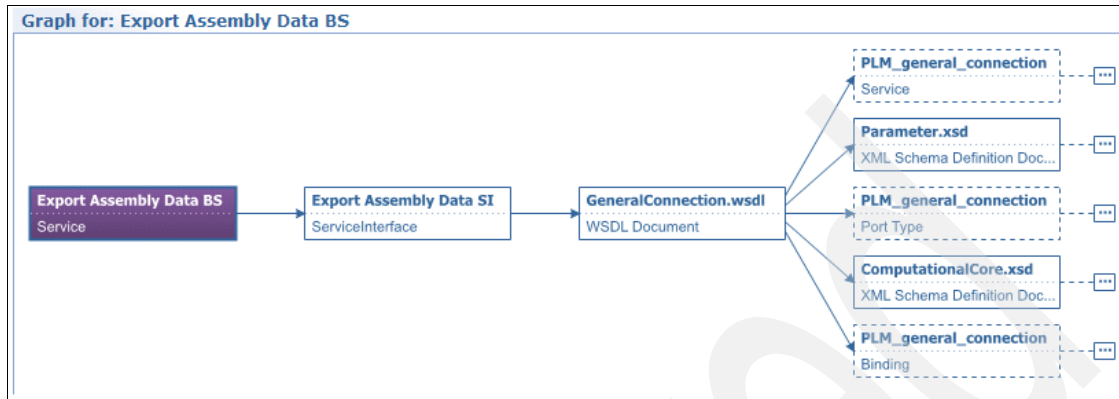


Figure 6-36 The graph view of Export Assembly Data Business Service show dependencies with other artifacts

- Organization

An organization represents an organizational unit at a business level. Effectively a line of business. This high-level representation is often needed in SOA to identify the provider or consumer of services and the prime owner in an organizational sense.

- Contract

A contract represents an agreement between two organizations about services provided and the policies agreed between provider and consumer organization.

Archived



## Part 3

# Overview of working examples

In the previous chapters, we explored those segments of PLM to help you grasp an in-depth understanding of the IBM approach to enable a service-oriented architecture for PLM. In the following chapters, we provide you with implementation procedures based on real-life experiences, as well as work performed in our Center for Excellence labs.

- ▶ Chapter 7, “Implementing PLM Services 2.0-based solutions on IBM middleware” on page 177

This chapter describes the steps to set up the development environment. The chapter introduces several utilities that help implement PLM Services 2.0 solutions on IBM middleware. This includes the following utilities:

- PLM Services library containing the PLM Services 2.0 interfaces and data types
- JAX-RPC handler to deal with sessionIds in the SOAP header
- Java wrapper classes for the PLM Services data types

In addition, this chapter provides step-by-step instructions on how to implement a PLM Services 2.0 compliant server. This server is tested using ProStep iVip's PLM Services 2.0 reference client as outlined in Figure 7-25 on page 212. Chapter 7, "Implementing PLM Services 2.0-based solutions on IBM middleware" on page 177 also outlines the necessary steps on how to provide a PLM Service 2.0 wrapper for existing PDM systems.

- Chapter 8, "A framework to develop and run flexible PLM Services 2.0-based solutions" on page 221

This chapter builds on the artifacts introduced and created in Chapter 7, "Implementing PLM Services 2.0-based solutions on IBM middleware" on page 177 but focuses on flexibility and reusability. A framework is introduced that acts as intermediate between PLM Services clients and PDM and PLM systems and allows to execute business logic (hosted, for example, on WebSphere Process Server) upon client requests. Figure 8-1 on page 223 gives a high-level overview of the framework. The simple query scenario demonstrated in Chapter 7, "Implementing PLM Services 2.0-based solutions on IBM middleware" on page 177 is extended to a federated query over two PDM systems.

- Chapter 9, "Design and implementation of an engineering change scenario including SAP integration" on page 255

This chapter uses the components of Chapter 7, "Implementing PLM Services 2.0-based solutions on IBM middleware" on page 177 and Chapter 8, "A framework to develop and run flexible PLM Services 2.0-based solutions" on page 221 to build a more complex scenario: An implementation of an engineering change request (ECR) following the outline of ECR interaction scenario 1 'ECR participant proposal for a change' defined in the VDA 4965-1 recommendation. In case the ECR is finally approved, the changes are propagated to an SAP system. As not all readers have access to a real SAP system, the scenario can also be configured to run without a real SAP system. Figure 9-1 on page 256 outlines the scenario of Chapter 9, "Design and implementation of an engineering change scenario including SAP integration" on page 255.

- Chapter 10, "SOA realization of integrated PLM solutions: Working with examples" on page 403

This chapter discusses the role and implementation of WSRR and WebSphere Enterprise Service Bus (WESB). In addition, this chapter discusses the implementation to integrate the WSRR in the dynamic routing process in an SOA-based OMG PLM Services 2.0-compliant solution.

# Implementing PLM Services 2.0-based solutions on IBM middleware

The first part of this book covered PLM in general and addressed the challenges of integrating multiple PLM systems. In the second part, the IBM approach to solve these issues through SOA was discussed. After the theory in the first two parts, this chapter, and the ones that follow, focuses on and demonstrates the development of SOA-based PLM Services 2.0-compliant solutions on IBM middleware.

This chapter includes the following topics:

- ▶ Installation of WebSphere Integration Developer 6.1
- ▶ Download and use of the ProStep iViP PLM Services reference client
- ▶ Introduction of utilities that help you build PLM Services 2.0-based solutions: a PLM Services library containing the PLM Services 2.0 interfaces and data types, a JAX-RPC handler to deal with sessionIds in the SOAP header, Java wrapper classes for the PLM Services data types.
- ▶ Implementation of a PLM Services 2.0 compliant server on WebSphere Process Server.
- ▶ Discussion on how to implement PLM Services 2.0 connectors to existing PDM systems by exploiting the SCA features and how to handle events and requests from the PDM system.

**Note:** Part III requires familiarity and a basic understanding of the following products and topics:

- ▶ WebSphere Integration Developer
- ▶ WebSphere Process Server
- ▶ Service Component Architecture (SCA)
- ▶ Service Data Objects (SDO)
- ▶ Java

Visit the following Web pages to learn more about these products and topics:

- ▶ WebSphere Process Server Infocenter:  
[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome\\_top\\_wps.htm](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm)
- ▶ Articles on Developer Works:  
<http://www.ibm.com/developerworks/websphere/>

## 7.1 Preparing the development environment

Trucks Inc. must integrate the existing PLM applications of the two merged companies. Their integration solution must be highly flexible, and needs to promote the reusability of existing IT assets. Also, time to the market is one of the pain points that the CIO of the Trucks Inc. pointed out earlier (see other pain points stated in the opening paragraphs in Chapter 5, “Implementing end-to-end Business Process Management” on page 103). As the company continues to grow, integrating additions and changes to their systems must be possible with reasonable effort.

As their SOA-based solution platform, Trucks Inc. has chosen IBM WebSphere Process Server. WebSphere Process Server is a business integration server, it implements a Web Services Business Process Execution Language (WS-BPEL) V2.0 compliant process engine. The adaptation of Service Component Architecture (SCA) promotes reusability and visibility of existing IT assets in their composite integration applications. The Figure 7-1 on page 179 gives a brief overview of the building blocks of WebSphere Process Server.



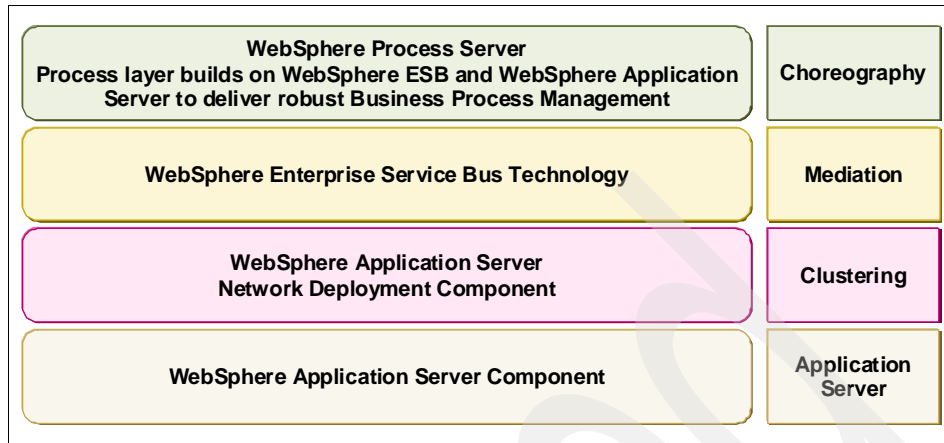


Figure 7-1 WebSphere Process Server product overview

WebSphere Application Server V6 Network Deployment provides WebSphere Process Server with J2EE functionality including high availability, workload management and security. The WebSphere ESB technology provides a communication backbone for Service Oriented Integration (SOI).

IBM WebSphere Integration Developer provides an unified environment for building SOA-based integration applications to run on WebSphere Process Server, as well as WebSphere ESB.

The integration solution for Trucks Inc. is implemented and tested on WebSphere Integration Developer. The sample applications introduced in this chapter were developed using WebSphere Integration Developer V6.1, and tested on the WebSphere Process Server V6.1 test environment

## 7.1.1 Installing WebSphere Integration Developer V6.1

The detailed installation instructions of WebSphere Integration Developer V6.1 are available through IBM Information Center.

[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.install.doc/topics/c\\_inintro.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.install.doc/topics/c_inintro.html)

To recreate the same development environment, perform the following steps during your installation of WID:

1. Select the IBM WebSphere Integration Developer and IBM WebSphere Process Server or WebSphere Enterprise Service Bus Test Environment check boxes from the Installation, as shown in Figure 7-2.



Figure 7-2 Installation package

Leave the default options selected in the Features panel as shown in Figure 7-3. You may select the Portal tools radio button if you wish.



Figure 7-3 Features

2. Click **Next** to continue. Click **Finish** to complete installation

When you have completed the installation, a WebSphere Process Server profile must be created with default settings.

## 7.1.2 The Prostep iViP PLM Services 2.0 reference client

This Redbooks publication focuses on the implementation of PLM Services 2.0 server-side components. In order to demonstrate also end-to-end scenarios, we are using ProStep iViPs PLM Services 2.0 reference client.

### Downloading the PLM Services 2.0 reference client

Download and install the ProSTEP iViP PLM Services V2.0 reference client. This application is a PLM Services 2.0 specification compliant client, and downloadable from the ProSTEP iViP Web site.

Follow the instructions below to download and install the reference client.

1. Open a Web browser, and navigate to the following Web site:  
<http://www.prostep.org/en/projektgruppen/pdm-if/plmservices.htm>
2. Click the Demonstrator-Client (Documentation) link to download the Usage of the Client for OMG PLM Services Version 2.0 document. Read the documentation to get a basic understanding on how the reference client is used.
3. Click the Demonstrator-Client (Source and Binaries) link to download plms\_client\_061230.zip file to a temporal folder.
4. Extract the contents of the plms\_client\_061230.zip file to your desired install location.

**Note:** The PLM Services reference client requires Java Runtime Environment V1.4 or above to run. If the ProSTEP application does not start due to missing JRE™, append  
<WID\_HOME>\runtimes\bi\_v61\java\jre\bin to your system Path variable.

5. Double-click PLM-Services V2.0 Client.exe under the installed location to verify that the ProSTEP iViP reference client starts with no errors.

## 7.2 PLM Service 2.0 library and Java utilities

This section introduces some utilities that help you building PLM Services 2.0-based solutions. The utilities are packaged as a library containing the PLM Services 2.0 interfaces, data types, and a Java project.

## 7.2.1 Importing PLM Services library and Java utilities in WebSphere Integration

To review the utilities, import the PLM Services library and Java utilities in WebSphere Integration Developer. Perform the following steps to import the library and utilities:

1. Start WebSphere Integration Developer with a new workspace.
  - a. Select **Start** → **Programs** → **IBM WebSphere Integration Developer** → **IBM WebSphere Integration Developer V6.1** → **WebSphere Integration Developer V6.1**.
  - b. Specify the workspace location you wish to create (Figure 7-4), and click **OK**.

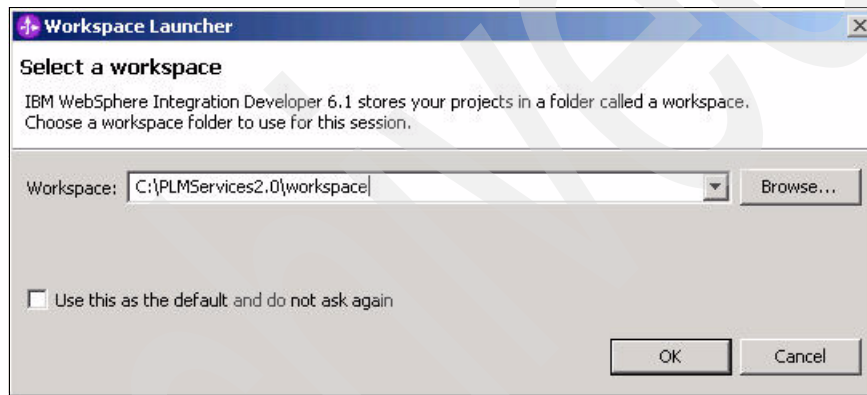


Figure 7-4 Workspace launcher

WebSphere Integration Developer starts and welcome panel opens as shown in Figure 7-5.

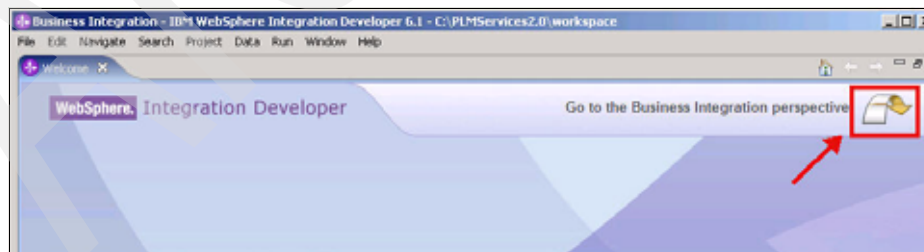


Figure 7-5 WebSphere Integration Developer Welcome panel

- c. Click the arrow highlighted in Figure 7-5 to close the welcome panel, which brings you to the Business Integration window.

2. Import the PLM Services 2.0 library (PLMSServices2WithSessionIDLib) and Java utility project (PLMS2Util) into your workspace.
  - a. Select **File** → **Import...** The Select window opens
  - b. Expand **Other** node in the Select window, select Project Interchange, then click **Next >**.
  - c. Go to Additional Materials folder for Redbook SG24-7593, and download the lab file Redbook247593\_lab.zip. Extract the files for use. See Appendix A, “Additional material” on page 465.
  - d. Click **Browse...** in the Import Projects window, and navigate to the directory to which you extracted Redbook247593\_lab.zip lab file.
  - e. Select Utilities\_PL.zip, and click **Open**.
  - f. Select PLMSServices2WithSessionIDLib and PLMS2Util, then click **Finish**.

## 7.2.2 PLM Services library

The PLMServices2WithSessionIDLib library contains slightly modified versions of the PLM Services 2.0 Web Services platform specific model (PSM) XSD and WSDL files (see Figure 7-6 on page 184). In addition, the library contains the following extensions specified by the VDA to support ECR and ECM scenarios:

- ▶ InformationalECM.xsd
- ▶ InformationalECR.xsd, ComputationalECM.xsd
- ▶ ComputationalECR.xsd

**Note:** The PLM Services 2.0 WSDL and XML schema files can be downloaded from the following Web site:

<http://schema.omg.org/specs/PLM/2.0/>

The VDA 4965 ECR/ECM files can be downloaded from the following Web site:

<http://www.prostep.org/en/standards/doku/standards/>

Name	Size	Type
ConnectionFactory.wsdl	6 KB	WSDL File
GeneralConnection.wsdl	15 KB	WSDL File
MessageConnection.wsdl	11 KB	WSDL File
PLMServicesAll.wsdl	2 KB	WSDL File
SchemaInfo.xsd	2 KB	XSD File
ComputationalCore.xsd	4 KB	XSD File
Exception.xsd	3 KB	XSD File
GenericQueries.xsd	6 KB	XSD File
Information.xsd	2 KB	XSD File
InformationalCore.xsd	3 KB	XSD File
InformationalModel.xsd	211 KB	XSD File
MessageQueries.xsd	2 KB	XSD File
Parameter.xsd	10 KB	XSD File
PdtnetQueries.xsd	11 KB	XSD File
ProxyQueries.xsd	5 KB	XSD File
SpecificQueries.xsd	31 KB	XSD File
UtilityQueries.xsd	3 KB	XSD File
xml.xsd	3 KB	XSD File
XPathQueries.xsd	1 KB	XSD File

Figure 7-6 PLM Services 2.0 XSD and WSDL files

## Modifications

To enable successful completion of query results, the element name of the Container\_Parameter complex type (defined in Parameter.xsd) was changed from 'return' to 'queryReturn' by applying the code in Example 7-1.

### Example 7-1 Changing 'return' to 'queryReturn'

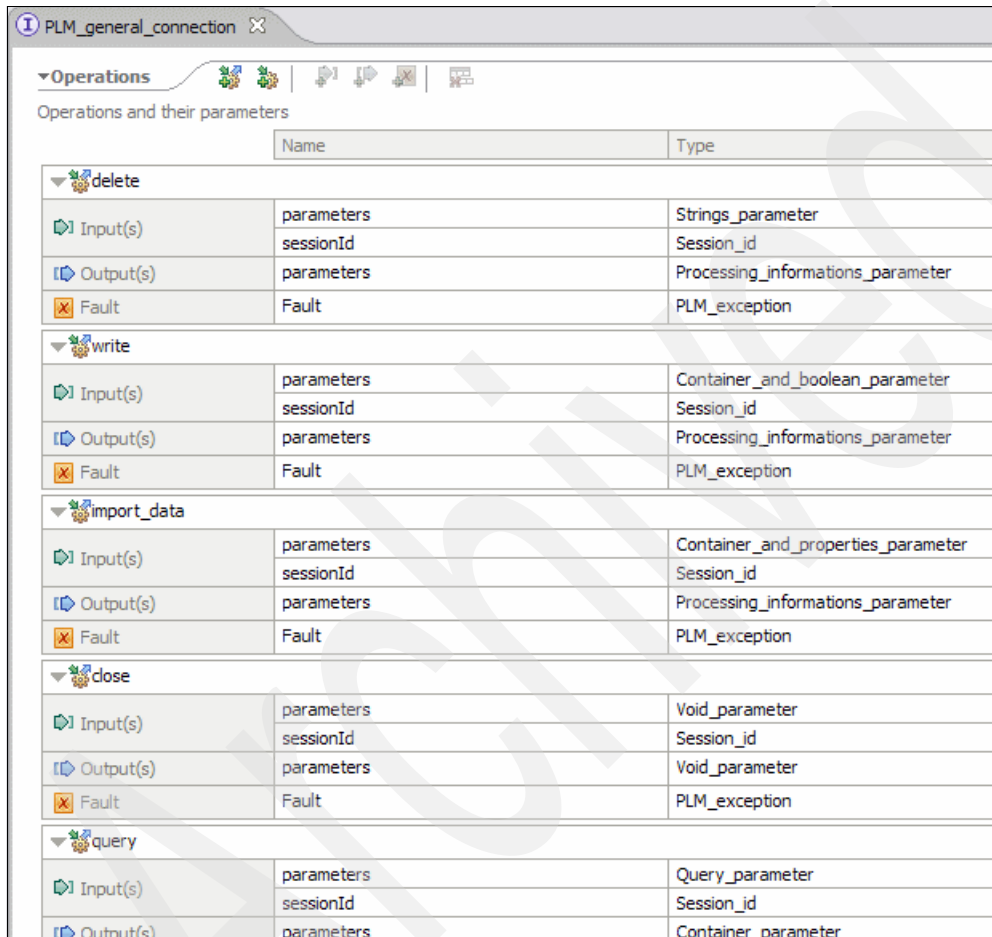
```
<xsd:complexType name="Container_parameter">
  <xsd:sequence>
    <xsd:element xmlns:ns3="urn:omg.org/plm20/informational/core"
form="qualified" name="queryReturn" type="ns3:PLM_core_container"/>
  </xsd:sequence>
</xsd:complexType>
```

Without that change the PLM Services reference client would not recognize responses of query requests.

## Modifications to ease processing of the sessionId

According to the PLM Services 2.0 Web Services PSM, the sessionId (returned by a previous get\_general\_connection/get\_message\_connection call on the PLM\_connection\_factory interface) has to be specified in the SOAP header when sending PLM\_general\_connection/PLM\_message\_connection requests. In order to release the developer from the burden to process the SOAP header when

receiving client requests, the operations in the `PLM_general_connection` and `PLM_message_connection` interface were modified by adding the `sessionId` as an additional input parameter (see Figure 7-7). This makes it easier for the developer to access this parameter in the implementation of a PLM Services 2.0 server or connector.



The screenshot shows a software interface titled "PLM\_general\_connection". It features a tabbed view with "Operations" selected. Below the tab, there's a section "Operations and their parameters" containing a table. The table lists five operations: delete, write, import\_data, close, and query. Each operation has four rows of parameters: Input(s), Output(s), and Fault. The "delete" operation has parameters "parameters" (Strings\_parameter), "sessionId" (Session\_id), "parameters" (Processing\_informations\_parameter), and "Fault" (PLM\_exception). The "write" operation has parameters "parameters" (Container\_and\_boolean\_parameter), "sessionId" (Session\_id), "parameters" (Processing\_informations\_parameter), and "Fault" (PLM\_exception). The "import\_data" operation has parameters "parameters" (Container\_and\_properties\_parameter), "sessionId" (Session\_id), "parameters" (Processing\_informations\_parameter), and "Fault" (PLM\_exception). The "close" operation has parameters "parameters" (Void\_parameter), "sessionId" (Session\_id), "parameters" (Void\_parameter), and "Fault" (PLM\_exception). The "query" operation has parameters "parameters" (Query\_parameter), "sessionId" (Session\_id), and "parameters" (Container\_parameter).

	Name	Type
<b>delete</b>		
Input(s)	parameters	Strings_parameter
	sessionId	Session_id
Output(s)	parameters	Processing_informations_parameter
Fault	Fault	PLM_exception
<b>write</b>		
Input(s)	parameters	Container_and_boolean_parameter
	sessionId	Session_id
Output(s)	parameters	Processing_informations_parameter
Fault	Fault	PLM_exception
<b>import_data</b>		
Input(s)	parameters	Container_and_properties_parameter
	sessionId	Session_id
Output(s)	parameters	Processing_informations_parameter
Fault	Fault	PLM_exception
<b>close</b>		
Input(s)	parameters	Void_parameter
	sessionId	Session_id
Output(s)	parameters	Void_parameter
Fault	Fault	PLM_exception
<b>query</b>		
Input(s)	parameters	Query_parameter
	sessionId	Session_id
Output(s)	parameters	Container_parameter

Figure 7-7 Modified version of the `PLM_general_connection` interface

In order to stay compliant with PLM Services 2.0 Web Services PSM, `PLM_general_connection` and `PLM_message_connection` requests are intercepted by a JAX-RPC handler that copies the `sessionId` from the SOAP header into the `sessionId` input parameter of the called `PLM_general_connection` and `PLM_message_connection` operation (see Example 7-2 on page 187 or open `PLMS2Util/com.ibm.pdf.framework.jaxrpc.handler.ServiceExportHandler.java`).

In addition, the WSDL binding of the PLM\_general\_connection and PLM\_message\_connection interfaces were modified to copy the sessionId parameter from the message body into the SOAP header when acting as PLM Service 2.0 client. Both scenarios are illustrated in Figure 7-8.

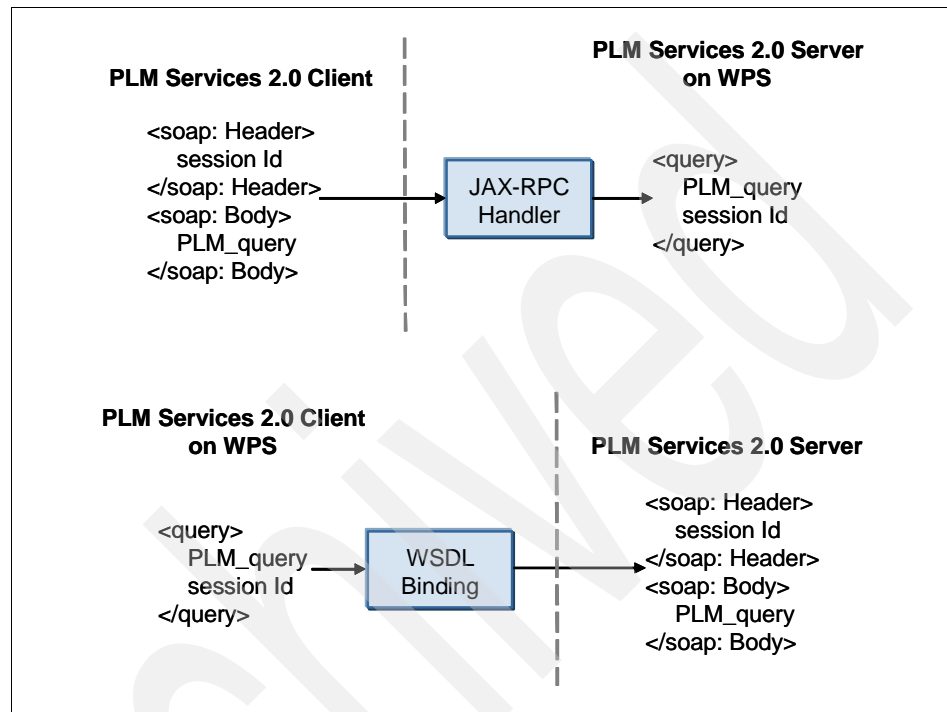


Figure 7-8 SessionId processing



```
public boolean handleRequest(MessageContext ctx) {
    try
    {
        SOAPMessageContext smc = (SOAPMessageContext) ctx;
        SOAPMessage sm = smc.getMessage();
        SOAPPart sp = sm.getSOAPPart();
        SOAPEnvelope se = sp.getEnvelope();
        SOAPHeader header= se.getHeader();

        String sessionId = null;
        Node currentNode = header.getFirstChild();
        while(currentNode != null)
        {
            String localname = currentNode.getLocalName();
            if(localname!=null &&
                localname.equalsIgnoreCase("session_id"))
            {
                sessionId =
                    currentNode.getFirstChild().getNodeValue();
            }
            currentNode = currentNode.getNextSibling();
        }

        SOAPBody body = se.getBody();
        Element el = body.getOwnerDocument().createElementNS(
            "urn:omg.org/plm20/services/parameter",
            "parameter:session_id");
        el.setAttribute("xmlns:parameter",
            "urn:omg.org/plm20/services/parameter");
        el.appendChild(body.getOwnerDocument().createTextNode(
            sessionId));
        body.appendChild(el);
    }
    catch(SOAPException e){
        e.printStackTrace();
        return false;
    }
    return super.handleRequest(ctx);
}
```

---

From the Business Integration window, perform the following steps to begin easing the processing of the sessionId (as shown in Figure 7-9):

1. Expand **PLMServices2WithSessionIDLib** → **Data Types**.

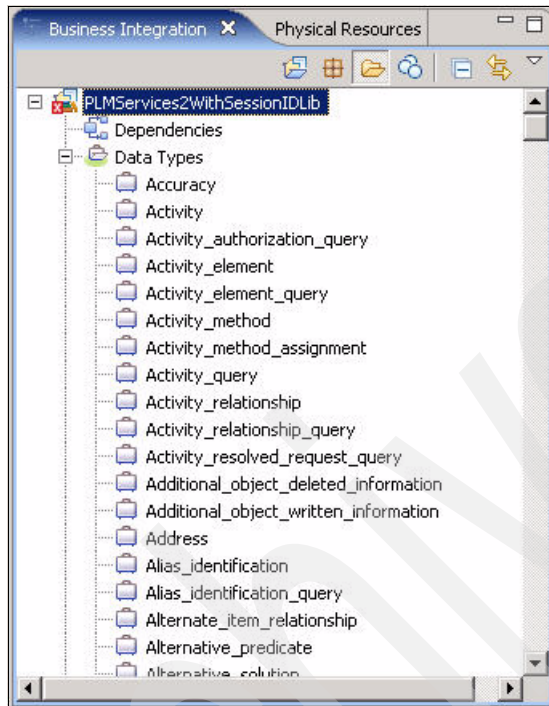


Figure 7-9 Business integration view with PLM Services 2.0 data types

- Double-click the Item business object to open it in the Business Object editor. You see a window similar to Figure 7-10.

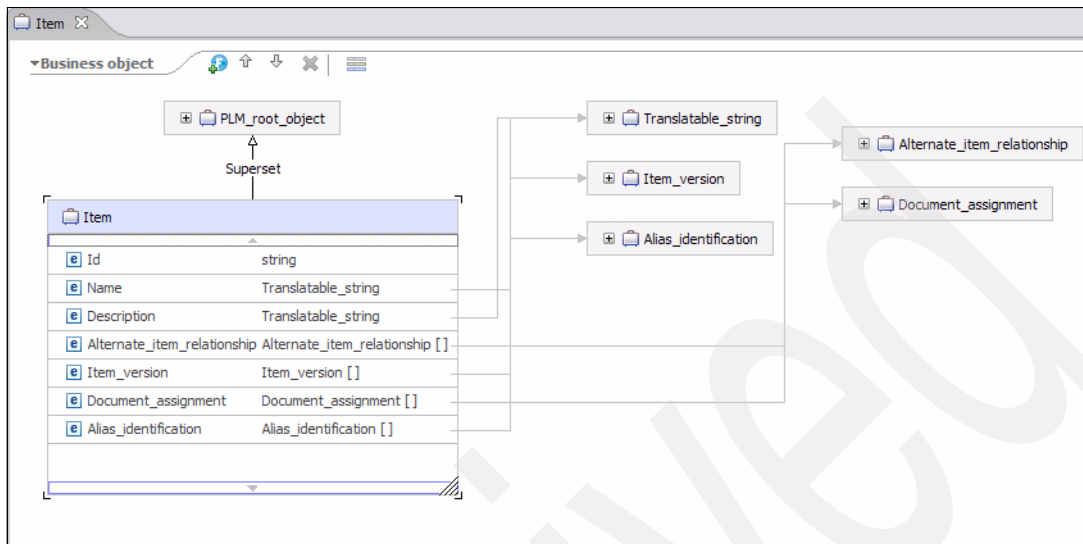


Figure 7-10 Item business object

- Double-click the PLM\_container business object to open it in the Business Object editor.
- Expand **PLMServices2WithSessionIDLib** → **Interfaces**.
- Double-click the **PLM\_connection\_factory** interface to open it in the **Interface** editor.
- Double-click the **PLM\_general\_connection** interface to open it in the **Interface** editor (see Figure 7-7 on page 185).
- Double-click the **PLM\_message\_connection** interface to open it in the **Interface** editor.

### 7.2.3 Java wrapper classes for the PLM Services 2.0 data types

The data model used in WebSphere Process Server is based on Service Data Objects (SDO), an open standard for enabling applications to handle data from heterogeneous data sources in a uniform way. SDO provides an API to access the data dynamically. For example, on an Item data object you could retrieve the id property (defined as xsd:string) by using the following code:

```
item.getString("id"). Or, you could retrieve the the Item_version property (defined as array of Item_version data type) through: item.getList("Item_version"). In order to access the data, you have to be aware of the name and the data types.
```

Considering the complexity of the PLM Services 2.0 data model, it is obvious that the use of the plain SDO API in context of PLM Services is elaborate and error prone. The exact spelling of properties and their type have to be looked up constantly. Therefore Java wrapper classes for each PLM Services 2.0 data type are provided in this Redbooks publication. They allow type-safe access and use the code completion facility (by pressing Ctrl-Space) of the Java editor. For example, the Item Java wrapper class provides a `getId()` method returning a String and a `getItemVersion()` method returning a `ItemVersionList`.

## Mapping between SDOs and the Java wrapper classes

As the WebSphere Process Server components pass and expect service data objects, a mapping between SDOs and the Java wrapper classes have to be performed.

The wrapper classes were designed with this requirement in mind. They provide get/set methods for every property of the respective PLM Services type. The get/set methods execute on an internal service data object. Every Java wrapper class has a `getSDO()` method that returns this internal service data object that can be consumed by the WebSphere Process Server components.

The opposite mapping, for example, from a service data object into the respective Java wrapper class can be done in two ways:

- Create a new instance of the respective Java wrapper class, passing the service data object, for example, `Item it = new ItemImpl(sdo);`

This approach requires knowledge of the exact PLM Services 2.0 type.

**Note:** For every PLM Services 2.0 type there is one interface with the name of the PLM Services type (Java naming conventions applied) and one implementation class for that interface named `<interface name>Impl`.

- Use the `PLMServicesUtil.getPLMServicesType()` method.

If you browse the PLM Services 2.0 data model, you see extensive use of inheritance. The `getPLMServicesType(DataObject sdo)` method considers that and returns an instance with the correct PLM Services type.

## Support for type-safe lists

The PLM Services 2.0 data model makes intensive use of lists/arrays. In order to support the developer, the Java wrapper classes contain additional type-safe list classes for every PLM Services 2.0 type that is used somewhere as list. For example, the `PLM_container` data object has a property `Item` of type `Item` that is defined as array. Therefore, the Java wrapper classes also contain a `ItemList`

(and ItemListImpl) class with type-safe access as outlined in Example 7-3 on page 191.

*Example 7-3 Type-safe lists, ItemList example*

```
public interface ItemList
{
    public List getSDOList();
    public void add(int index, Item element) ;
    public boolean add(Item o) ;
    public void clear() ;
    public boolean contains(Item o) ;
    public Item get(int index) ;
    public int indexOf(Item o) ;
    public boolean isEmpty() ;
    public int lastIndexOf(Item o) ;
    public Item remove(int index) ;
    public boolean remove(Item o) ;
    public Item set(int index, Item element);
    public int size() ;
}
```

### Miscellaneous utility methods

The PLMServicesUtil class in the omg.org.plm20.util package contains some additional utility methods as listed in Table 7-1.

*Table 7-1 Utility methods in PLMServicesUtil class*

Method	Description
copyBO	Returns a copy of the passed business object.
serializeBO	Serializes the passed business object to the passed output stream. This method can be used to write a business object into a file or the console.
deserializeBO	Deserializes a business object from the passed input stream. This method can be used to read a business object from a file.
getConnectionProperties	Returns a Properties_parameter business object containing the passed user, password and source properties.
createPLMAuthException	Returns Authentication_exception business object.
createPLMException	Returns an PLM_exception business object.

Method	Description
createInvalidSessionIdException	Returns Invalid_session_id_exception business object.
getPLMServicesType	Converts the passed business object in the respective Java wrapper class.

### Reviewing the Java wrapper classes

1. Examine the Java wrapper classes.
  - a. Expand **PLMS2Util** → **omg.org.plm20.informational.model**. See Figure 7-11.
  - b. Double-click the **item.java** interface to open it in the Java editor. Review the code and compare it with the **Item** business object in the **PLMServices2.WithSessionIDLib** library.

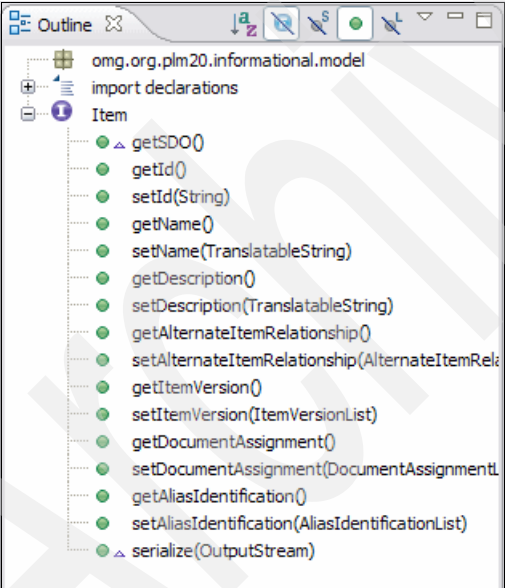


Figure 7-11 *Item.java outline*

- c. Expand **PLMS2Util** → **omg.org.plm20.informational.model.impl**.
    - d. Double-click **ItemImpl.java class** to open it in the Java editor.
    - e. Review also **ItemList.java** and **ItemListImpl.java**.
    - f. Review also **PLMContainer.java** and **PLMContainerImpl.java**. Review the code and compare it with the **PLM\_container** business object in the **PLMServices2.WithSessionIDLib** library.

- g. Expand **PLMS2Util** → **omg.org.plm20.util**.
- h. Double-click **PLMServicesUtil.java class** to open it in the Java editor.
- i. Review the class.

## 7.2.4 A simple PLM connection factory implementation

The last section reviewed the functionality that is required to implement the `PLM_connection_factory` interface. The `PLMServerConnectionFactory.java` class in the `com.ibm.pdf.framework.plmservices2.server` package contains methods that implement a simple PLM connection factory. It uses a Java properties file as user registry. See Figure 7-12. The constructor of this class takes the resource bundle name of a properties file and the endpoints of the `PLM_general_connection` and `PLM_message_connection` Web services as arguments.

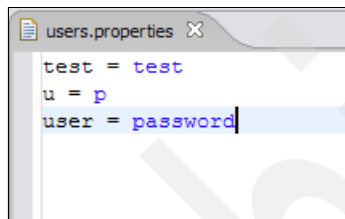


Figure 7-12 A Java properties file acts as user repository

The `get_general_connection` and `get_message_connection` create a `sessionId` and returns it with the respective Web service endpoint. Both methods take `user`, `password`, and an optional `source` property as arguments.

**The optional source property:** Chapter 8, “A framework to develop and run flexible PLM Services 2.0-based solutions” on page 221 provides more details on the optional `source` property.

The `sessionId` is stored together with those properties in a hash table so the information is accessible in subsequent calls to `PLM_general_connection` and `PLM_message_connection` operations.

The `closePLMSession` removes this information, so that further calls using the now invalid `sessionId` results in an `InvalidSessionIdException`. The `PLMServerConnectionFactory.java` class also contains a method to check if a `sessionId` is valid.

**Extending the implementation:** In this simple implementation, sessions do not expire after a certain period of time. It is up to the reader to extend the implementation. The same applies to the simple user registry. For example, a possible extension is to use the WebSphere user registry.

In order to review the implementation, perform the following steps:

1. Switch to the Java perspective.
2. Expand the PLMS2Util project.
3. Double-click **PLMServerConnectionFactory.java** under the `com.ibm.pdf.framework.plmservices2.server` package (Figure 7-13) to open it in the Java editor.
4. Review the Java source code.

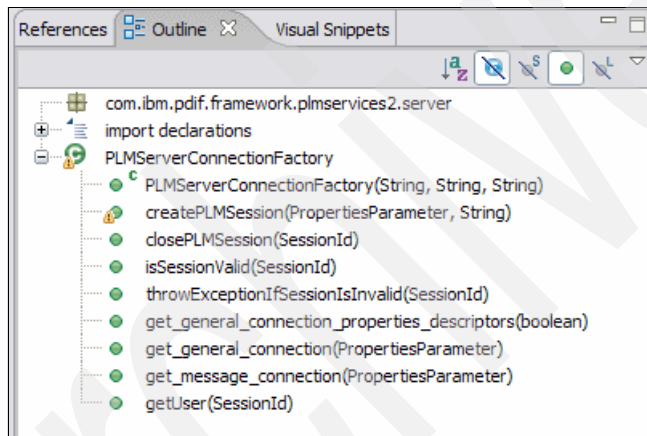


Figure 7-13 The *PLMServerConnectionFactory* class

## 7.2.5 Utility classes to implement a PLM Services 2.0 server

The scenarios described in Part III used simulated PDM systems that were used to retrieve data. To provide you the opportunity to reproduce these simulations quickly, we used an XML file with data similar to that of a PDM system. The Trego.xml data set was developed by ProStep iViP and can be downloaded from their Web page. It contains several items with related documents and relationships. The item names in the data set are in German. Table 7-2 on page 195 provides an English translation. Figure 7-14 on page 195 captures the hierarchy of the items used. In case you want to include real PDM systems instead of the one simulated, see 7.4, “Implementing PLM Services 2.0



connectors” on page 217, which outlines the necessary steps to provide a PLM Services 2.0 compliant wrapper for existing PDM systems.

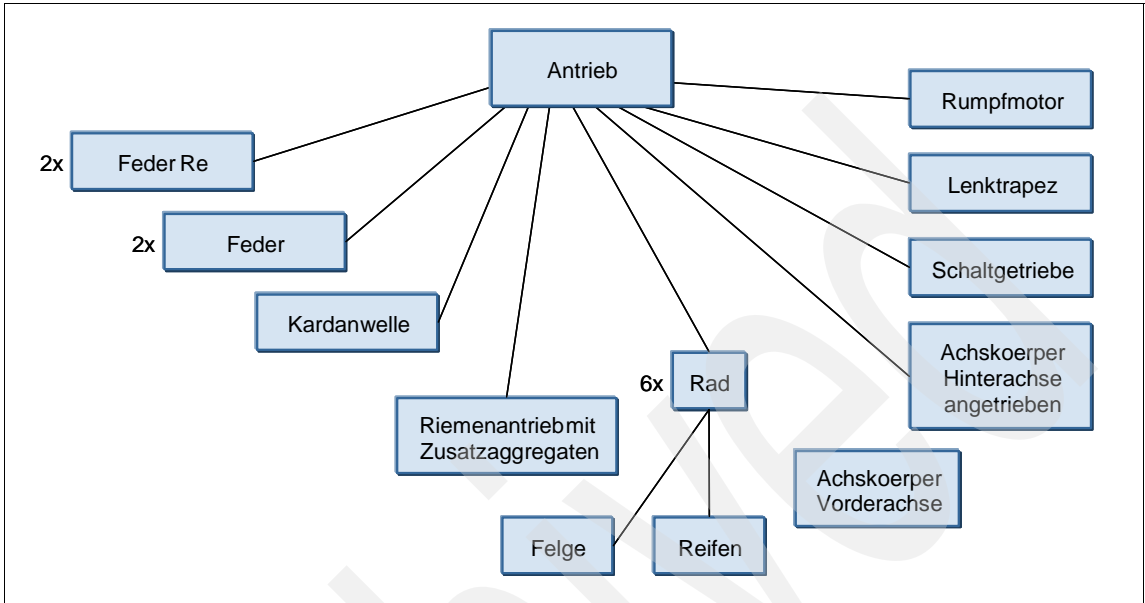


Figure 7-14 Hierarchy of items

Table 7-2 Items used in Trego.xml

Item name (German)	Item name (English)
Antrieb	power train
Feder	spring
Kardanwelle	cardan shaft
Lenktrapez	steering arm
Rumpfmotor	stripped engine
Riemenantrieb mit Zusatzaggregaten	belt transmission
Schaltgetriebe	control gear
Achskoerper Hinterachse angetrieben	axle casing, rear axle driven
Feder Re	spring
Achskoerper Vorderachse	axle casing, front axle
Rad®	wheel

Item name (German)	Item name (English)
Felge	rim
Reifen	tire

In order to demonstrate more complex scenarios in the following chapters, the data set was divided into two parts. The simulated PLMS2Server PDM system contains the following items:

- ▶ Antrieb
- ▶ Feder
- ▶ Kardanwelle
- ▶ Lenktrapez
- ▶ Rumpfmotor
- ▶ Riemenantrieb mit Zusatzaggregaten
- ▶ Schaltgetriebe
- ▶ Achskoerper Hinterachse angetrieben
- ▶ Feder Re, Achskoerper Vorderachse

The simulated PLMS2Server2 PDM system contains the following items:

- ▶ Antrieb
- ▶ Rad
- ▶ Felge
- ▶ Reifen

## Reviewing the code

As both simulated PDM system are implemented in the same hardware system, the common code base was packaged in the PLMS2Util package and is described in this section.

1. Examine the `com.ibm.pdif.framework.plmservices2.server` package under PLMS2Util project.
  - a. In the Package Explorer, expand **PLMS2Util project** → **com.ibm.pdif.framework.plmservices2.server package**.
  - b. Double-click **PLMServer.java** to open it in the Java editor.
  - c. Locate the `PLMServer(String configRB)` constructor (Example 7-4 on page 197) and review the implementation. The constructor takes the name of a resource bundle that must define values for the properties listed in Table 7-3 on page 197.

Table 7-3 Configuration properties

Configuration property	Description
xmlFile	Absolute filename of the used data set (PLMContainer instance).
messageConnectionUrl	The Web service endpoint that is returned by the PLM connection factory on a get_message_connection request.
generalConnectionUrl	The Web service endpoint that is returned by the PLM connection factory on a get_general_connection request.
usersResourceBundle	Name of the resource bundle that is used as user registry.

Example 7-4 PLMServer(String) constructor

```

public PLMServer(String configRB) {

    try {
        ResourceBundle config =
            ResourceBundle.getBundle(configRB,
                Locale.getDefault(), this.getClass().getClassLoader());
        String xmlFile = (String)config.getObject("xmlFile");
        DataObject plmc = PLMServicesUtil.deserializeBO(new
            FileInputStream(new File(xmlFile)));
        plmcontainer = new PLMContainerImpl(plmc);
        String gcUrl =
            (String)config.getObject("generalConnectionUrl");
        String mcUrl =
            (String)config.getObject("messageConnectionUrl");
        String usersResourceBundle =
            (String)config.getObject("usersResourceBundle");
        connFactory = new
            PLMServerConnectionFactory(usersResourceBundle,
                gcUrl, mcUrl);
        queryHelper = new PLMServerQuery();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

The constructor deserializes the specified xmlFile that must contain a serialized PLM\_container representing the data of a simulated PDM system. The constructor also creates a PLMConnectionFactory with the respective parameters and a PLMServerQuery instance that handles query requests.

2. Examine the PLMServerQuery.java class implementation.
  - a. Expand **com.ibm.pdf.framework.plmservices2.server package**→**PLMServerQuery.java** to open it in the Java editor.

This class implements a public query method (Example 7-5), and private methods, item\_selection\_query, item\_traversal\_query, and item\_detail\_query. This is the query helper class for the PLMServer.

*Example 7-5 query() method implementation*

---

```
public ContainerParameter query(QueryParameter parameter,
    PLMContainer pc) {

    PLMContainer container = null;
    try {
        PLMQuery plmq = parameter.getQuery();
        System.out.println("parameter.getQuery():"+plmq);
        if (plmq instanceof ItemSelectionQuery)
            container = item_selection_query(
                (ItemSelectionQuery)plmq, pc);
        else if (plmq instanceof ItemTraversalQuery)
            container = item_traversal_query(
                (ItemTraversalQuery)plmq, pc);
        else if (plmq instanceof ItemDetailQuery)
            container = item_detail_query((ItemDetailQuery)plmq,pc);
    } catch (Exception e) {
        e.printStackTrace();
    }

    ContainerParameter result = new ContainerParameterImpl();
    result.setQueryReturn(container);
    return result;
}
```

---

Depending on the type of query that comes in from a PLM Services client, it invokes one of the three private methods:

- item\_selection\_query
- item\_detail\_query
- item\_traversal\_query

3. Review the `item_selection_query()`, `item_detail_query()` and `item_structure_query()` methods.

The implementation iterates over the items in the sample data set (Trego.xml) and searches for matching items. The best way to understand the code is to look at the Trego.xml data (Figure 7-15) in parallel.

**Note:** There is no error handling in the code. The code also makes some assumptions. For example, that there is only one version of the item, and so forth. The intent of the samples is not to give you a solution, but to give you hints and a quick start on how to build PLM Services 2.0 solutions on WebSphere Process Server.



```
<Item uid="id_10019_786">
  <Id>A5000100010</Id>
  <Name>Reifen</Name>
  <Item_version uid="id_10022_812">
    <Id>0001,1</Id>
    <Design_discipline_item_definition uid="id_10005_416"
      xsi:type="Design_discipline_item_definition">
      <Id>pv00000002</Id>
      <Initial_context>id_10035_54</Initial_context>
      <Item_instance uid="id_10_901"
        xsi:type="Single_instance">
        <Id>ohbisUb000---d45d_45lapg</Id>
      </Item_instance>
      <Document_assignment uid="id_20_597">
        <Assigned_document>id_10014_577</Assigned_document>
        <Role>mandatory</Role>
      </Document_assignment>
    </Design_discipline_item_definition>
  </Item_version>
</Item>
<Item uid="id_10019_793">
  <Id>A4000102101</Id>
  <Name>Rad</Name>
  <Item_version uid="id_10022_819">
    <Id>0001,1</Id>
    <Design_discipline_item_definition uid="id_10002_57"
      xsi:type="Assembly_definition">
      <Id>pv00000009</Id>
      <Initial_context>id_10035_54</Initial_context>
      <Item_instance uid="id_10_895"
        xsi:type="Single_instance">
        <Id>ohbiKNh000---d45d_45laph</Id>
      </Item_instance>
    </Design_discipline_item_definition>
  </Item_version>
</Item>
```

Figure 7-15 Trego.xml

## 7.3 Implementing a PLM Services 2.0 server

The PLM Services 2.0 computational viewpoint shown in Figure 7-16 defines five specific components. The `PLM_connection_factory` allows a PLM Services client to establish a connection to a PLM Services server. The `PLM_general_connection` interface exposes functionality in a synchronous communication style whereas the `PLM_message_connection` interface can be used to communicate with the server in a message exchange style.

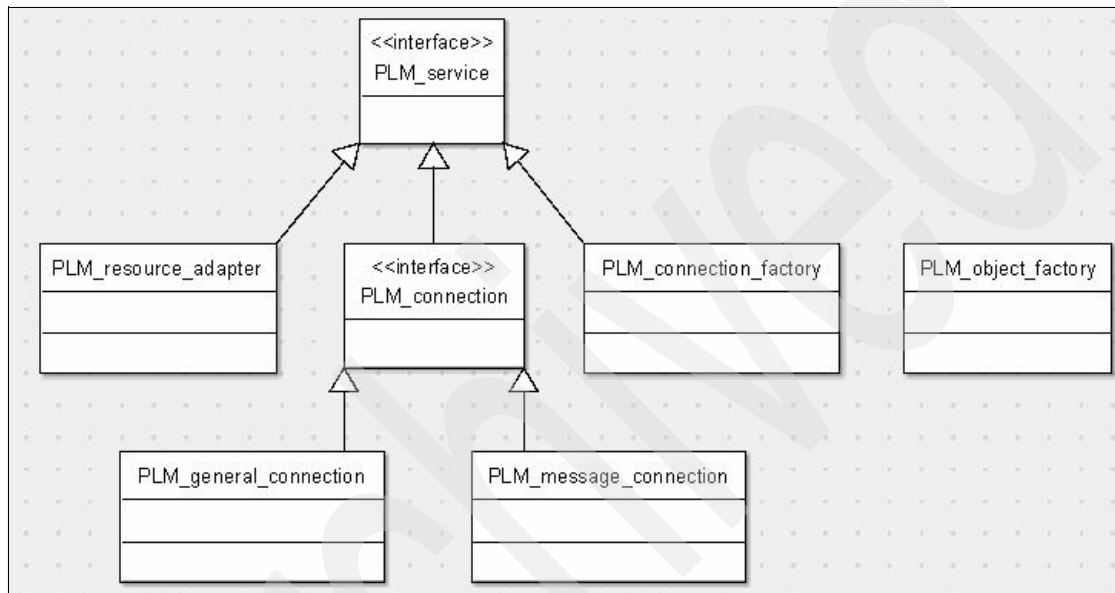


Figure 7-16 Class diagram of the PLM Services 2.0 computational view

In this section, you create a PLM Services 2.0 compliant server by creating a SCA module that implements the following interfaces:

- ▶ `PLM_connection_factory`
- ▶ `PLM_general_connection`
- ▶ `PLM_message_connection`

The implementations of `PLM_connection_factory`, `PLM_general_connection`, and `PLM_message_connection` is exposed as Web services using the SOAP/HTTP protocol. This is achieved by simply generating SCA exports with a Web services binding (SOAP/HTTP) for each of the components.

**Note:** This section shows how a PLM Services 2.0 server could be implemented. In this Redbooks publication, only the query operation is populated with business logic. The other PLM Services 2.0 operations are not implemented. It is up to the reader to extend the implementation.

First, create a new SCA module and add the PLMServices2WithSessionIDLib library and PLMS2Util Java project as dependencies:

1. Create a new module named PLMS2Server.
  - a. Select **File** → **New** → **Project....**
  - b. Select **Module**, then click **Next >**.
  - c. Specify PLMS2Server as module name and click **Next >**.
  - d. Select PLMServices2WithSessionIDLib as required library and click **Finish**.
2. Add the PLMS2Util project as dependent Java project.
  - a. In the Business Integration perspective, double-click **Dependencies** in the PLMS2Server module (Figure 7-17) to open the Dependency editor.

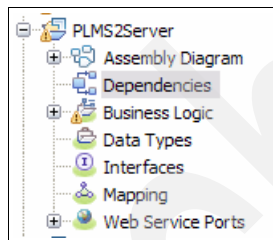


Figure 7-17 Double-click Dependencies to open the Dependency editor

- b. Expand the **Java** section (Figure 7-18) and click **Add...**
  - c. Select the PLMS2Util project and click **OK**.

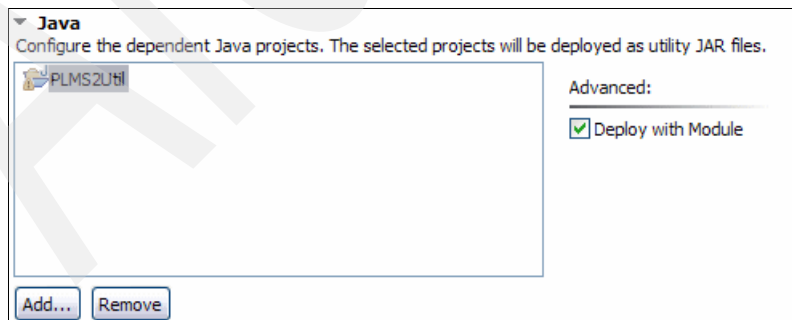


Figure 7-18 Add the PLMS2Util project

- d. Save the changes and close the Dependency editor.

After creating the module, we are now importing and reviewing some artifacts:

1. Import `users.properties`, `config.properties`, `Trego.xml` and `PLMerverInstance.java` into the `PLM2Server` module.
  - a. Right-click `PLM2Server` and select **Import...**
  - b. Select **General** → **File System** in the Select window of Import dialog box, then click **Next >**.
  - c. Go to the Additional Materials folder for Redbook SG24-7593, and download the lab file `Redbook247593_lab.zip`. Extract the files for use. See Appendix A, “Additional material” on page 465.
  - d. Click **Browse...** and select the `PLMS2Server` folder in the directory where you extracted the downloaded `Redbook247593_lab.zip` lab file.
  - e. Check the box beside the `PLM2Server` folder to import `users.properties`, `config.properties`, `Trego.xml` and `PLMerverInstance.java`.
  - f. Make sure the into folder shows `PLM2Server` and click **Finish**.
2. Examine the `com.ibm.pdif.framework.plmservices2` package under `PLMS2Server` project.
  - a. In the Package Explorer, expand **`com.ibm.pdif.framework.plmservices2` package** under `PLMS2Server` project.
  - b. Double-click **`PLMServerInstance.java`** (Example 7-6) to open it in the Java editor. The `getInstance()` returns a `PLMServer` instance. If necessary the instance is created first, specifying the resource bundle name of the configuration. The `PLMServer` class is reviewed later on.

*Example 7-6 PLMServerInstance.java*

---

```
package com.ibm.pdif.framework.plmservices2;

import com.ibm.pdif.framework.plmservices2.server.PLMServer;

public class PLMServerInstance {

    private static PLMServer instance = null;

    public static PLMServer getInstance() {
        if (instance==null) {
            instance = new
PLMServer("com.ibm.pdif.framework.plmservices2.config");
        }
        return instance;
    }
}
```

---



3. The config.properties contains the configuration for the PLM server instance. Modify the xmlFile parameter value in the config.properties (Table 7-4) based on your current workspace path.

Table 7-4 Configuration properties

Configuration property	Description
xmlFile	Absolute filename of the used data set (PLMContainer instance).
messageConnectionUrl	The Web service endpoint that is returned by the PLM connection factory on a get_message_connection request.
generalConnectionUrl	The Web service endpoint that is returned by the PLM connection factory on a get_general_connection request.
usersResourceBundle	Name of the resource bundle that is used as user registry.

- a. Expand **PLMS2Server** → **com.ibm.pdf.framework.plmservices2.server** package, double-click **config.properties** file (Figure 7-19) to open it in a Text editor.
- b. Modify the xmlFile property and replace D:\\workspaces61\\pdf2 with your current workspace path.

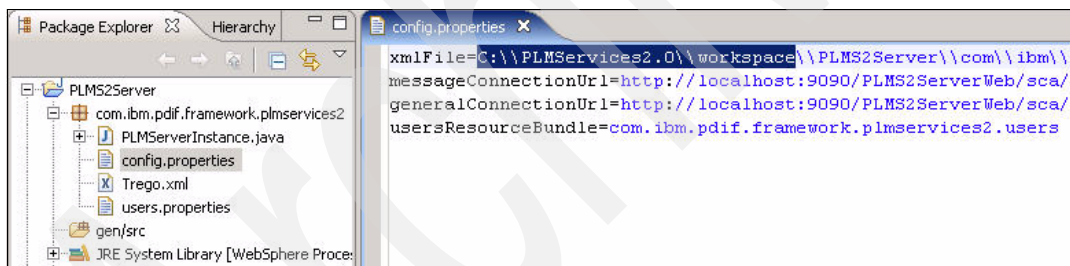


Figure 7-19 config.properties file

- c. Save and close the config.properties file.
4. Review the Trego.xml data set by expanding **PLMS2Server** → **com.ibm.pdf.framework.plmservices2.server** package and double-click the **Trego.xml** file to open it.

The following sections describe how to create and implement the PLMConnectionFactory, PLMGeneralConnection and PLMMesageConnection components and expose them as webservices. First,you will implement the PLM\_connection\_factory interface allowing a PLM services client to establish a connection to the server, as shown in 7.3.1, “Implementing PLMConnectionFactory service component” on page 204. Next, the

PLM\_general\_connection interface is implemented to support synchronous communication between the client and the server. See 7.3.2, “Implementing PLMGeneralConnection service component” on page 207. Finally, the PLM\_message\_connection interface is implemented to support communication in a message exchange style. See 7.3.3, “Implementing PLMMessageConnection service component” on page 208.

### 7.3.1 Implementing PLMConnectionFactory service component

In this section, you are going to implement the PLM\_connection\_factory service.

1. In the PLMS2Server assembly editor, create a service component, PLMConnectionFactory with PLM\_connection\_factory interface with Java implementation.
  - a. Switch back to the Business Integration view, expand **PLMServices2WithSessionIDLib → Interfaces**.
  - b. Locate PLM\_connection\_factory interface. Drag and drop it onto PLMS2Server assembly editor.
  - c. Select Component with no Implementation Type in the Component Creation dialog box, and click **OK**.
  - d. Change the name of the newly created component to PLMConnectionFactory, and save the changes.
  - e. Right-click PLMConnectionFactory, and select **Generate Implementation... → Java**.
  - f. At the Select the package where the Java implementation will be generated window, select **com.ibm.pdif.framework.pmlservices2**, and click **OK**. The PLMConnectionFactoryImpl.java must open in the Java editor.
  - g. Implement the get\_general\_connection(), get\_message\_connection(), get\_general\_connection\_properties\_descriptors() and get\_message\_connection()\_properties\_descriptors methods, by using the code in Example 7-7 on page 205. Basically the code reuses the PLMConnectionFactory implementation described in 7.2.4, “A simple PLM connection factory implementation” on page 193.

```
public DataObject get_general_connection(DataObject
aPropertiesParameter) {
    SessionParameter sp =
PLMServerInstance.getInstance().getConnectionFactory().get_general_conn
ection(new PropertiesParameterImpl(aPropertiesParameter));
    if (sp!=null) {
        return sp.getSDO();
    }
    return null;
}

public DataObject get_general_connection_properties_descriptors(
    DataObject aVoidParameter) {
    return
PLMServerInstance.getInstance().getConnectionFactory().get_general_conn
ection_properties_descriptors(false).getSDO();
}

public DataObject get_message_connection(DataObject
aPropertiesParameter) {
    SessionParameter sp =
PLMServerInstance.getInstance().getConnectionFactory().get_message_conn
ection(new PropertiesParameterImpl(aPropertiesParameter));
    if (sp!=null) {
        return sp.getSDO();
    }
    return null;
}

public DataObject get_message_connection_properties_descriptors(
    DataObject aVoidParameter) {
    // properties are the same as for general_connection
    return
get_general_connection_properties_descriptors(aVoidParameter);
}
```

---

- h. When errors appear, right-click anywhere inside the Java editor, and select **Source** → **Organize Imports** (Figure 7-20). Save the file, and the errors must be cleared.

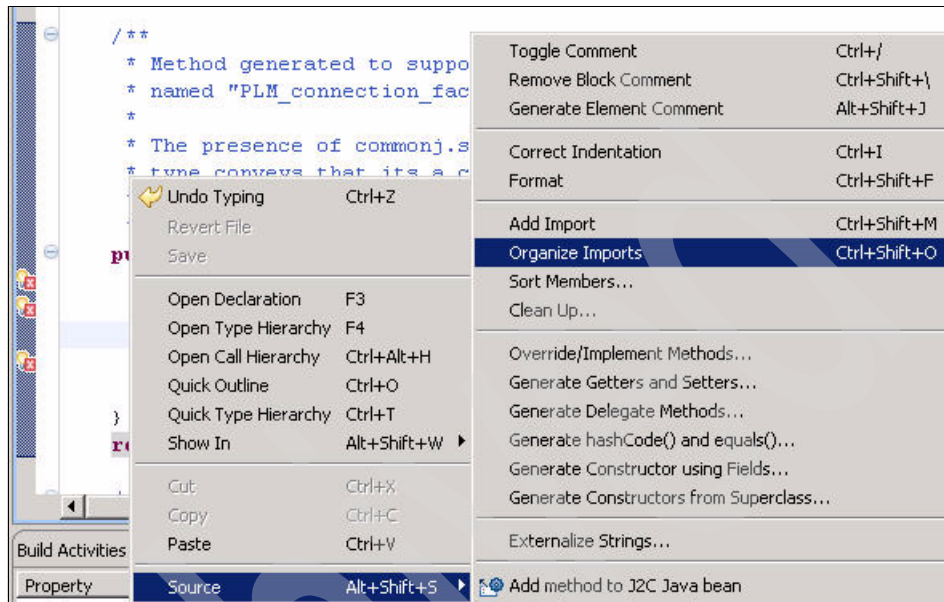


Figure 7-20 Organize imports

2. Expose the **PLMConnectionFactory** component as a Web service by generate an export component with Web services binding.
  - a. Right-click the **PLMConnectionFactory** component, and select **Generate Export...** → **Web Services Binding**.

Notice that there are several binding types supported by the WebSphere Process Server (Figure 7-21). The services provided by the **PLMConnectionFactory** component can be made available through Web service binding, HTTP binding, or Message bindings or SCA binding.

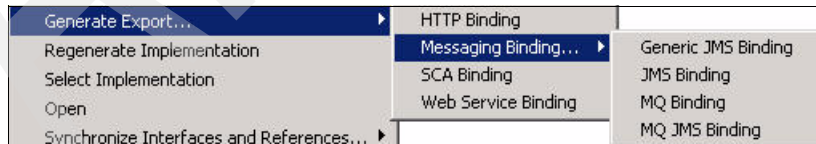


Figure 7-21 SCA export binding options

- b. Select **soap/http** from the Transport Selection dialog box, and click **OK**. This generates **PLM\_connection\_factoryExport1**.

- c. Right-click the **PLM\_connection\_factoryExport1** export component, and select **Refactor** → **Rename**. When the Save All Modified Resources dialog box appears, click **OK**.
- d. In the Rename Artifact dialog box, enter `PLM_connection_factory` in the New name field as shown in the Figure 7-22, then click **OK**.

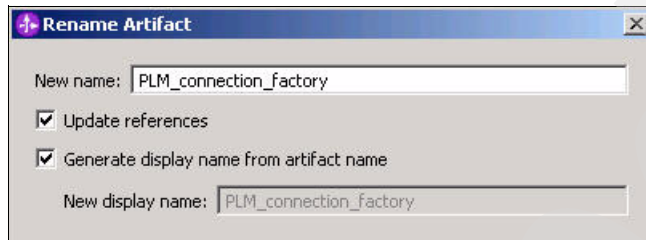


Figure 7-22 Rename `PLM_connection_factoryExport1` component

**Note:** If an error appears in the Problems view, select **Project** → **Clean...** to recompile the module or library codes.

### 7.3.2 Implementing **PLMGeneralConnection** service component

In this section, you are going to implement the `PLM_general_connection` service.

1. Create a service component, **PLMGeneralConnection** service component with `PLM_general_connection` interface with Java implementation.
  - a. In the Business Integration view, select **PLMServices2WithSessionIDLib** → **Interfaces** → **PLM\_general\_connection**.
  - b. Drag and drop **PLM\_general\_connection** interface onto PLMS2Server assembly editor as **Component with no Implementation Type**. Change the name of the newly created component to **PLMGeneralConnection**, and save the changes.
  - c. Right-click **PLMGeneralConnection**, and select **Generate Implementation...** → **Java**.
  - d. At the Select the package where the Java implementation will be generated window, select **com.ibm.pdf.framework.plmservices2**. Click **OK**. The `PLMGeneralConnectionImpl.java` must open in the Java editor.
  - e. Complete the implementation of `query()` method, as listed in Example 7-8 on page 208. The implementation resuses the PLM server functionality as described in XR.

```
public DataObject query(DataObject parameters, DataObject
    sessionId){
    return PLMServerInstance.getInstance().query(((QueryParameter) new
    QueryParameterImpl(parameters))).getSDO();
}
```

**Note:** The scenario demonstrated in this chapter covers only the query functionality. Therefore, the other method stubs can be left unchanged. It is up to the reader to extend the scenario and implement the other stubs.

- f. When errors appear, right-click anywhere inside the Java editor, and select **Source** → **Organize Imports**. Select `omg.org.plm20.services.parameter.QueryParameter` from the Choose type to import window, and click **Finish**.
- g. Save and close the Java editor. There must be no more errors.
2. Expose the `PLMGeneralConnection` component as a Web service by generating an export component with Web services binding.
  - a. Right-click **PLMGeneralConnection** component, and select **Generate Export...** → **Web Services Binding**.
  - b. Select `soap/http` from the Transport Selection dialog box, and click **OK**. This generates `PLM_general_connectionExport1`.
  - c. Right-click **PLM\_general\_connectionExport1** export component, and select **Refactor** → **Rename**. When the Save All Modified Resources, window appears, click **OK**.
  - d. In the Rename Artifact dialog box, enter `PLM_general_connection` in the New name field, then click **OK**.

### 7.3.3 Implementing `PLMMessageConnection` service component

**Note:** Implementing and exposing the `PLMMessageConnection` is optional, because the scenario in this chapter does not use the `PLM_message_connection` interface. It is described for your reference. It is up to the reader to extend the scenario and implement the stubs.

In order to support message exchange style communication between the client and the server, you have to implement the `PLM_message_connection`.

1. Create a `PLMMessageConnection` service component with `PLM_message_connection` interface with Java implementation.
  - a. In the Business Integration view, select **PLMServices2WithSessionIDLib** → **Interfaces** → **PLM\_message\_connection**.
  - b. Drag and drop the **PLM\_message\_connection** interface onto PLMS2Server assembly editor as Component with no Implementation Type. Change the name of the newly created component to `PLMMessageConnection`, and save the changes.
2. Right-click the **PLMMessageConnection** component and select **Generate Implementation...** → **Java** to generate Java implementation in the `com.ibm.pdf.framework.plmservices2` package.
3. Expose the `PLMMessageConnection` component as a Web service by generating an export component with Web services binding.
  - a. Right-click **PLMMessageConnection component**, and select **Generate Export...** → **Web Services Binding**.
  - b. Select soap/http from the Transport Selection dialog box, and click **OK**. This generates `PLM_message_connectionExport1`.
  - c. Right-click **PLM\_message\_connectionExport1 export component**, and select **Refactor** → **Rename**. When the Save All Modified Resources window appears, click **OK**.
  - d. In the Rename Artifact dialog box, enter `PLM_message_connection` in the New name field, then click **OK**. Your final PLMS2Server assembly diagram must look like Figure 7-23.

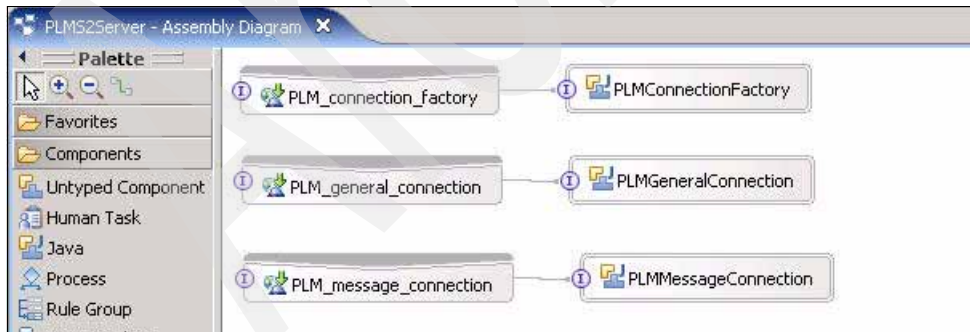


Figure 7-23 PLMS2Server assembly diagram

- e. Save and close the assembly diagram.

## Modifying the deployment descriptor

Edit the PLMS2Server deployment descriptor to add the JAX-RPC handler to intercept the requests to PLM\_general\_connection and PLM\_message\_connection. The JAX-RPC handler copies the sessionId in the SOAP header into the sessionId input parameter of the called PLM\_general\_connection / PLM\_message\_connection operation as described in “Modifications to ease processing of the sessionId” on page 184.

1. In the Business Integration view, right-click **PLMS2Server** and select **Open Deployment Editor**. This opens the deployment description in the Module Deployment Editor.
2. Select Exports tab. The Export Specific Settings section must be expanded.
3. Add ServiceExportHandler to PLM\_general\_connection Web Service Export.
  - a. Select **PLM\_general\_connection** from the Web Service Exports menu, and expand **JAX-RPC Handlers** node.
  - b. Click **Add...**
  - c. In the Web Service Handler window, click **Browse...** to select `com.ibm.pdf.framework.jaxrpc.handler.ServiceExportHandler` and click **OK**.
  - d. Enter `PDIFServiceExportHandler` in the Display Name, Description, and Handler Name fields. Click **OK**.
4. Modify the URL-Mapping for PLM\_general\_connection Web Service Export.
  - a. Select **PLM\_general\_connection** from the Web Service Exports window, and expand the URL Mappings node.
  - b. Click **Add...** and specify `/sca/PLM_general_connection` as URL pattern. Click **OK**.
  - c. Clear the Include default mapping check box.



5. Repeat the step to add ServiceExportHandler to PLM\_message\_connection Web Service Export as shown in Figure 7-24.

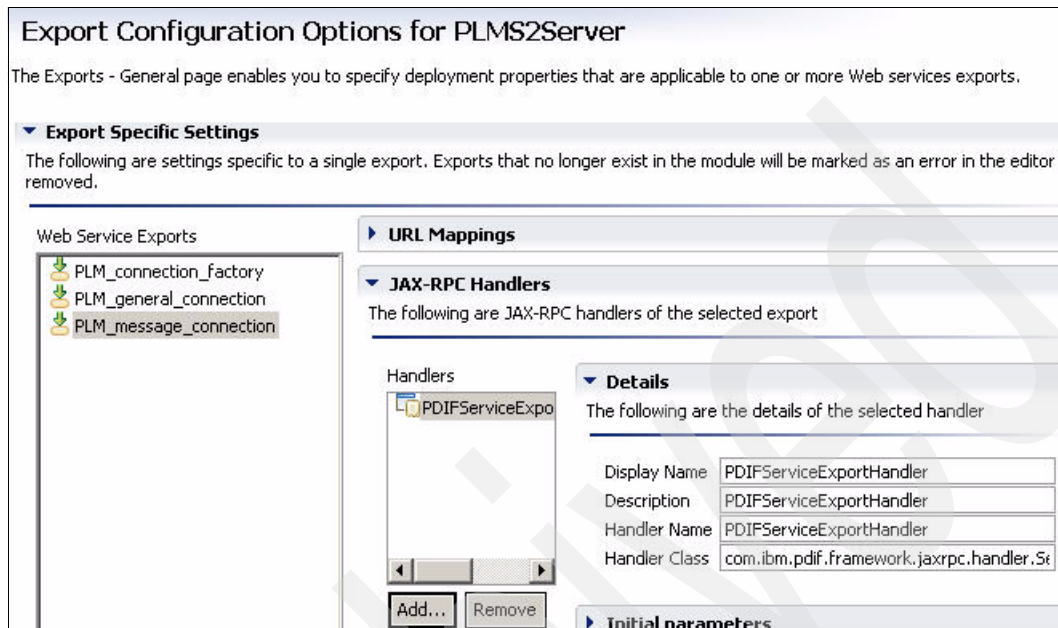


Figure 7-24 PLMS2Server deployment descriptor

6. Modify the URL-Mapping for PLM\_message\_connection Web Service Export.
  - a. Select PLM\_message\_connection from the Web Service Exports window, and expand the URL Mappings node.
  - b. Click **Add...** and specify /sca/PLM\_message\_connection as URL pattern. Click **OK**.
  - c. Clear the Include default mapping check box.
7. Modify the URL-Mapping for PLM\_connection\_factory Web Service Export.
  - a. Select PLM\_connection\_factory from the Web Service Exports window, and expand the URL Mappings node.
  - b. Click **Add...** and specify /sca/PLM\_connection\_factory as URL pattern. Click **OK**.
  - c. Clear the Include default mapping check box.
8. Save and close the deployment descriptor.

**Note:** When Inconsistent Files warning dialog box appears, click **Yes** to continue.

### 7.3.4 Running the PLM Services 2.0 server

In this section, we use the ProSTEP iViP PLM Services 2.0 reference client to test the server implementation running on WebSphere Process Server as depicted in Figure 7-25.

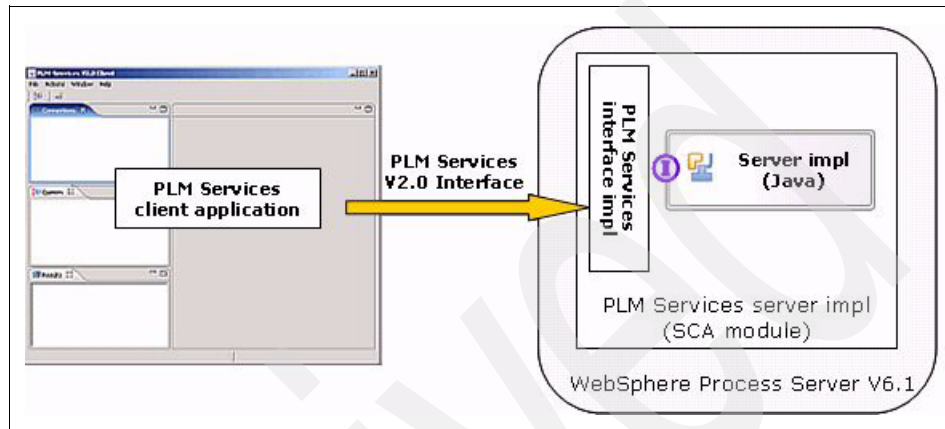


Figure 7-25 PLM services reference client accessing the PLM Services 2.0 server on WPS

The PLM Services reference client directly accesses the PLM Services server implementation done in the previous section. The next chapter demonstrates a federated query over two PLM Services 2.0 server.

1. Start the test server from WebSphere Integration Developer, and deploy PLMS2Server SCA applications.
  - a. In the Servers view, right-click **WebSphere Process Server V6.1** and select **Start** in the Server view.
  - b. Wait for the server to complete the startup procedure. The server has finished starting when you see the “Server server1 ready for e-business in the console” message.
  - c. Right-click **WebSphere Process Server v6.1** and select **Add and remove projects...** from the context menu.
  - d. Select PLMS2ServerApp in the Available projects list, and click **Add >**.
  - e. Click **Finish**. Wait for the PLMS2Server application to start.
2. Start the TCP/IP monitor (Figure 7-26 on page 213) on port 9090.
  - a. Select **Windows® → Preferences**.
  - b. Double-click **TCP/IP Monitor** under the Run/Debug node to open its properties window, and click **Add**.

c. In the New Monitor window, enter the following values:

- Local monitoring port: 9090
- Host name: localhost
- Port: 9080

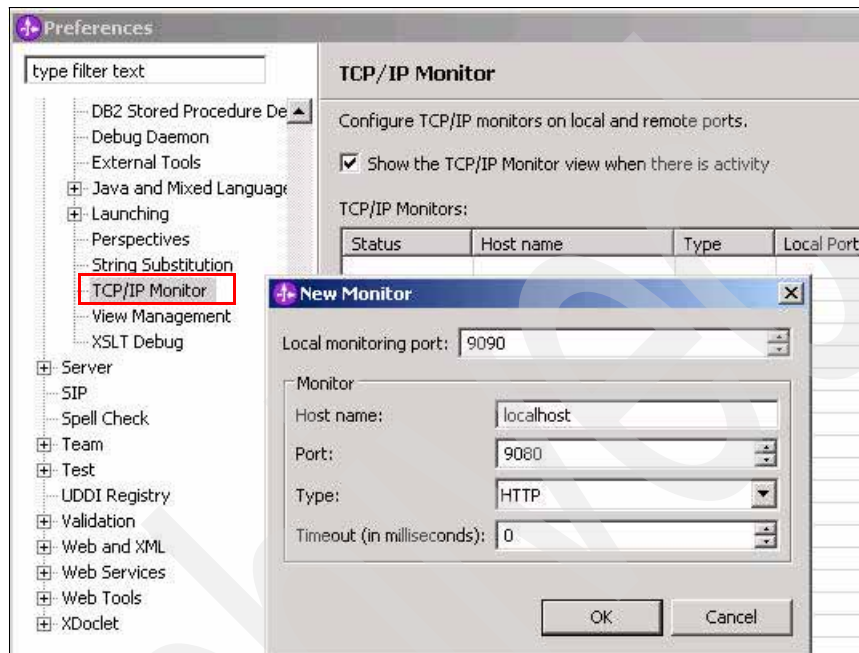


Figure 7-26 TCP/IP Monitor

- d. Click **OK** to close the window, then **Start** to start the monitor.
- e. Click **OK** to close the Preferences window.
3. Start the reference client, and create a new general connection to the PLM Services 2.0 server that you have deploy onto WebSphere Process Server environment.
- a. Double-click **PLM-Services V2.0 Client.exe** to start the ProSTEP iViP.
  - b. In the Connections view, right-click and select **New Connection...** from the context menu as shown in Figure 7-27.

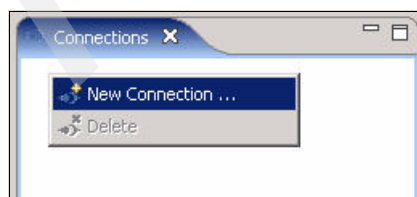


Figure 7-27 Create a new connection to PLM Services server

- c. In the New Connection Factory window, enter the following value in the Value field of Endpoint property:  
`http://localhost:9090/PLMS2ServerWeb/sca/PLM_connection_factory`

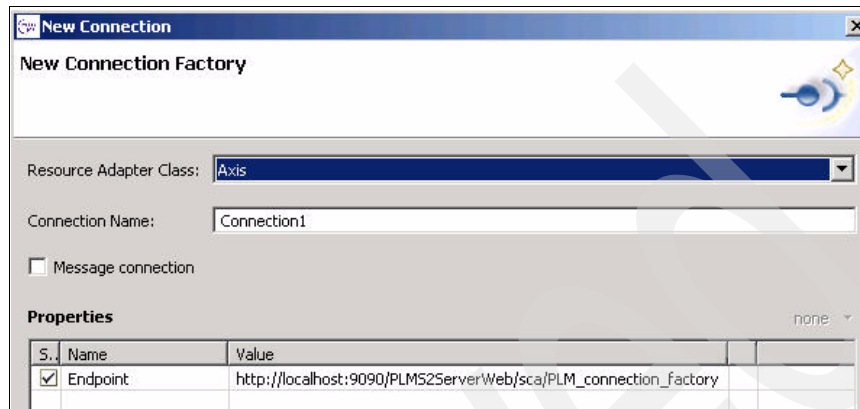


Figure 7-28 Create a new connection

The reference client connects to your PLM Services 2.0 server through the port 9090, which you have opened using the TCP/IP Monitor.

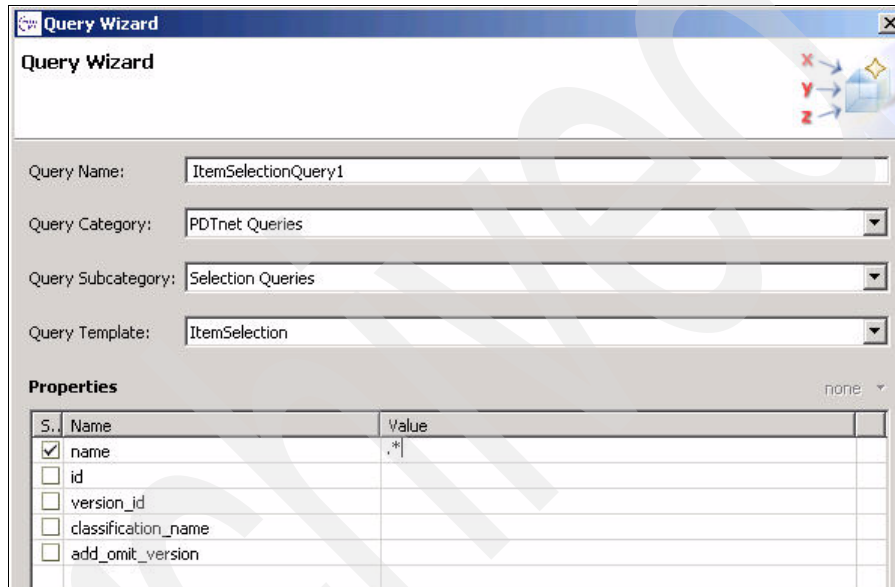
- d. Click **Connect** to make a connection. After successful connection to the server, you can click **Next >** to proceed.
- e. In the New Connection window, enter the value of test in the Value fields for both user and password properties.

**Note:** The connection properties are specific to each PLM Services 2.0 server implementation, and its descriptors are accessed through the `get_general_connection_properties_descriptor()` and `get_message_connection_property_descriptors()` methods defined in the implementation of `PLM_connection_factory`, which, in this example, is `PLMConnectionFactoryImpl.java`.

- f. Click **Finish**.
4. The PLM Services reference client supports PDTnet queries, specific queries, and utility queries. In the Redbooks publication only the PDTNet item queries are implemented. Selection queries are the starting point of data retrieval. Run PDTnet item selection query:
  - a. In the Queries view, right-click and select **New Query...** from the context menu.
  - b. From the drop down list, select **PDTnet Queries** for the **Query Category**.

- c. Select **Selection Queries** from the drop down list of **Query Subcategory**.
- d. Select **ItemSelection** from the drop down list of **Query Template**.
- e. Select the check box for **name** property, and enter **.\*** (wildcard) in the value field as shown in Figure 7-29.

**Note:** The server implementation supports Java regular expressions. **'.\*'** is the the Java regular expression for the usual **'\*'** wildcard.



**Query Wizard**

Query Name:

Query Category:

Query Subcategory:

Query Template:

**Properties** none ▾

S.	Name	Value
<input checked="" type="checkbox"/>	name	.*
<input type="checkbox"/>	id	
<input type="checkbox"/>	version_id	
<input type="checkbox"/>	classification_name	
<input type="checkbox"/>	add_omit_version	

Figure 7-29 Run a new query

- f. Click **Next >**. Verify that **Connection1** is selected, and click **Finish**.

**Note:** Refer to the *Usage of the Client for OMG PLM Services Version 2.0* document for detail.

g. The result is displayed as shown in Figure 7-30.

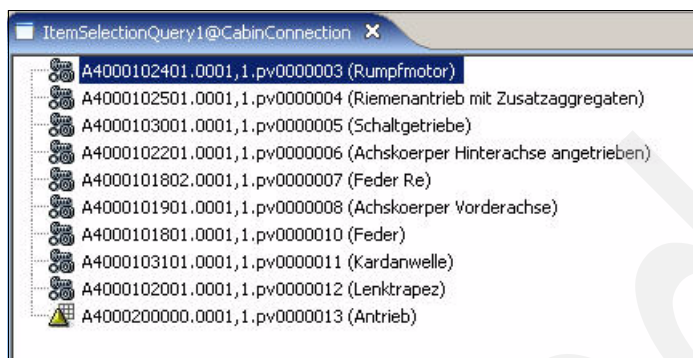


Figure 7-30 PDTnet ItemSelectionQuery result

5. Run detail query from the item selection query to retrieve the detail information of an item.
  - a. Select and right-click a desired product from the item selection query result you wish to see the detail.
  - b. Select **Launch Detail Query** from the context menu. It displays as shown in Figure 7-31.

Type	Name	Value
DesignDisciplineItemDefinition	UID	id_10005_417
DesignDisciplineItemDefinition	Id	pv0000003
ItemVersion	UID	id_10022_813
ItemVersion	Id	0001,1
Item	UID	id_10019_787
Item	Id	A4000102401
Item	Name	Rumpfmotor

Figure 7-31 ItemDetailQuery result

## 7.4 Implementing PLM Services 2.0 connectors

The sections so far covered the implementation of a PLM Services 2.0 compliant server. In many cases there are already existing PDM systems with proprietary data models and access protocols. This section discusses how these systems can be exposed as PLM Services 2.0 resources.

### 7.4.1 Exposing PDM systems as PLM Services 2.0 resources

As described in Part 2, “Scenarios and patterns” on page 63, using a common language like PLM Services 2.0 has several advantages. The business logic does not have to cope with different data formats and access methods when multiple PDM systems are involved. It makes it easier for the developer to focus on solving the domain problems instead of struggling with IT-related problems like different data formats and protocols.

Writing a PLM Services 2.0 connector/wrapper for existing PDM systems is quite similar to what was done in section 7.3, “Implementing a PLM Services 2.0 server” on page 200. A PLM Services 2.0 Web Service skeleton has to be created as in XR. Instead of populating the skeleton with code to access and process the Trego.xml data set, the connector code has to access the existing PDM system using the PDM system specific protocol. Usually those systems are accessible through Web services, JMS, JCA (Java Connector Architecture) connectors, RMI/IIOP, or Java APIs. Leveraging the SCA tooling makes it easy to access those systems. An appropriate SCA import has to be created and wired in the assembly diagram. Mapping the PLM Services 2.0 methods (like query, write, and so forth) to the appropriate PDM system specific methods is usually easy, as most PDM systems offer similar functionality (for example, to retrieve, create, update, or delete entities).

While connectivity to the PDM systems can usually be developed within minutes, mapping the PLM Services 2.0 data model to the PDM system-specific data model and vice versa requires much more effort. This is a domain-specific problem and depends on the specific data model of the existing PDM system and its similarity to the PLM Services 2.0 data model.

Defining the data mapping is the most complex part in designing and developing a PLM Services 2.0 connector and most effort has to be spent on that topic. In addition, it also requires a good understanding of both the PLM Services 2.0 and the PDM-specific data model.

After evaluating and specifying the data mapping rules and semantics, WebSphere Integration Developer provides several ways to implement this mapping, including graphical and programmatic approaches:

- ▶ Interface and data maps
- ▶ Mediations and XSLT transformations
- ▶ Java code

Depending on the preference and the complexity of the mapping, one or a mix of these approaches can be chosen.

## **7.4.2 Handling requests/events from the PDM system**

So far we discussed how to process outbound requests, that is, requests from a business component targeting a PDM system. The opposite flow represents a valid scenario as well. For example, if the status of entities in a PDM system changes from in-work to released some business logic must be triggered to synchronize the status in other involved systems.

Again, leveraging the SCA tooling provides an easy and fast way to build components to capture the requests from the PDM system. Instead of using SCA Imports, SCA Exports (using JMS, JCA, Web services, and so forth) are used to provide the PDM system means to notify requests to the connector. The connector could process these events, for example, by triggering appropriate business logic.

As for outbound requests, the data mapping between the PDM-specific data model and the PLM Services 2.0 one is the most complex part when developing the capability to handle requests/events from the PDM system.



### 7.4.3 PLM Services 2.0 connector example

As connectors are specific to the targeted PDM system, this section does not have a running example, but shows how the assembly diagram of a PLM Services 2.0 connector (Figure 7-32) connects to a PDM system that is accessible through JMS can look like.

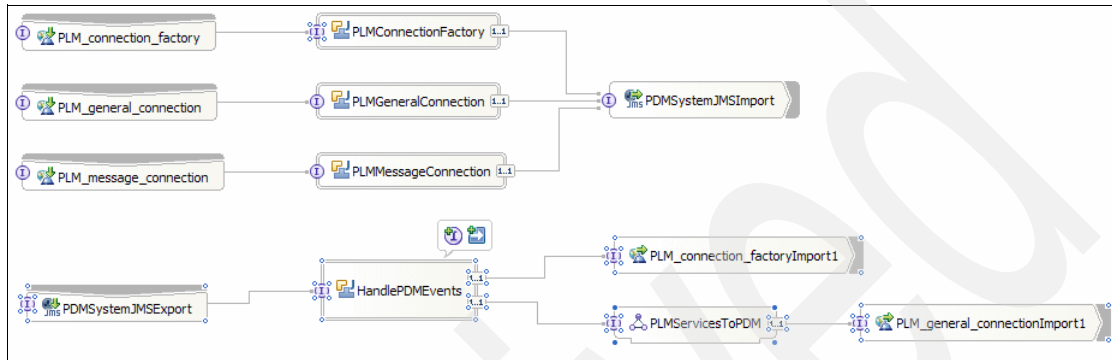


Figure 7-32 Assembly diagram of a sample PLM Services 2.0 connector

The upper part of Figure 7-32 shows the components to handle outbound requests, accessing the PDM system through a JMS Import. The data mapping is not shown, it is implemented in Java code in the Java components.

The bottom part of Figure 7-32 captures the components to handle events/requests received from the PDM system through a JMS Export. In the example the request is processed by calling a PLM Services 2.0 component (the framework described in Chapter 8, “A framework to develop and run flexible PLM Services 2.0-based solutions” on page 221) that, for example, triggers specific business processes to synchronize life cycle events with other involved systems. There is an interface map with an associated data map that transforms the PDM-specific data model into the PLM Services 2.0 data model.

Archived

# A framework to develop and run flexible PLM Services 2.0-based solutions

The previous chapter provided insights and utilities to build basic PLM Services 2.0-based solutions. This chapter focuses on flexibility and reusability. A framework is introduced that acts as intermediate between PLM Services clients and PDM/PLM systems and allows to execute business logic (hosted, for example, on WebSphere Process Server) upon client requests.

## 8.1 Motivation

The previous chapter provided an overview and hands-on exercises to develop PLM Services 2.0-based solutions in WebSphere Integration Developer. It focused on developing server-side components exposing a PLM Services 2.0-compliant interface. Another factor to consider is the development of clients that access PLM Services 2.0 servers and connectors. Implementing such client applications in WebSphere Integration Developer is straightforward when using SCA tooling. Chapter 9, “Design and implementation of an engineering change scenario including SAP integration” on page 255 contains a sample client application demonstrating the development of a PLM Services client.

Several aspects have to be considered to ensure that the emerging solutions is maintainable and reusable when implementing PLM Services 2.0 clients. Later in this chapter, we implement a component that provides a federated view of the two PDM systems in the Trucks Inc. business scenario. To achieve this, we implement a business process that queries the two PDM systems, merges the results, and returns the merged result. A simple straightforward implementation would hardwire the endpoints of the PDM systems. In that case, whenever those endpoints change, we would have to adjust the business logic. This situation is rendered worse if another department requires the same functionality but accesses different PDM systems. In that case, we have to duplicate the business process with exactly the same business logic but wiring different endpoints. Such solutions have limited potential for reuse and maintenance is hard to achieve.

In order to avoid such tightly coupled solutions and their disadvantages, the business logic has to be designed and implemented with aspects like reusability and maintainability in mind. One approach that is described in this chapter is the parameterization of business logic with meta-information about, for example, which endpoints to use. In the scenario described above, the federated query business process would take the endpoints of the two involved PDM systems as input parameters, and use them to invoke the respective PDM systems. The other department could reuse the same business process by just passing different PDM system endpoints.

Of course, this flexibility does not come without caveats. There is some additional effort required in the business logic to handle the passed meta-information. For example, instead of invoking hard-wired PDM systems, the passed endpoints have to be assigned to a partner reference before invoking those endpoints dynamically. In addition the business logic has to follow some conventions. For example, it must fulfill certain interfaces to allow passing of the meta-information.

In addition, the question arises how the meta-information (the endpoints in the example above) is passed to the business logic (the federated query business process). Hardcoding the information in the client application is not a viable solution either. The framework described in this chapter uses an XML configuration file that holds the meta-information. It provides a Web user interface that allows modification of this information at runtime. Another approach that stores the meta-information in WebSphere Service Registry and Repository is outlined in Chapter 10, “SOA realization of integrated PLM solutions: Working with examples” on page 403.

Compared to the simple solution, the approach to parameterize the business logic and store the actual parameters in an XML file or WSRR requires some additional investment in building the business logic. But it eases the development of reusable and maintainable business logic and pays off as the size of the business solution is growing over time.

## 8.2 Subscription-based routing

The infrastructure component (also referred to as SubscriptionManager in this chapter) described in this chapter acts as an intermediate between PLM Services 2.0 clients and PLM/PDM systems. Based on its configuration, and dependent on the request, the infrastructure component can execute specific business logic that, for example, accesses one or more PLM/PDM systems. As the component exposes a PLM Services 2.0 interface, it can be called by any PLM Services 2.0 client in the same way as a respective PLM Services 2.0 server would be called. Based on the incoming request the SubscriptionManager tries to find a matching subscription in its current configuration. In case nothing is found, an appropriate fault is raised. Otherwise, the configuration data is used to invoke the specified business logic passing the respective meta-information found in the configuration. This high-level view is captured in Figure 8-1.

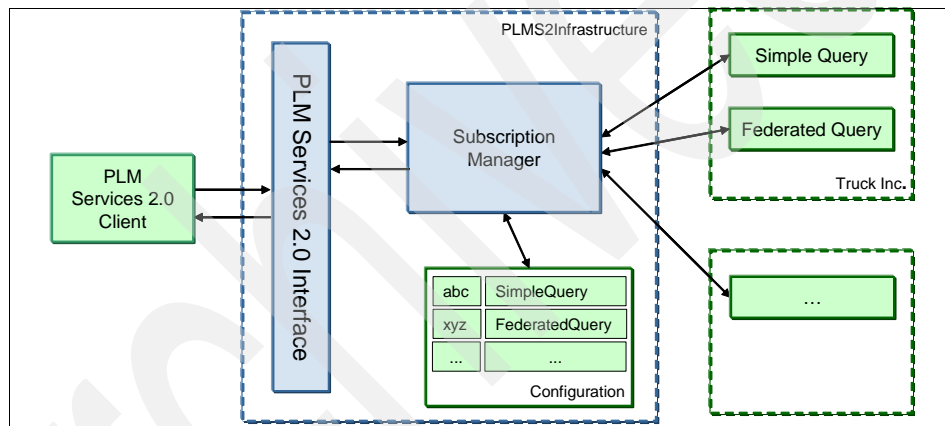


Figure 8-1 High-level view of the SubscriptionManager.

### 8.2.1 Matching subscriptions

The matching between the incoming request and the configuration is done based on meta-information of the request. The criteria are as follows:

- ▶ PLM Services 2.0 request operation (for example, query, write\_messages)
- ▶ User on which behalf the request is executed
- ▶ Source property identifying the request source

**Note:** Chapter 10, “SOA realization of integrated PLM solutions: Working with examples” on page 403 contains examples that trigger specific business logic based on the content of the request.

Both the user and source property are specified as properties in the `get_general_connection/get_message_connection` request to the `PLM_connection_factory` service of the infrastructure component. As the infrastructure component is designed to be used by multiple clients, a mechanism has to be found to distinguish between the request originators in order to allow the execution of different business logic for different clients. The implemented solution uses the mentioned source property as an additional property that has to be specified by a client in the `get_general_connection / get_message_connection` request to the `PLM_connection_factory` service.

## 8.2.2 Interface requirements for the business logic

As the incoming client request has specific input and return parameter types, the business logic that must be executed to fulfill the request has to match those parameter types. In addition, the `SubscriptionManager` must be able to pass meta-information (like endpoints of PDM systems or arbitrary properties) to the business logic. This requires the business logic that likes to use the framework to fulfill the respective interface based on the request operation listed in Table 8-1 and Table 8-2 on page 225.

Table 8-1 *PLM\_general\_connection operations and associated interfaces*

Operation	Associated interface
delete	PLMDeleteIF
write	PLMWriteIF
import_data	PLMImportDataIF
export_data	PLMExportDataIF
query	PLMQueryIF
get_download_url, get_upload_url	PLMGetUrlIF
get_property_descriptors	PLMGetPropertyDescriptorsIF
get_properties	PLMGetPropertiesIF
set_properties	PLMSetPropertiesIF

Figure 8-2 on page 225 shows the `PLMQueryIF`. It has the signature of the `PLM_general_connection` query operation, plus an additional input parameter of type `Action`. The `Action` parameter allows the `SubscriptionManager` to pass meta-information to the invoked business logic and is described in one of the next sections.

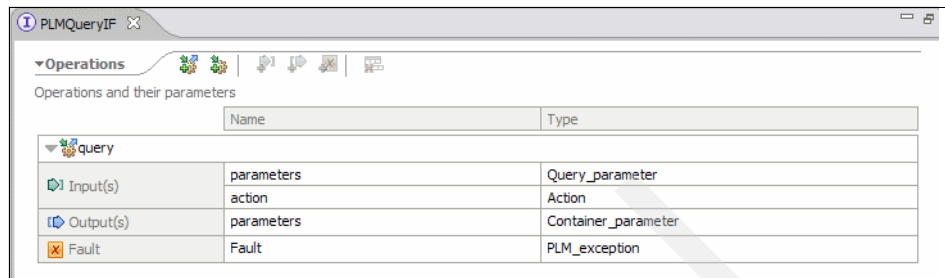


Figure 8-2 PLMQueryIF interface

Table 8-2 PLM\_message\_connection operations and associated interfaces

Operation	Associated interface
delete_messages	PLMDeleteMessagesIF
write_messages	PLMWriteMessagesIF
query_messages	PLMQueryMessagesIF
get_download_url, get_upload_url	PLMGetUrlIF
get_property_descriptors	PLMGetPropertyDescriptorsIF
get_properties	PLMGetPropertiesIF
set_properties	PLMSetPropertiesIF

### 8.2.3 User credential propagation

In many cases, the business logic that is called on behalf of an incoming request has to access one or multiple PDM/PLM systems in order to fulfill the request. As those systems usually require authentication, the question arises how the respective user credentials are provided. The SubscriptionManager supports the use of two simple approaches to pass user credentials:

- ▶ A user/password combination that was used to authenticate against the infrastructure component.
- ▶ A technical user

The propagation mechanism can be set on an endpoint level through the useDefaultUser flag. In case this flag is set to true, the user/password of a technical user specified in the configuration is passed when calling the chosen business logic. Otherwise, the user/password combination that was used by the requestor is passed to the business logic.

## 8.2.4 Configuration data model

The configuration that serves as the basis for the routing decisions of the SubscriptionManager consists of an array of subscriptions, each of type Subscription that is outlined in Table 8-3. The key attribute consists of the request operation and the user name and source property. The enabled flag allows to quickly activate/deactivate subscriptions. In case the flag is set to false, the SubscriptionManager behaves as though there is no such subscription. The action attribute specifies the business logic that must be performed in case a request is processed that matches the key properties of the subscription.

Table 8-3 Subscription attributes

Attribute	Description
key	The key consists of the type of request, a user name and the source property. The key properties are the criteria to find matching subscriptions.
enabled	A flag indicating whether the subscription is active.
description	An optional description of the subscription.
action	The action that must be executed in case the incoming request matches the key.

The type attribute of the Action data type specifies the interface that the business logic addressed by the URL attribute has to fulfill. It has to be one of the interfaces listed in Table 8-1 on page 224 and Table 8-2 on page 225.

The URL attribute specified the endpoint address of the business logic that must be executed. The URL can be a Web service endpoint (like <http://localhost:9090/mywebservice>) or an SCA export (like <sca://myModule/myExport>).

The useDefaultUser and defaultUser attributes are not used in the current implementation. But these attributes can be used in the future to execute the action on behalf of the specified user.

The endpoints attribute specifies an array of endpoints (of type Endpoint) that is passed to the business logic. In addition to endpoints, arbitrary properties (key/value pairs) can be specified in the configuration and is passed by the SubscriptionManager to the business logic. This represents a generic way to parameterize the business logic. For example, in the XR SAPSample, this capability is used to specify whether BOM data is passed to a real SAP system or just written to the console due to the absence of such a system. Table 8-4 on page 227 summarizes the Action attributes.



Table 8-4 Action attributes

Attribute	Description
type	The type attribute contains the name of the interface that the endpoint specified by the URL attribute fulfills.
url	The endpoint address of the business logic to be executed. The URL attribute can specify a Web service address or a SCA Export.
description	An optional description of the action.
useDefaultUser	A boolean flag telling the infrastructure component to use the defaultUser attribute (if true) or overwrite the defaultUser with the current user otherwise.
defaultUser	A user (userId/password combination) that must be used to execute the action. Note: in the current implementation this attribute is not used.
endpoints	An array of endpoints that must be passed to the business logic.
properties	An array of key/value pairs that must be passed to the business logic.

The Endpoint data type defines the properties of an endpoint. The id attribute is a string that can be used by the business logic to uniquely identify endpoints in the passed endpoint array in the Action instance that is passed to the business logic. An example is given in the XR SimpleQueryProcess. The URL attribute specifies the endpoint address that can either be a Web service or an SCA Export. The useDefaultUser and defaultUser are used to specify the user credential propagation mechanism. In case useDefaultUser is set to true, the SubscriptionManager passes the user specified in defaultUser. Otherwise, the SubscriptionManager overwrites the defaultUser with the user/password combination that was used by the requestor to authenticate against the infrastructure component. As for the Action data type the framework provides the capability to pass arbitrary key/value pairs in the properties attribute also on the endpoint level.

Dynamically calling an endpoint requires some extra work in the business logic. For instance, the endpoint URL has to be transformed into a format that can be passed to the WPS invocation facilities. In order to make it easier for the developer of business logic that is called by the SubscriptionManager, the framework already constructs a WS-Addressing ServiceRefType out of the endpoint URL and passes it in the epr attribute. This attribute can be assigned, for example, to a partner reference in a business process. An example is given in the XR SimpleQueryProcess. Table 8-5 on page 228 summarizes the Endpoint attributes.

Table 8-5 Endpoint attributes

Attribute	Description
id	The ID of this endpoint. The ID can be used to identify the desired endpoint in the endpoints list of the action.
url	The endpoint address, for example, a Web service endpoint or a SCA Export.
description	An optional description of the endpoint.
useDefaultUser	A boolean flag telling the infrastructure component to use the defaultUser attribute (if true) or overwrite the defaultUser with the current user otherwise.
defaultUser	A user (userId/password combination) that must be used when accessing the endpoint.
properties	An array of key/value pairs that can contain arbitrary data associated with the endpoint.
epr	A WS-Addressing ServiceRefType. This attribute is set by the infrastructure component using the URL attribute.

**Note:** The interfaces and data types described in the this chapter are packaged in the PDIFLibrary.

## 8.2.5 SubscriptionManager Administration

The configuration data is stored in an XML file. Changes can be done directly by editing the respective file (SubscriptionManagerConfig.xml). In addition, a Web administration client is provided to view and modify the subscriptions. Configuration changes take effect immediately. It is not necessary to restart the infrastructure component. The administration client (Figure 8-7 on page 235) is demonstrated and used Section 8.5, “Running the scenario” on page 248.

Subscription Manager					
<input type="radio"/>	type: query source: CabinDept user: * description:	true	type: PLMQueryIF url: sca://TruckInc/SimpleQueryExport description:	id: <input type="text"/>	pdmsystem: http://localhost:9090/PLMS
<input type="radio"/>	type: query source: TrailerDept user: * description:	true	type: PLMQueryIF url: sca://TruckInc/SimpleQueryExport description:	id: <input type="text"/>	pdmsystem: http://localhost:9090/PLMS
<input type="radio"/>	type: query source: TruckDept user: * description:	true	type: PLMQueryIF url: sca://TruckInc/FederatedQueryExport description:	id: <input type="text"/>	pdmsystem1: http://localhost:9090/PLM pdmsystem2: http://localhost:9090/PLM
<input type="radio"/>	type: writeMessages source: ECRClient user: * description:	true	type: PLMWriteMessagesIF url: sca://TruckInc/InitializeECRExport description:	id: <input type="text"/>	url: <input type="text"/> description: <input type="text"/> useDefaultUser: <input type="checkbox"/>
<input type="radio"/>	type: query:Messages source: ECRClient user: * description:	true	type: PLMQueryMessagesIF url: sca://TruckInc/QueryECRMessagesExport description:	id: <input type="text"/>	url: <input type="text"/> description: <input type="text"/> useDefaultUser: <input type="checkbox"/>
<input type="radio"/>	type: query source: ECRClient user: * description:	true	type: PLMQueryIF url: sca://TruckInc/FederatedQueryExport description:	id: <input type="text"/>	pdmsystem1: http://localhost:9090/PLM pdmsystem2: http://localhost:9090/PLM
<input type="button" value="Edit subscription"/> <input type="button" value="Remove subscription"/> <input type="button" value="Add subscription"/> <input type="button" value="Reload configuration"/> <input type="button" value="Update configuration"/>					

Figure 8-3 Subscription Manager Web administration client

**Note:** The current implementation does not provide the possibility to save modified configurations (for example, after publishing or restarting the server configuration changes are lost). It is up to the reader to extend the Web administration client with the capability to persist changes.

## 8.2.6 Sequence diagram of a sample request (query)

To deepen the understanding of the infrastructure component, this section shows the sequence diagram of a sample request (query) as shown in Figure 8-4 on page 230.

A PLM Services 2.0 client calls the `get_general_connection` method of the `PLMS2Infrastructure` connection factory specifying user, password, and source information. A `PLM_session` (wrapped in a `Session_parameter` object) is returned in case the user/password combination is valid. The `PLM_session`

consists of a session\_id and the URL of the PLM\_general\_connection Web service endpoint to be used in further calls. In this example, the client calls the query method of the PLM\_general\_connection interface passing the session\_id and the query (of type PLM\_query) to be executed.

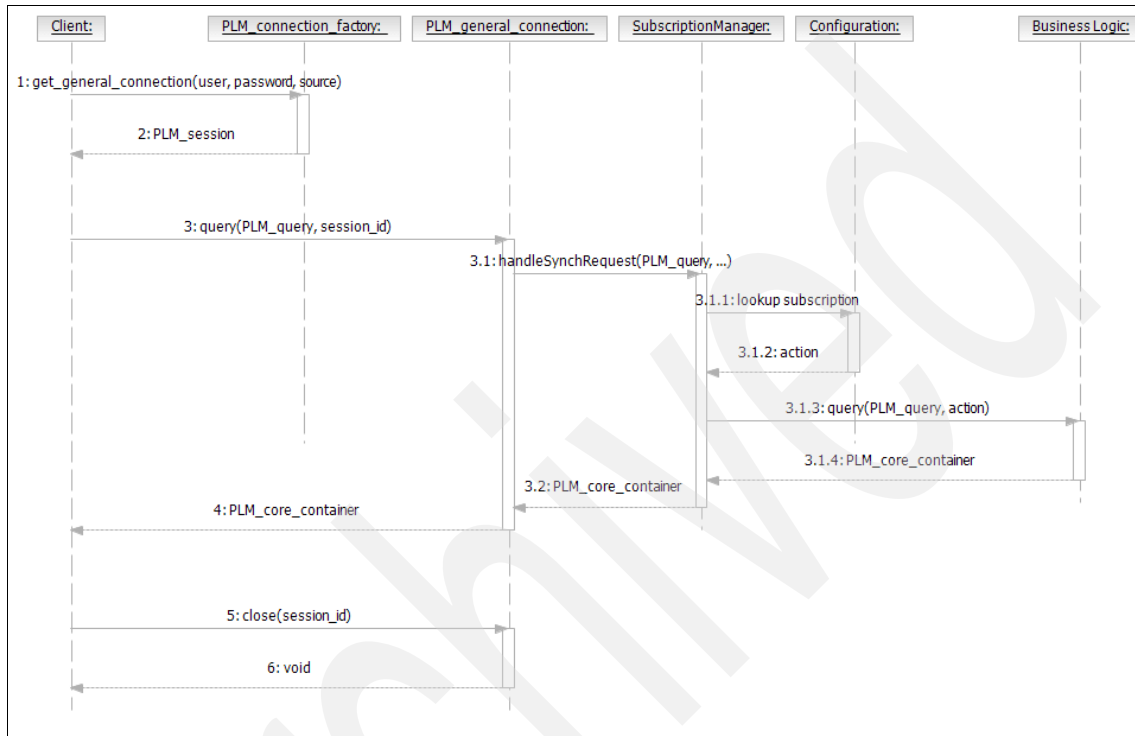


Figure 8-4 Sequence diagram of a query request

The SubscriptionManager implementation uses the session\_id to retrieve the associated user and source property values that were passed in the get\_general\_connection call and uses this information together with the request type (query) to search for a matching subscription in the configuration. In case no matching subscription for the specified user is found the search is extended to a subscription with source, type and arbitrary user (\*). If this search fails too, an appropriate exception is raised. In case a matching subscription is found, the associated action data structure is aligned. For example, the user/password properties are populated and WS-Addressing compliant endpoint references are created out of the endpoint URLs. The specified business logic (a Web service or a SCA component exposed with an Export) is called, passing the original parameter (PLM\_query) and the preprocessed Action object. The called business logic can use that information, for example, to dynamically invoke the

passed endpoint references to calculate a result. The result is returned to the SubscriptionManager that itself passes it back to the client.

The client can issue further calls. At some point in time the client closes the session by calling the close method of the PLM\_general\_connection interface passing the session\_id. After closing the session, further calls on behalf of that session id raises an invalid session id exception.

The scenario shown in Figure 8-4 on page 230 uses the PLM\_general\_connection interface. Call sequences using the PLM\_message\_connection interface work analogous.

### 8.2.7 Building components that use the framework

The previous section introduced the functionality and various aspects of the SubscriptionManager. Now the steps to design and develop business logic that can be plugged into the framework are outlined briefly:

1. Decide what type of requests (query, write, and so forth) must trigger the business logic.
2. Use Table 8-1 on page 224 and Table 8-2 on page 225 to pick the appropriate interface for the business logic.
3. Create a component (for example, a business process) that implements a this interface.
4. Populate the component skeleton with business logic.
  - a. Decide what endpoints and properties are variant and must be externalized into the SubscriptionManager configuration.
  - b. Build the business logic that takes those endpoints and properties from the passed Action instance and uses it accordingly.
5. Generate an SCA Export for the component or expose it as Web service.
6. Add one or multiple subscriptions to the SubscriptionManager configuration, among others specifying the endpoint address of the business logic and the parameters (endpoints, properties) that must be passed.

Section 8.4, “Building and reviewing a query scenario” on page 237 contains an example with step-by-step instructions.

## 8.3 Reviewing the implementation

Section 8.1, “Motivation” on page 221 and Section 8.2, “Subscription-based routing” on page 223, discussed the SubscriptionManager functionality. This section provides an overview on how it is implemented.

### 8.3.1 Reviewing SubscriptionManager implementation

This section explains a step-by-step implementation of the SubscriptionManager functionality.

1. Start a new workspace, and import PLMS2Infrastructure\_P1.zip project interchange file into your workspace. This project interchange file contains pre-built SCA modules and libraries.
  - a. Select **File** → **Switch Workspace....**
  - b. Enter a new workspace name in the Workspace Launcher window to start a clean workspace, then click **OK**. This restarts the WebSphere Integration Developer.
  - c. When the new workspace is launched, close the Welcome window to proceed.
  - d. Select **File** → **Import....**
  - e. Select **Project Interchange** in the Select window then click **Next >**.
  - f. In the Import Projects window, click **Browse...** and navigate to where you extracted the downloaded Redbook247593.zip lab file (see Appendix A, “Additional material” on page 465).
  - g. Select PLMS2Infrastructure\_P1.zip, and click **Open**.
  - h. Click **Select All**, then click **Finish**.

**Note:** During the implementation, we experience some common install errors. These errors are resolved by completing the next steps.

2. Modify the xmlFile parameter value in the config.properties based on your current workspace path for PLMS2Server and PLMS2Server2.

**Note:** Refer to step 2 of Chapter 7, “Implementing PLM Services 2.0-based solutions on IBM middleware” on page 177 for step by step instructions on modifying the xmlFile parameter value.

3. Modify the Application Deployment Descriptor file for PLMS2InfrastructureApp to include PLMS2AdminWeb. This module contains the SubscriptionManager administration Web client.
  - a. Switch to Java perspective, and expand **PLMS2InfrastructureApp** → **EarContent** → **META-INF**.
  - b. Navigate to the application.xml file and double-click the file to open it in the deployment descriptor editor.
  - c. Select the Module tab.
  - d. Select **Web PLMS2InfrastructureWeb.war**, and click **Remove**.
  - e. Click **Add...**, select **PLMS2AdminWeb** from the list, and click **Finish**.
  - f. Click **Add...**, select **PLMS2InfrastructureWeb** from the list, and click **Finish**.

**Important:** PLMS2AdminWeb.war and PLMS2InfrastructureWeb.war must be added in this order to run the application successfully.

- g. The deployment descriptor must look as it is shown in Figure 8-5.

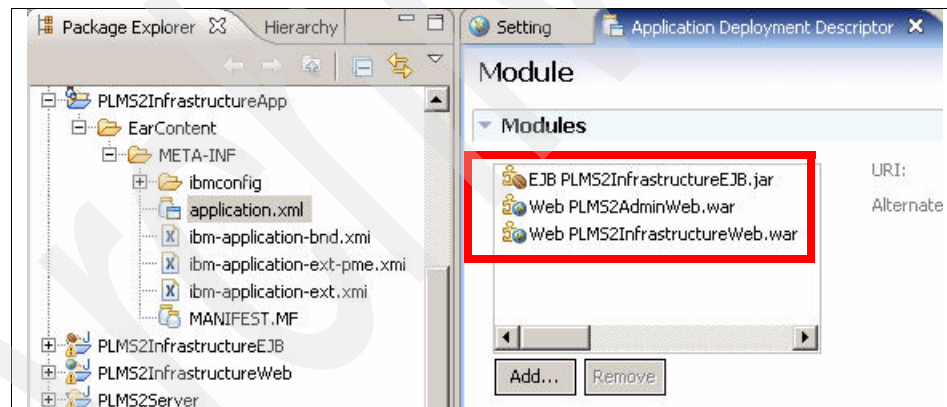


Figure 8-5 PLMS2InfrastructureApp deployment descriptor

- h. Save and close the deployment descriptor.
4. Review the contents of the PDIFLibrary. This library contains the interfaces and data types discussed earlier.
5. Open the PLMS2Infrastructure assembly diagram to review the implementation.

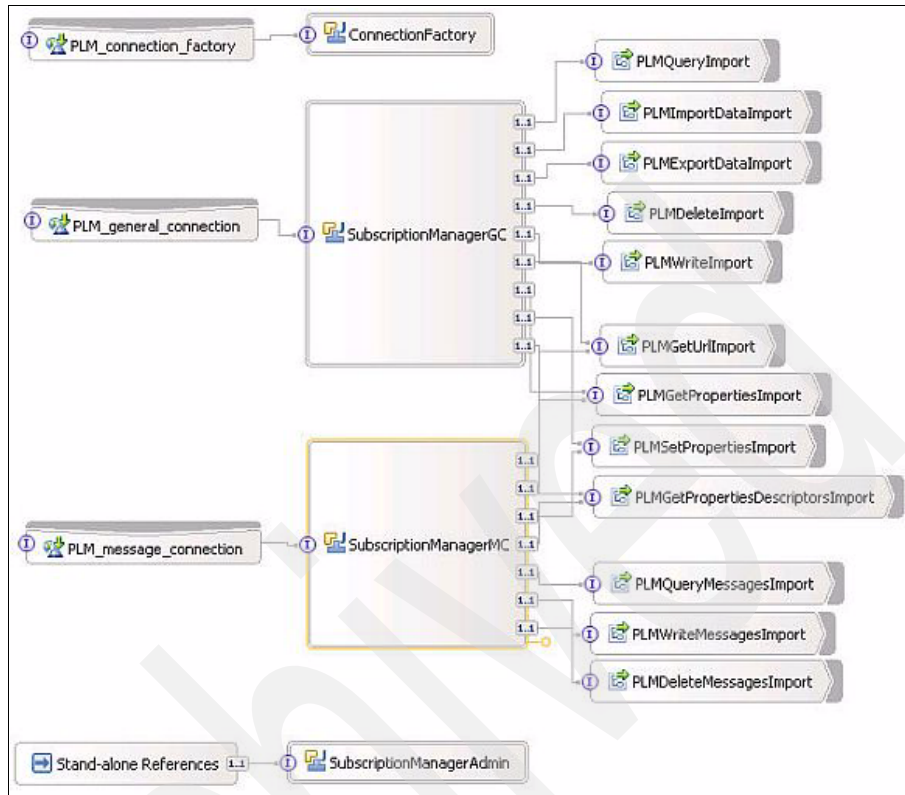


Figure 8-6 PLMS2Infrastructure assembly diagram

The PLMS2Infrastructure module implements the PLM\_connection\_factory, PLM\_general\_connection, and PLM\_message\_connection interfaces as you have done for the PLMS2Server in Chapter 9, “Design and implementation of an engineering change scenario including SAP integration” on page 255. This allows PLM Services 2.0 clients to communicate with the infrastructure component as it would be a PLM Services 2.0 server.

After the client establishes a connection to the PLMS2Infrastructure, the client can send requests (like query, write, and delete).

6. Double-click the SubscriptionManagerGC service component to open its implementation, SubscriptionManagerGCImpl.java.

Review how the PLM\_general\_connection operations are implemented. They call the SubscriptionManager.handleSynchRequest() method passing the request type, sessionId, and received parameter. Figure 8-1 on page 223 shows the implementation for the query() method.



### Example 8-1 query() method implementation in SubscriptionManagerGC

```
public DataObject query(DataObject parameters, DataObject sessionId) {  
    return SubscriptionManager.getInstance().handleSyncRequest("query",  
        sessionId, parameters);  
}
```

Press Ctrl and hover your cursor under the `handleSyncRequest` in this line of code. This opens the `SubscriptionManager.java` in the Java editor.

First, the `handleSyncRequest()` method checks the validity of the passed `sessionId` and provides an `InvalidSessionIdException` if this is not the case. If the `sessionId` is valid, the associated user, password, and source properties are retrieved. The key consisting of request type, user, and source property is constructed and the configuration (represented in the form of a hash table) is searched for a matching subscription. If none is found, a matching subscription for request type, arbitrary user ("\*") and source is looked up. If nothing is found, an appropriate exception is thrown to the caller.

If a matching subscription was found, the Action data is preprocessed as described in 8.2.3, "User credential propagation" on page 225 and 8.2.4, "Configuration data model" on page 226. Afterwards, the retrieved business logic is called to fulfill the request. The request input parameter and the preprocessed Action data is passed. The result is returned to the caller.

**Note:** `SubscriptionManager.java` contains also the methods to read the configuration from an XML file and modify the configuration.

## Reviewing SubscriptionManager Web administration client

Subscriptions can be configured through the SubscriptionManager Web administration client at runtime. For example, endpoints can be added or modified.

**Note:** In Section 8.5, "Running the scenario" on page 248, you are going to launch the SubscriptionManager client.

1. Open the PLMS2Infrastructure assembly diagram. Notice that there is a stand-alone reference component for SubscriptionManagerAdmin service component as shown in Figure 8-7. This standalone reference is used by the PLMS2AdminWeb Web application to access the SubscriptionManager.



Figure 8-7 Stand-alone reference for SubscriptionManagerAdmin

2. Double-click the SubscriptionManagerAdmin component to open its Java implementation. The getConfig() method invokes the getConfiguration() method of SubscriptionManager class. Also, the setConfig() method invokes the setConfiguration() method of the SubscriptionManager.

The getConfig() and setConfig() methods are used by the PLMS2Admin Web application to retrieve and change the SubscriptionManager configuration.

3. In the Java perspective, expand **PLMS2AdminWeb** → **Java Resources: src** → **dataTypes**. Locate ConfigHandler.java to view its implementation. Review the getConfiguration() and setConfiguration() methods that access the getConfig()/setConfig() methods of the SubscriptionManger through the standalone reference. Example 8-2 shows the getConfiguration() method.

*Example 8-2 getConfiguration() method*

---

```
public DataObject getConfiguration() {
    try {
        if (adminIF==null)
            adminIF =
                (SubscriptionManagerAdminIF)ServiceManager.INSTANCE.locateService("SubscriptionManagerAdminIFPartner");
        DataObject config = adminIF.getConfig(null);

        if (config==null)
            config =
                DataFactory.INSTANCE.create("http://PLMS2Infrastructure/datatypes",
                "Subscriptions");
        return config;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

---

**Note:** The PLMSAdminWeb JSF application is not reviewed in this chapter. If you are interested in the implementation, review ConfigHandler.java, DisplayConfig.jsp, and EditSubscription.jsp.

## 8.4 Building and reviewing a query scenario

Two departments inside Trucks Inc, CabinDept and TrailerDept, are responsible for designing the truck cabin and the truck trailer respectively. Each department has its own PDM system, exposed as PLM Services 2.0 server.

The scenario implemented in this section describes how the infrastructure component described earlier can be used to provide different views on the PDM data. The following conditions apply:

- ▶ Members of the Cabin department must only see data from within the Cabin PDM system.
- ▶ Members of the Trailer department must only see the data of Trailer PDM system.
- ▶ Members of an additional department (TruckDept) must have a federated view of the data.

### 8.4.1 Building business logic that plugs into the framework

The first part of this section describes how to build a business process (SimpleQuery) that can be plugged into the SubscriptionManager framework. The intent of this section is to give an example for the steps described in 8.2.7, “Building components that use the framework” on page 231. Building the business process is optional, as the full solution is imported later.

The SimpleQuery business process implements a query call to one PLM Services 2.0-exposed PDM system. Implementing the query requires the `get_general_connection` method of the respective `PLM_connection_factory` to be called (passing a valid user/password combination). The call returns a session object containing a `sessionId` and the endpoint address of the respective `PLM_general_connection` service. This endpoint, and the `sessionId`, are used to do the query call. Afterwards, the session is closed by calling the `close` operation on the `PLM_general_connection` service. Figure 8-8 on page 238 shows the SimpleQuery business process.

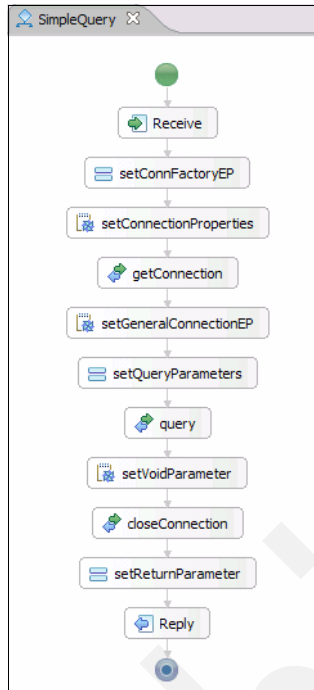


Figure 8-8 SimpleQuery business process

**Note:** Following the steps to build the SimpleQuery process is optional, as the complete business logic is imported as project interchange later. But it is highly recommended to perform the steps, as they demonstrate the following points:

- ▶ How to deal with and use the passed meta-information
- ▶ Utility methods that help you implement the solution
- ▶ The order of creating activities that decrease the effort to build the solution

The process of building business logic that plugs into the framework involves creating a new SCA module and creating a business process that implements a simple query. To do this, perform the following steps:

1. Create a new module named TruckInc or, for example, MyTruckInc if you want to prevent it getting overwritten when importing the whole solution.
  - a. Select **File** → **New** → **Project...**
  - b. Select **Module**, then click **Next >**.
  - c. Specify a module name (in this example, we use TruckInc). Click **Next >**.
  - d. Select PLMServices2WithSessionIDLib and PDIFLibrary as required libraries and click **Finish**.
2. Add the PLMS2Util project as dependent Java project.
  - a. From the Business Integration perspective, double-click **Dependencies** in the TruckInc module to open the Dependency editor.
  - b. Expand the **Java** section and click **Add...**
  - c. Select the PLMS2Util project (Figure 8-9) and click **OK**.

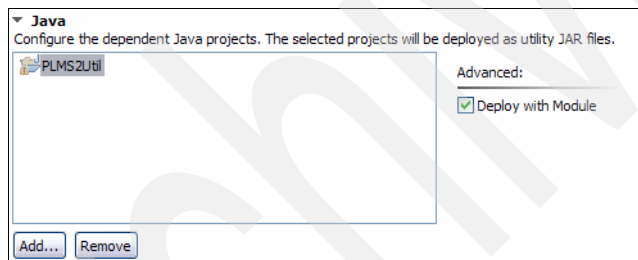


Figure 8-9 Add the PLMS2Util project.

- d. Save the changes and close the Dependency editor.

Create a business process that implements a simple query.

1. Create a business process named SimpleQuery.
  - a. Right-click **Business Logic** in the TruckInc project and select **New** → **Business Process** as shown in Figure 8-10.

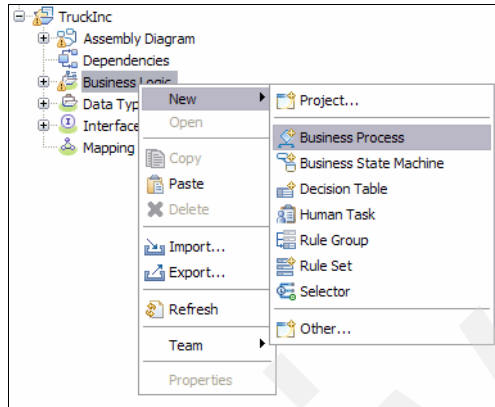


Figure 8-10 Create a new business process.

- b. Specify SimpleQuery as the process name. Click **Project**. Click **Next >**.
    - c. Select the Microflow radio button and click **Next >**.
    - d. Specify **Select existing interface** and click **Browse....**
    - e. Select the PLMQueryIF interface (Figure 8-11) and click **OK**.

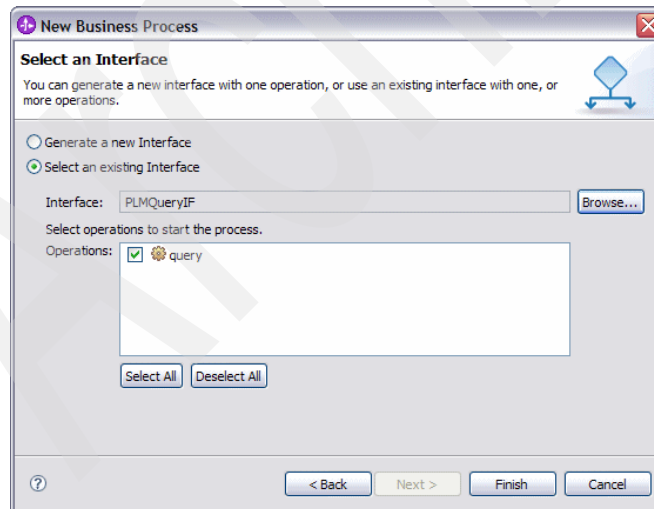


Figure 8-11 Select the PLMQueryIF interface-

- f. Click **Finish**. The new business process is displayed in the BPEL editor.

2. Implement the SimpleQuery business process.
  - a. Create a partner reference for PLM\_connection\_factory by dragging and dropping the PLM\_connection\_factory in PLMServices2WithSessionIDLlib into the canvas as shown in Figure 8-12.

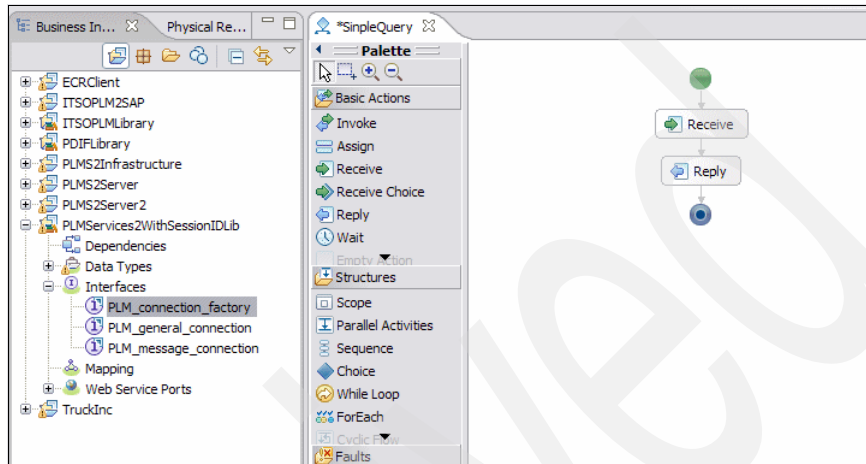


Figure 8-12 Creating a partner reference for PLM\_connection\_factory

- b. Create a partner reference for PLM\_general\_connection by dragging and dropping the PLM\_general\_connection in PLMServices2WithSessionIDLlib into the canvas as shown in Figure 8-13.

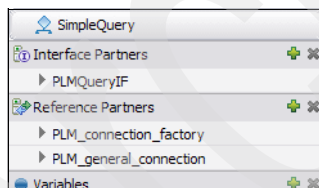


Figure 8-13 The created reference partners

- c. Rename the parameters variable to inputParameter.
  - d. Rename the parameters1 variable to returnParameter.
  - e. Create a new variable named user of data type User.
  - f. Create an Assign activity after the Receive activity and rename it to setConnFactoryProperties.

**Note:** The following steps show how to access and use endpoint and user information passed by the SubscriptionManager.

- g. In the details view of the assign activity, specify **Partner reference** → **PLM\_connection\_factory** in the Select To column.
- h. In the details view of the assign activity, specify **action** → **endpoints** → **epr** in the Select From column,
- i. Right-click **action.endpoints[1]/epr** (Figure 8-14) and select **Edit Query...** Replace “1” with id=”pdmsystem”.

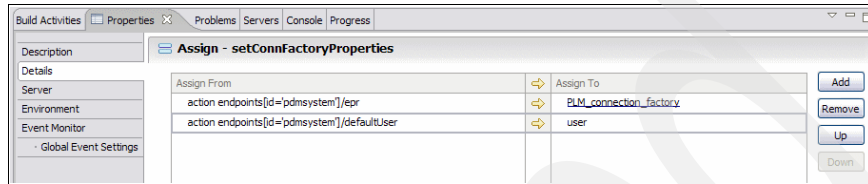


Figure 8-14 Assign setConnFactory Properties

- j. Add another assignment  
**action.endpoints[id='pdmsystem']/defaultUser** → **user**.
- k. Create an Invoke activity after the Assign activity and rename it to getConnection.
- l. In the details view of the Invoke activity, select PLM\_connection\_factory as Partner and the get\_general\_connection as operation.
- m. In the details view of the Invoke activity (Figure 8-15), click ... in the input(s) parameters row and create a new variable named connectionProperties.

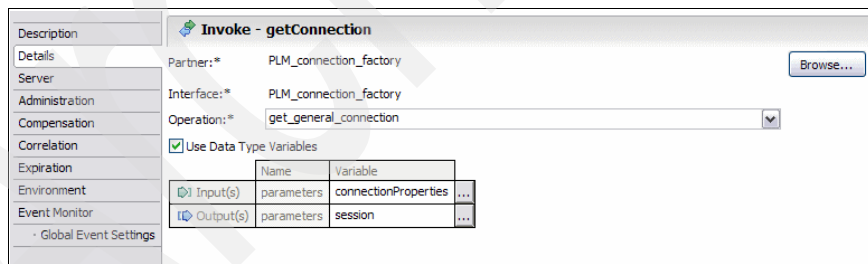


Figure 8-15 Details view of getConnection invoke activity.

- n. In the details view of the Invoke activity, click ... in the Output(s) parameters row and create a new variable named session.
- o. Create a snippet activity between the Assign and Invoke activity and name it setConnectionProperties.



- p. In the details view of the snippet, switch to Java and enter the following code to create connection factory properties with user and password information:
- q. Create a snippet activity after the getConnection Invoke activity and name it setGeneralConnectionEP.
- r. In the details view of the snippet, switch to Java and enter the following code to set the partner reference of the PLM\_general\_connection partner to the endpoint address returned by the get\_general\_connection call in the session variable:

```
connectionProperties =
omg.org.plm20.util.PLMServicesUtil.getConnectionProperties(user);

setServiceRefToPartnerLink("PLM_general_connection",
omg.org.plm20.util.PLMServicesUtil.getEndpointReferenceFromPLMSes
sion(session));
```

**Note:** The setServiceRefToPartnerLink method is available in Java snippets and sets the endpoint address of the named partner reference. As the session variable contains the endpoint address in form of a URL string, a utility method is used to construct a WS-Addressing ServiceRefType instance.

- s. Create an Invoke activity after the setGeneralConnectionEP snippet and rename it to query.
- t. In the details view of the Invoke activity, perform the following steps:
  - i. Select PLM\_general\_connection as Partner and the query as operation.
  - ii. Click ... in the input(s) parameters row and create a new variable named queryParameter.
  - iii. Click ... in the input(s) sessionId row and create a new variable named sessionId.
  - iv. Click ... in the Output(s) parameters row and create a new variable named queryReturn.
- u. Create an Assign activity between the setGeneralConnectionEP and the query activity and rename it to setQueryParameters
- v. In the details view of the assign activity, specify the following assignments:
  - inputParameter query → queryParameter query**
  - session return/Id → sessionId value.**
- w. Create an Invoke activity after the query Invoke activity and rename it to close.

- x. In the details view of the closeConnection Invoke activity, select **PLM\_general\_connection** as Partner and **close** as operation.
- y. In the details view of the closeConnection Invoke activity, click ... in the input(s) parameters row and create a new variable named voidParameter.
- z. In the details view of the Invoke activity, click ... in the input(s) sessionId row and select the existing sessionId variable.
- aa. In the details view of the Invoke activity, click ... in the output(s) parameters row and select the existing voidParameter variable.
- ab. Create a snippet activity between the query and closeConnection activities and name it setVoidParameter.
- ac. In the details view of the snippet, switch to Java and enter the following code to create a new VoidParameter instance:
 

```
voidParameter = (new
omg.org.plm20.services.parameter.impl.VoidParameterImpl()).getSDO
();
```
- ad. Create an Assign activity between the closeConnection and the Reply activities and rename it to setReturnParameter.
- ae. In the details view of the assign activity, specify the following assignment:
 **queryReturn queryreturn → returnParameter queryReturn.**
- af. Save the business process.

## Reviewing the complete query scenario

Here are the steps to import and review the complete TruckInc module:

1. Import the TruckInc\_P1.zip.
  - a. Select **File → Import....**
  - b. In the Select window, select **Project Interchange**, then click **Next >**.
  - c. In the Import Projects window, click **Browse...** and navigate to where you extracted the downloaded Redbook247593.zip lab file (see Appendix A, “Additional material” on page 465).
  - d. Select TruckInc\_P1.zip, and click **Open**.
  - e. Click **Select All**, then click **Finish**.

**Note:** There are some errors because the ECRWebClient Web project is not associated with the ECRClientApp enterprise application. But the next steps explain how to fix these errors.

2. Modify the Application Deployment Descriptor file for ECRClientApp to include the ECRWebClient project.
  - a. Switch to Java perspective, and expand **ECRClientApp** → **EarContent** → **META-INF**.
  - b. Locate the application.xml file and double-click it to open it in the deployment descriptor editor.
  - c. In the Module tab, select **Web ECRClientWeb.war**, and click **Remove**.
  - d. Click **Add...**, select **ECRWebClient** from the list, and click **Finish** to add.
  - e. Click **Add...**, select **ECRClientWeb** from the list, and click **Finish** to add.
  - f. Save and close the deployment descriptor. There must be no more errors in the Problems view.

**Note:** The TruckInc module imported in this step also includes the ECR/SAP scenario described in the next chapter.

3. Open the TruckInc assembly diagram (Figure 8-16) to review its implementation.

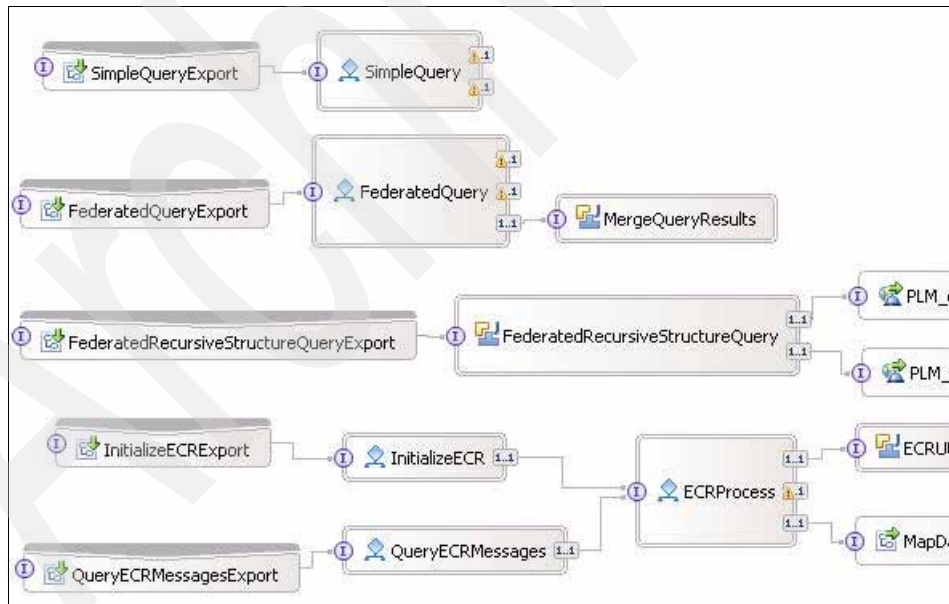


Figure 8-16 TruckInc assembly diagram

- Double-click the SimpleQuery component in the assembly diagram shown in Figure 8-17 to open the BPEL process and review its implementation in case you skipped the “Building business logic that plugs into the framework” on page 237. Refer to this section in case of questions.

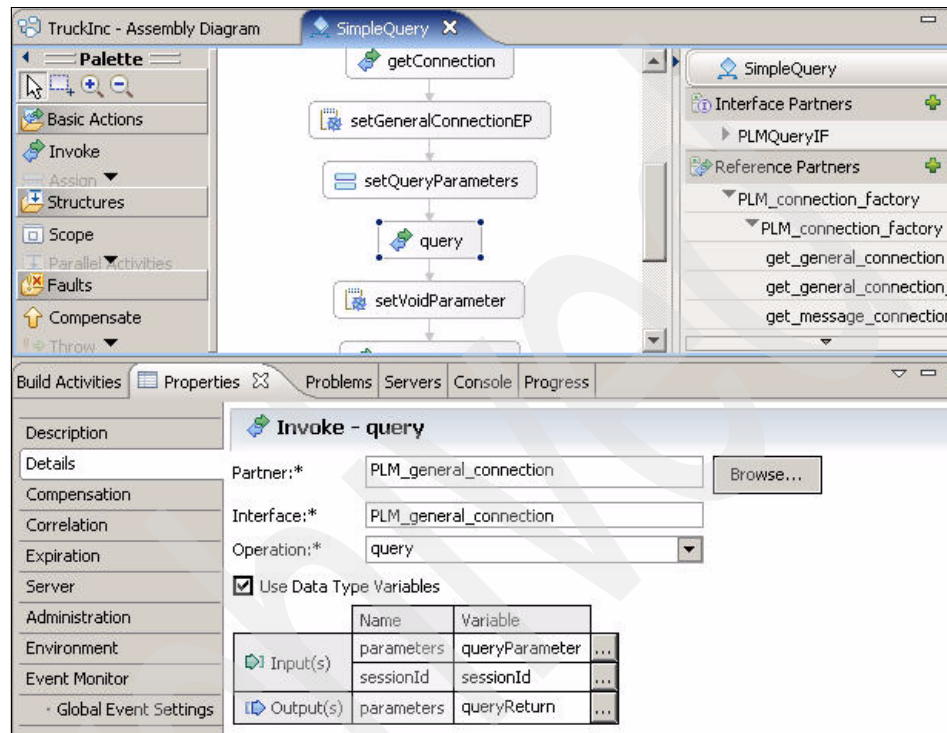


Figure 8-17 SimpleQuery business process template

- Double-click the FederatedQuery component (Figure 8-18) in the assembly diagram to open the BPEL process and review its implementation.

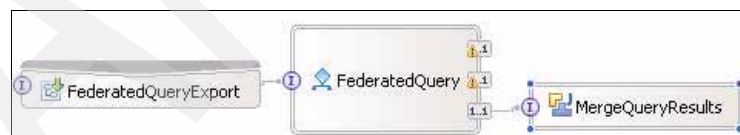


Figure 8-18 Federated query

The FederatedQuery business process is similar to the SimpleQuery business process. It expects two PDM system endpoints in the passed Action data and accesses both systems in the same way as the SimpleQuery executes the passed query. At the end the query results of both PDM systems are combined through the merge() method of the MergeQueryResults Java component. The

merge algorithm shown in Example 8-3 is kept quite simple. It copies every Item from the result set of the second query into the result set of the first query in case it does not already exist there.

**Note:** The current implementation queries the first PDM system and the second one afterwards. An alternative implementation would have been to do both queries in parallel. In addition the FederatedQuery process could have made use of the SimpleQuery process.

*Example 8-3 The merge() method.*

---

```
public DataObject merge(DataObject queryResult1, DataObject
queryResult2) {
    PLMContainer pc1 = new PLMContainerImpl(queryResult1);
    PLMContainer pc2 = new PLMContainerImpl(queryResult2);
    ItemList il = pc1.getItem();
    for (int i = 0; i < pc2.getItem().size(); i++) {
        Item item = pc2.getItem().get(i);
        if (!isItemContained(pc1, item.getUid()))
            il.add(item);
    }
    return pc1.getSDO();
}

private boolean isItemContained(PLMContainer pc, String uid) {
    Object item = pc.getSDO().get("item[uid='"+uid+"'");
    return item!=null;
}
```

---

**Note:** The other components in the assembly diagram are used in the Engineering Change Request (ECR)/SAP scenario and are reviewed in Chapter 9, “Design and implementation of an engineering change scenario including SAP integration” on page 255.

## 8.5 Running the scenario

In this section, you are going to run the query scenario.

**Note:** Some of the detail instructions have been omitted to avoid repetition. Refer to 9.4, “Running the scenario” on page 390 section if the step-by-step instructions are needed.

1. With the test server started, run all deployable applications in the workspace.
  - a. After the server has successfully started, right-click **WebSphere Process Server v6.1** in the Servers view, select **Add and remove projects...** from the context menu.
  - b. Click **Add All**, then **Finish**.
2. Start the TCP/IP monitor on port 9090 if it is not already started.
3. Launch the Subscription Manager Web tool to review the endpoint settings.
  - a. Start a Web browser, and enter the following address:  
`https://localhost:9443/PLMS2AdminWeb/Login.jsp`

**Note:** When the browser displays the security warning, select **Continue to this website (not recommended)** link to proceed.

The subscription manager login page appears as shown in Figure 8-19.

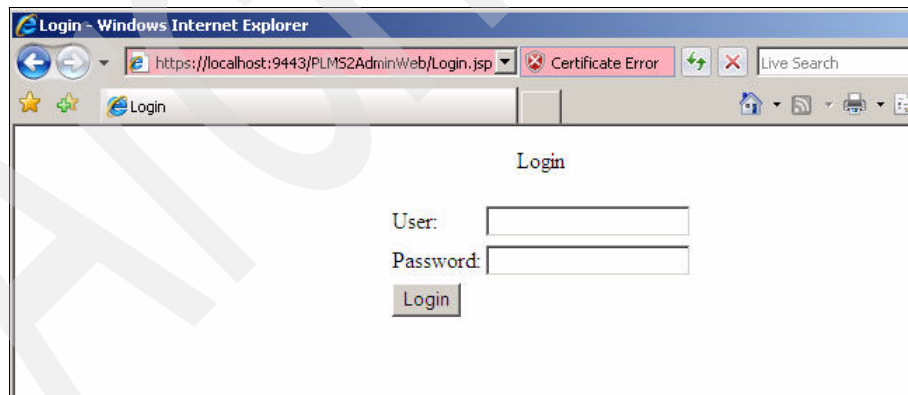


Figure 8-19 Subscription manager login page

The Web application is secured. The PLMS2InfrastructureApp enterprise application is configured to allow authenticated users access to the JSP™ pages.

- b. At the login page, enter a valid WebSphere user/password combination (for example, admin in both User and Password fields), and click **Login**. The Subscription Manager main page displays the list of subscriptions as shown in Figure 8-20.

IBM Subscription Manager									
<input type="radio"/>	type	query		type	PLMQueryIF				
	source	CabinDept	true	url	sca://TruckInc/SimpleQueryExport	id			
	user	*		description		pdmsystem	http://localhost:9090		
	description								
<input type="radio"/>	type	query		type	PLMQueryIF				
	source	TrailerDept	true	url	sca://TruckInc/SimpleQueryExport	id			
	user	*		description		pdmsystem	http://localhost:9090		
	description								
<input type="radio"/>	type	query		type	PLMQueryIF				
	source	TruckDept	true	url	sca://TruckInc/FederatedQueryExport	id			
	user	*		description		pdmsystem1	http://localhost:9090		
	description					pdmsystem2	http://localhost:9090		
<input type="radio"/>	type	writeMessages		type	PLMWriteMessagesIF				
	source	ECRCClient	true	url	sca://TruckInc/InitializeECRExport	id	url	description	useDefault
	user	*		description					
	description								
<input type="radio"/>	type	queryMessages		type	PLMQueryMessagesIF				
	source	ECRCClient	true	url	sca://TruckInc/QueryECRMessagesExport	id	url	description	useDefault
	user	*		description					
	description								
<input type="radio"/>	type	query		type	PLMQueryIF				
	source	ECRCClient	true	url	sca://TruckInc/FederatedQueryExport	id			
	user	*		description		pdmsystem1	http://localhost:9090		
	description					pdmsystem2	http://localhost:9090		
Edit subscription Remove subscription Add subscription Reload configuration Update configuration									

Figure 8-20 Subscription Manager configuration

Using the administration client, you can add, remove, and edit service subscription details. Also, you can update or reload the configuration.

There are already subscriptions defined:

- source: TruckDept → FederatedQueryExport in the TruckInc module gets invoked with the following endpoint parameters:  
`http://localhost:9090/PLMS2ServerWeb/sca/PLM_connection_factor`  
`y`  
`http://localhost:9090/PLMS2Server2Web/sca/PLM_connection_factor`  
`y`
- source: CabinDept → SimpleQueryExport in the TruckInc module gets invoked with the following endpoint parameter:  
`http://localhost:9090/PLMS2ServerWeb/sca/PLM_connection_factor`  
`y`
- source: TrailerDept → SimpleQueryExport in the TruckInc module gets invoked with the following endpoint parameter:  
`http://localhost:9090/PLMS2Server2Web/sca/PLM_connection_factor`  
`y`

**Note:** These subscriptions define the following situations:

- ▶ Members of the Cabin department only see data of the PLMS2Server system.
- ▶ Members of the Trailer department only see data of the PLMS2Server2 system.
- ▶ Members of the Truck department have a federated view on data of both PLMS2Server and PLMS2Server2 systems.

This setup can be easily changed by modifying the subscriptions.

4. Start the PLM Services reference client.
5. Create a new connection, CabinConnection.
  - a. In the Connections view of the client, right-click and select **New Connection...** from the context menu.
  - b. In the New Connection Factory window, enter CabinConnection in the Connection Name field.
  - c. Enter the following value in the Value field of Endpoint property.  
`http://localhost:9090/PLMS2InfrastructureWeb/sca/PLM_connection_f`  
`actory`
  - d. Click **Connect**. When the connection is established, click **Next >** to proceed.



- e. In the New Connection window, enter the following values:
    - user: test
    - password: test
    - source: CabinDept
  - f. Click **Finish** to establish a connection to the cabin department of the Trucks Inc.
6. Create a new connection, TrailerConnection.
- a. In the Connections view of the client, right-click **New Connection...** from the context menu.
  - b. Enter TrailerConnection in the Connection Name field.
  - c. Enter the Endpoint:  
`http://localhost:9090/PLMS2InfrastructureWeb/sca/PLM_connection_factory`
  - d. Click **Connect**. When the connection is established, click **Next >**.
  - e. In the **New Connection** window, enter the following values:
    - user: test
    - password: test
    - source: TrailerDept
  - f. Click **Finish** to establish a connection to the trailer department of the Trucks Inc.
7. Repeat the steps to create a new connection, TruckConnection, using the following values:
- Connection Name: TruckConnection
  - Endpoint:  
`http://localhost:9090/PLMS2InfrastructureWeb/sca/PLM_connection_factory`
  - user: test
  - password: test
  - source: TruckDept

Now, you have three connections as shown in Figure 8-21.



Figure 8-21 Connections to the PLM server

8. Use PDTnet query to query data from the cabin department.
  - a. In the Queries view, right-click **New Query...** from the context menu.
  - b. In the Query Wizard, select the following property values:
    - Query Category: PDTnet Queries
    - Query Subcategory: Selection Queries
    - Query Template: ItemSelection
    - name: .\* (wildcard)
  - c. Click **Next >**. The check box for all three connections must be selected.
9. Review the results of the queries.

Figure 8-22 shows the query result returned from the cabin department, ItemSelectionQuery1@CabinConnection.

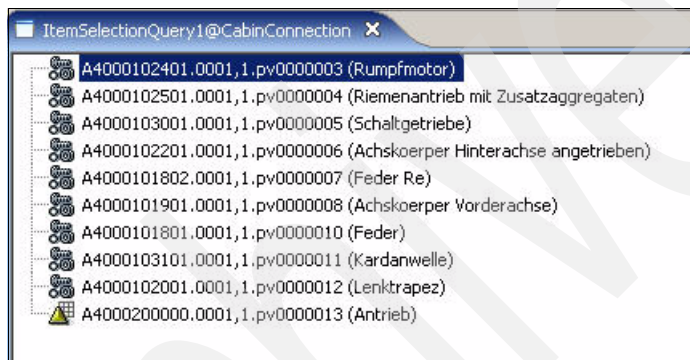


Figure 8-22 Item selection query result from Cabin Department

Figure 8-23 shows the query result returned from the trailer department, ItemSelectionQuery1@TrailerConnection.

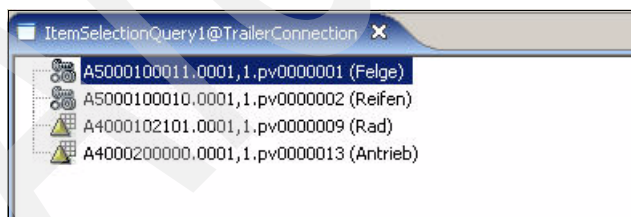


Figure 8-23 Item selection query result from Trailer Department

Figure 8-24 shows the query result returned from the trailer department, ItemSelectionQuery1@TruckConnection.

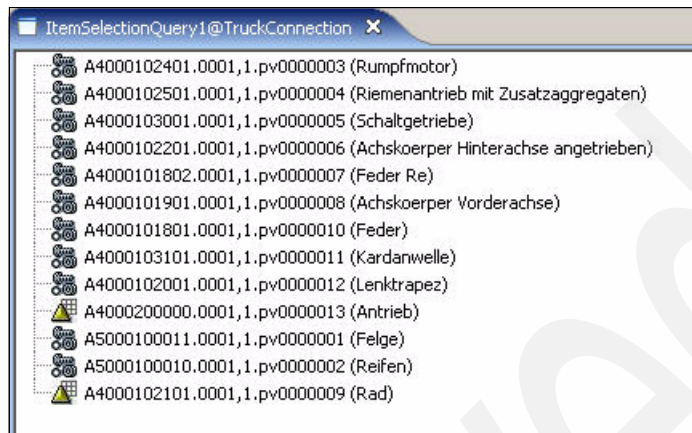


Figure 8-24 Item selection query result from Trucks Inc

Notice that the result of the TruckConnection query is the aggregated results of cabin and trailer department servers.

- Run the detail query from the item selection query to retrieve the detail information by double-clicking an item from the query result.
- In the ItemSelectionQuery1@TruckConnection view, right-click **Antrieb** and select **Launch Structure query**.

The subcomponents of item Antrieb are displayed in Figure 8-25:

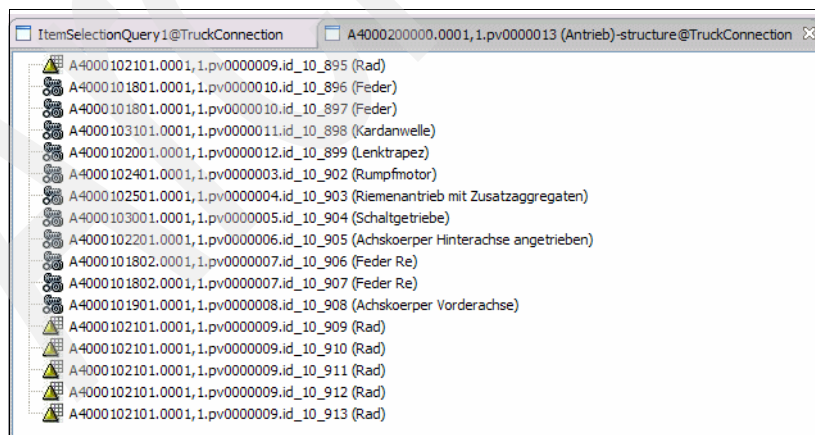


Figure 8-25 Subcomponents of item Antrieb

**Note:** If you try the structure query in the ItemSelectionQuery1@CabinConnection or ItemSelectionQuery1@TrailerConnection view, you get an error. The respective PDM systems do contain and return all subcomponents.

12. In the ItemSelectionQuery1@TruckConnection or ItemSelectionQuery1@TrailerConnection view, right-click **Rad** and select **Launch Structure query**.

## **Design and implementation of an engineering change scenario including SAP integration**

In this section, the components of the previous chapter are used to build a more complex scenario: An implementation of an engineering change request (ECR) that follows the outline of an ECR interaction scenario 'ECR participant proposal for a change' defined in the VDA 4965-1 recommendation. When the ECR is finally approved, the changes are propagated to an SAP system. All readers will not have access to a real SAP system, therefore the scenario can be configured where the data that would be transmitted to the SAP system is instead dumped to the console.

## 9.1 Scenario overview

This chapter discusses the process of how Trucks Inc. handles a mechanical issue that was raised during the manufacture, test, or in-service phase of a truck's manufacturing life cycle.

The ECR process is modelled in BPEL. The approval steps involved during the ECR are modelled as a human task and a WebSphere Process Server extension.

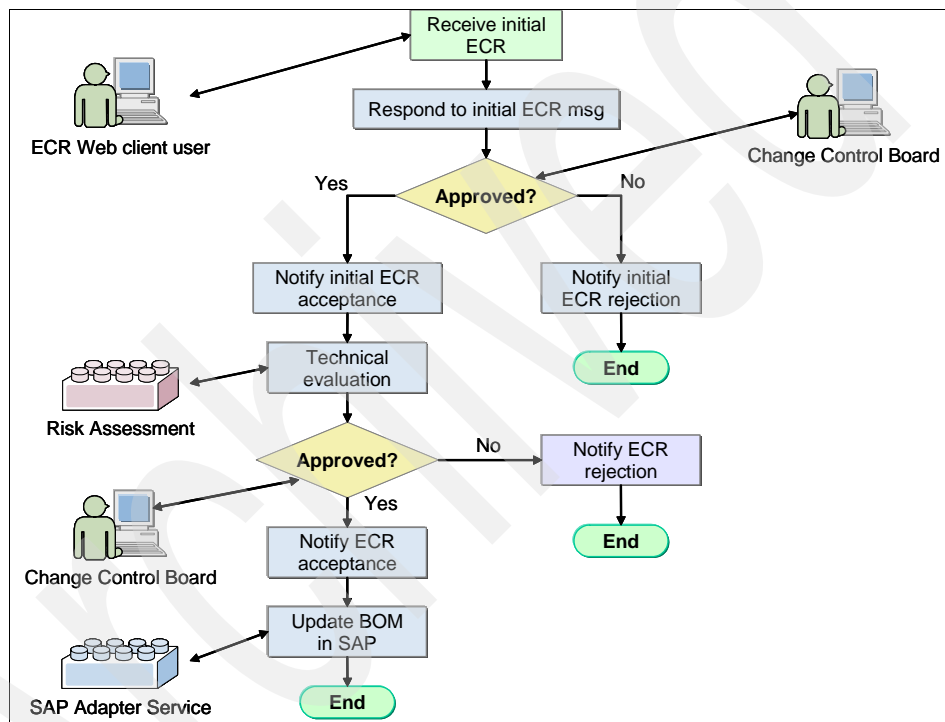


Figure 9-1 Scenario high level overview

The high level flow of the sample solution is captured in Figure 9-1 on page 256 and consists of the following steps:

1. A mechanical issue is detected. An authorized organization submits an ECR proposal description through a Web user interface. See Figure 9-2.

Figure 9-2 ECR Web client tool

2. The Change Control Board (CCB) reviews the submitted report and decides whether or not to raise an ECR. Any ECR can incur serious risk to the project in terms of time, cost, or quality, so there may be serious implications on proceeding. A to-do list for CCB members (Figure 9-3) can be generated to help in their review and appropriate tasks to be performed

The CCB usually consists of representatives from engineering, production, manufacturing, finance, procurement, supply chain management, and IT organizations.

Figure 9-3 To-do list for CCB members

3. The CCB generates an ECR if the problem is perceived to be of sufficient importance. The respective technical department analyzes the problem report and comes up with a solution proposal.
4. The proposed solution is assessed by the CCB and a decision is made whether the change is approved.

If the ECR is approved, the process triggers the release of the updated BOM to SAP.

5. After the data is sent to SAP, the SAP sends confirmation that the BOM has been consumed correctly.

During all the previous steps the ECR requestor can query the status of the ECR.

Section 9.2, “Engineering Change Request (ECR) process” on page 258 walks through the WS-BPEL processes, human task, and the Web clients in detail.

## 9.2 Engineering Change Request (ECR) process

Figure 9-4 captures the message exchange between a ECR requestor (P) and the ECR coordinator (C) party as defined in the VDA 4965-1 recommendation. The scenario implemented in this chapter follows this outline.

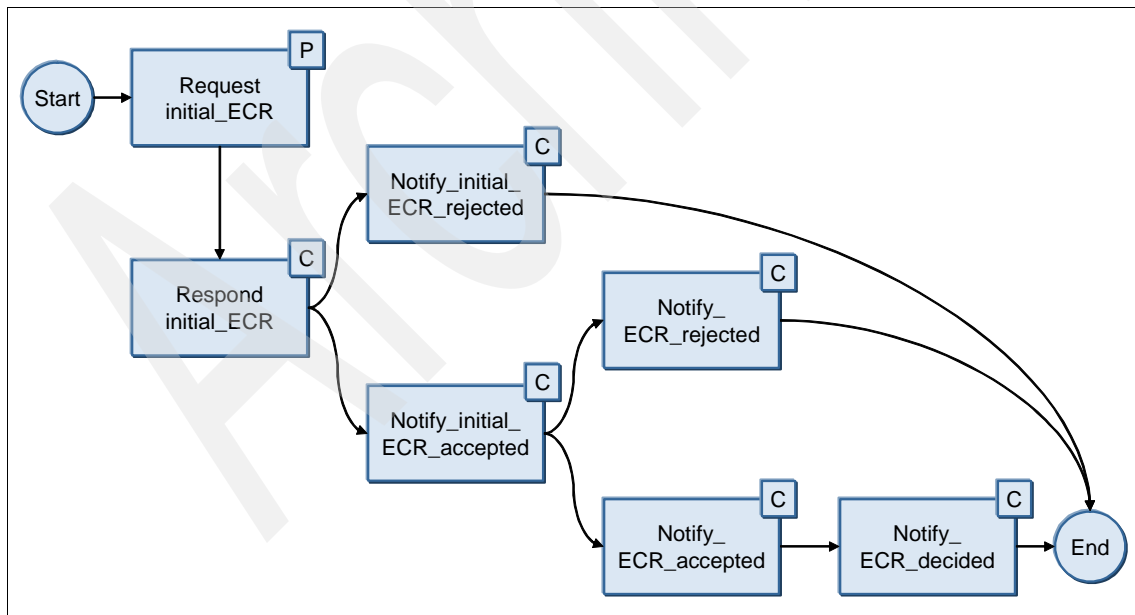


Figure 9-4 ECM participant proposal for a change



The ECR requestor triggers the ECR process by sending a Request\_initial\_ECR message to the ECR coordinator. This message is sent through the write\_messages operation of the PLM\_message\_connection service.

The coordinator notifies the requestor by sending respective messages as outlined in Figure 9-4 on page 258. But the messages are not sent directly. Instead, the ECR requestor asks for the current status by issuing a query\_messages call on the PLM\_message\_connection service. As the requestor can issue the query at any time, the coordinator can return zero, one, or multiple messages, reflecting the progress of the ECR process.

**Note:** The first implementation of this scenario used the ProStep iViP PLM Services reference client to start the ECR process. The reference client was based on a draft version of PLM Services 2.0. In the final version, the sections about ECM and ECR changed. The scenario in this chapter is still based on the old version.

The implementation of the scenario includes a simple JSF application that allows a ECR requestor to create an ECR request (send a Request\_initial\_ECR message through write\_messages) and retrieve the status of the ECR (using query\_messages). The ECR process itself is implemented as a BPEL process.

In order to decouple the client from the business logic, the SubscriptionManager framework described in the last chapter is used. The components called by the SubscriptionManager on behalf of the ECR client are the InitializeECR process to start the ECR process and QueryECRMessages process to retrieve the status of the ECR. When calling the InitializeECR process, the SubscriptionManager passes information about the involved PDM systems, and a flag that indicates whether an SAP system is present. The InitializeECR process itself passes that information to the ECR process.

As the ECR client can retrieve the status of the ECR at any time, event handlers in the ECR process are used to achieve that without blocking the processing of the ECR by the coordinator. In order to deliver messages from the coordinator to the ECR requestor only once, respective measures have to be taken in the business process. A flag (removeExistingMsgs) that indicates whether existing messages must be removed (in case they were already pulled by the ECR requestor) is used for that purpose.

## 9.2.1 Choreographing the message flow

The message flow is choreographed by the ECRProcess business process defined in the TruckInc service module. The following steps detail that process:

**Note:** It is assumed that you have run the scenarios described in chapters 7 and 8. If you did not do so, follow the steps to import PLMS2Infrastructure\_P1.zip and TruckInc\_P1.zip as described in 8.3.1, “Reviewing SubscriptionManager implementation” on page 232 and “Reviewing the complete query scenario” on page 244 to set up your environment before proceeding.

1. In the Business Integration view, double-click **Assembly Diagram** under the TruckInc module to open it in the assembly editor.

Chapter 8, “A framework to develop and run flexible PLM Services 2.0-based solutions” on page 221 discussed the components to implement simple and federated queries. This chapter focuses on the components of the assembly diagram shown in Figure 9-5.

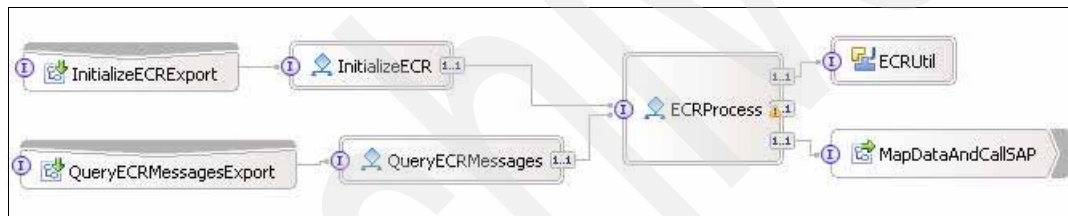


Figure 9-5 TruckInc assembly diagram

2. In the assembly editor, double-click **ECRProcess** to open it in the business process editor. See Figure 9-6 on page 261.

Select **initECR receive activity**, and review its properties, and notice that the arrival of initECR operation instantiates the ECRProcess business process.

The ECRProcess interface is located in the PDIFLibrary. This interface defines two operations, initECR and queryMessages.

Operations and their parameters		
	Name	Type
▼ initECR		
Input(s)	context	ECR_context
	container	PLM_container
	action	Action
▼ queryMessages		
Input(s)	query	ECR_message_query
Output(s)	messages	Messages_parameter
Fault	fault	PLM_exception

Figure 9-6 ECRProcess interface definition

The initECR operation takes ECR\_context, PLM\_container, and Action business objects as input.

The initECR and queryMessages operations get invoked by the InitializeECR and QueryECRMessages microflows respectively. Those components themselves get invoked by SubscriptionManager on behalf of requests from the ECR Web client. The ECR process is implemented as a BPEL process.

Figure 9-7 on page 262 displays the ECRProcess BPEL template.

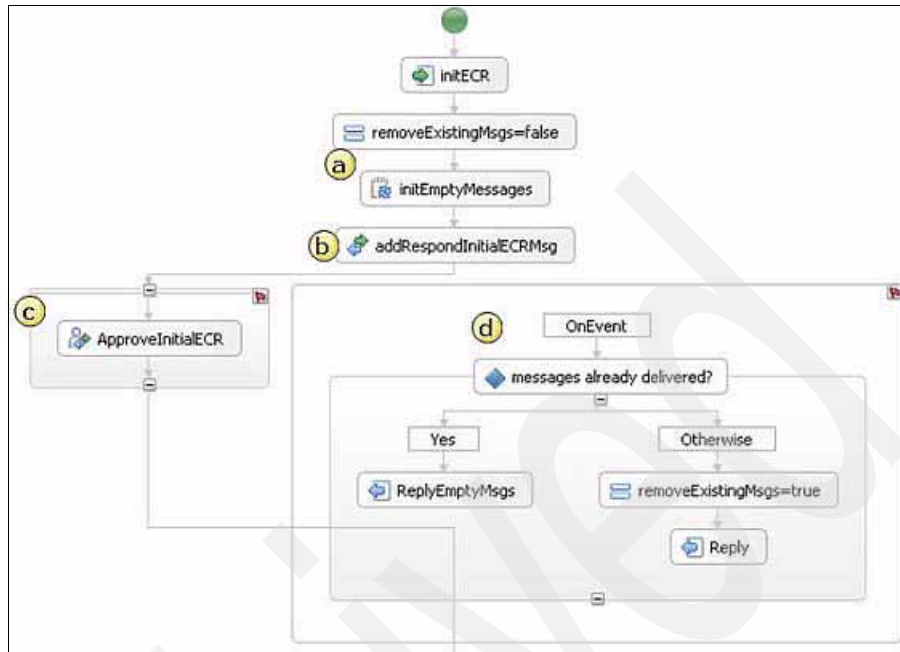


Figure 9-7 ECRProcess BPEL template

The removeExistingMsgs=False and initEmptyMessages activities initialize the Message\_parameter object.

The addRespondInitialECRMsg activity (Example 9-1 on page 263) invokes addRespondInitialECRMsg operation of the ECRUtil service and constructs the Respond\_initial\_ECR message using the Java wrapper classes described in 7.2.3, “Java wrapper classes for the PLM Services 2.0 data types” on page 189.

*Example 9-1 addRespondInitialECRMsg method implementation*

---

```
public DataObject addRespondInitialECRMsg(DataObject msgs,
    Boolean removeExistingMsgs, DataObject context, DataObject
    container) {
    MessagesParameter mp = null;
    if (msgs == null || removeExistingMsgs)
        mp = new MessagesParameterImpl();
    else
        mp = new MessagesParameterImpl(msgs);

    ECRContext ctx = new ECRContextImpl(context);
    ctx.setMessageName("Respond_initial_ECR");
    ctx.setRequesterDUNSNumber(ctx.getDUNSNumber());
    ctx.setRequesterMessageName("Request_initial_ECR");
    ctx.setSequenceNumber(2);
    ctx.setComplete(false);

    PLMMessageList msgList = mp.getReturn();
    PLMMessage msg = new PLMMessageImpl();
    msg.setPLMContext(ctx);
    msg.setPLMCoreContainer(new PLMContainerImpl(container));
    msgList.add(msg);

    return mp.getSDO();
}
```

---

When queryMessage operation is invoked by the ECR Web client, the Message\_parameter instance gets sent as the output.

- d. After the initial ECR report is submitted, it must be reviewed by the CCB members, which requires a human interaction. This is achieved through the ApproveInitialECR human task activity. Figure 9-8 shows the authorization setting of the ApproveInitialECR human task. The group 'CCB' is specified to ensure that only members of the CCB group can work on (approve) this human task.

To-do Task	
Name	ECRProcessTask1
Display Name	<Not Applicable>
Service Interface	
People Assignment (Receiver)	
Potential Owners	Group Members
	GroupName *
	IncludeSubgroups *
	CCB
	false

Figure 9-8 ECRProcessTask1 human task

- e. The ApproveInitialECR activity has an event handler associate with it. The OnEvent handles queryMessages events while the ApproveInitialECR activity is being worked on.

Keep in mind that the ECR owner and CCB members are two different parties in the ECR process. The ECR requestor uses the ECR Web client to create and query ECRs, while CCB members use BPC Explorer to work with the submitted ECRs.

- f. If the CCB decides that the ECR report is relevant, the next step is the `addNotifyInitialECRAcceptedMsg` activity (Figure 9-9). This activity invokes `addNotifyInitialECRAcceptedMsg` operation of the `ECRUtil` service.

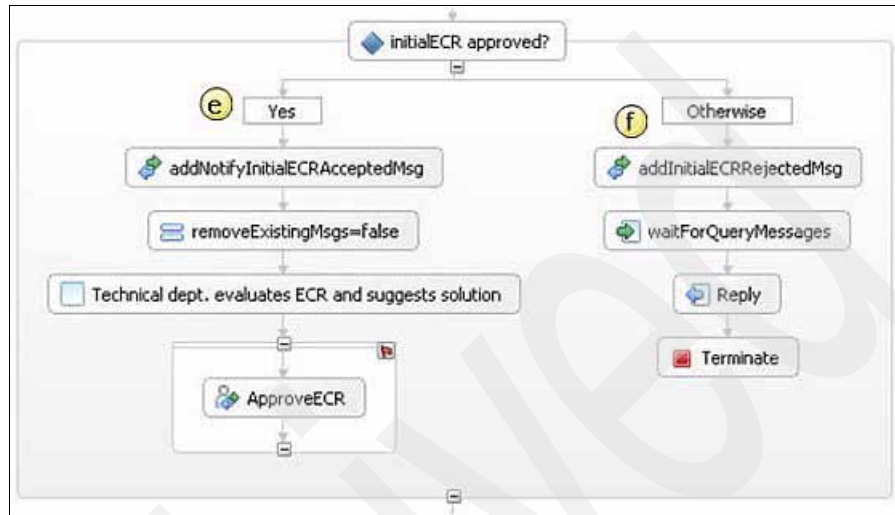


Figure 9-9 Choice activity, *initialECR approved?*

This method updates the `Message_parameter` instance and adds a `Notify_initial_ECR_accepted` message.

As mentioned in Section 9.1, "Scenario overview" on page 256, any ECR can have a serious impact in the production. Therefore, CCB members must evaluate the ECR from time, cost, and quality points of view before making the final decision to proceed with the engineering change of a vehicle. The BPEL empty activity `Technical dept evaluates ECR and suggests solution` represents that step.

After review, CCB member must either approve or disapprove the ECR.

- g. In case CCB decided that the ECR is not relevant to forgo, `addInitialECRRejectedMsg` operation of the `ECRUtil` service gets invoked. that adds a `Notify_initial_ECR_rejected` message.

The process waits until the ECR requestor queries the ECR status before terminating the process instance.

- h. If the ECR was approved in step f on page 265, the addNotifyECRAcceptedMsg operation of the ECRUtil service gets invoked. This method adds a Notify\_ECR\_accepted message.

The Assign activity sets the removeExistingMsgs local variable value to be False. See Figure 9-10.

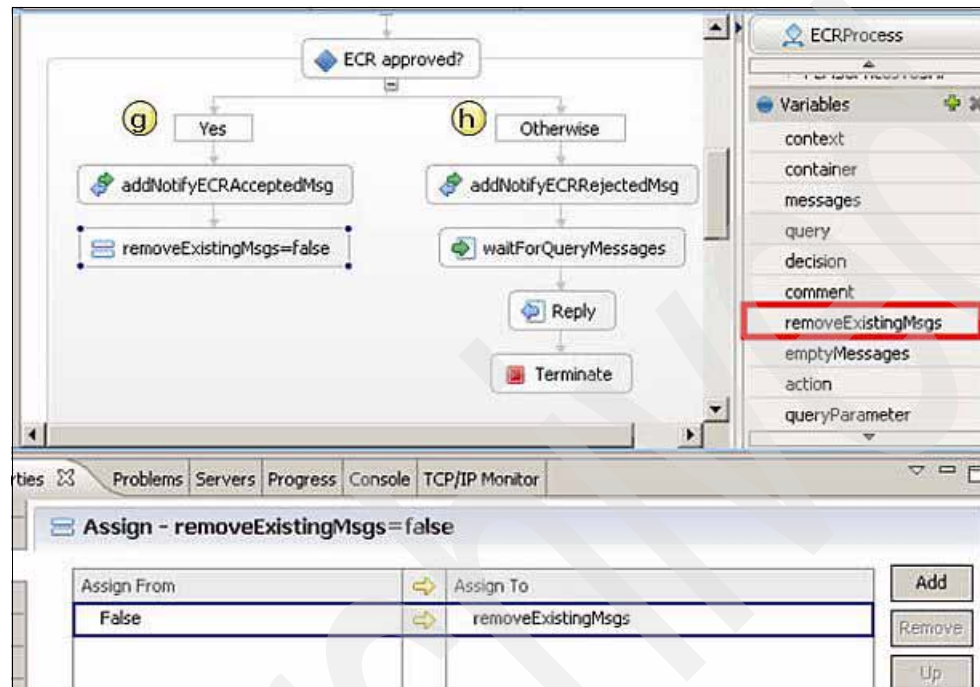


Figure 9-10 Choice activity, ECR approved?

- i. After the risk assessment, CCB rejects the ECR. In such case, addNotifyECRDeclinedMsg operation of the ECRUtil service gets invoked. This method adds a Notify\_ECR\_rejected message.

Similar to step g on page 265, the process waits until the ECR requestor queries the ECR status before terminating the process instance.



- j. When CCB decided to raise an ECR, a federated query (Figure 9-11) is performed to retrieve item details and documents of the affected part and all its subparts. For example, a wheel is consisted of rim and tire. Therefore, the federated query is performed in order to retrieve appropriate level of product data. In case you are interested in the implementation, double-click the FederatedRecursiveStructureQuery component in the Trucks Inc assembly diagram to review the Java code.

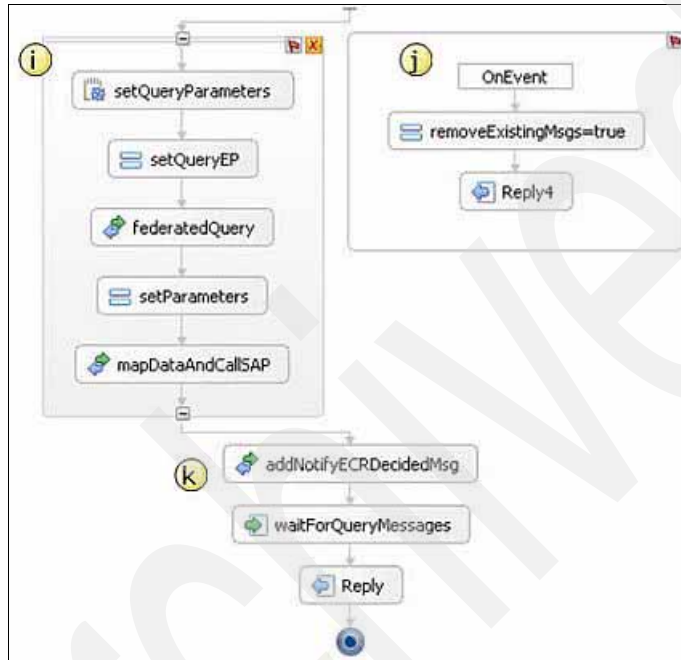


Figure 9-11 ECRProcess BPEL template

The ECR data gets sent to SAP. Figure 9-12 shows the detail property of the mapDataAndCallSAP activity.

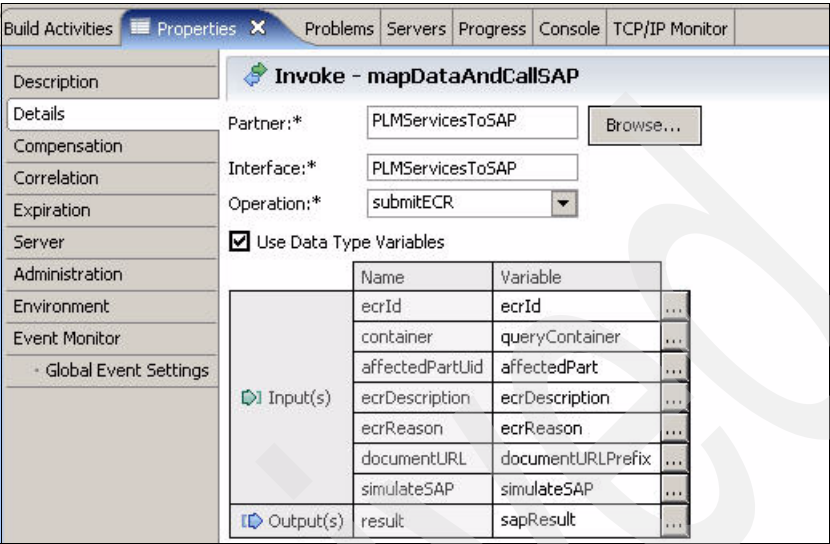


Figure 9-12 mapDataAndCallSAP invoke activity detail

- k. The event handler sets the removeExistingMsgs variable value to be True upon the receipt of a queryMessages event.
- l. The ECRProcess business process invokes the addNotifyECRDecidedMsg method of ECRUtil service. The process waits until the ECR requestor queries the ECR status before entering the final state.

### 9.2.2 The human task components

Both the ECRProcess and the BPELprocess contain two human task activities: one for initial ECR approval, and another for the final approval. In order to work with human tasks, a user interface is required. WebSphere Integration Developer comes with a tool, the BPC Explorer, that allows you to work with human tasks. You use the BPC Explorer in this scenario to handle the ApproveInitialECR and ApproveECR human tasks. As the BPC Explorer is a tool that addresses the needs of an administrator, the user interface is not adequate for a business user.

Usually, a custom user interface for human task handling is developed. WebSphere Integration Developer provides tooling support. For details, refer to WebSphere Integration Developer V6.1 Information Center at the following Web page:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.tel.ui.doc/topics/tgencInt.html>

### 9.2.3 The ECR Web client

A simple JSF-based Web client is provided to test the ECR scenario. This section outlines the ECRClient service module to demonstrate how a client application can be implemented.

In the Business Integration view, double-click **Assembly Diagram** under ECRClient module to open it. See Figure 9-13.

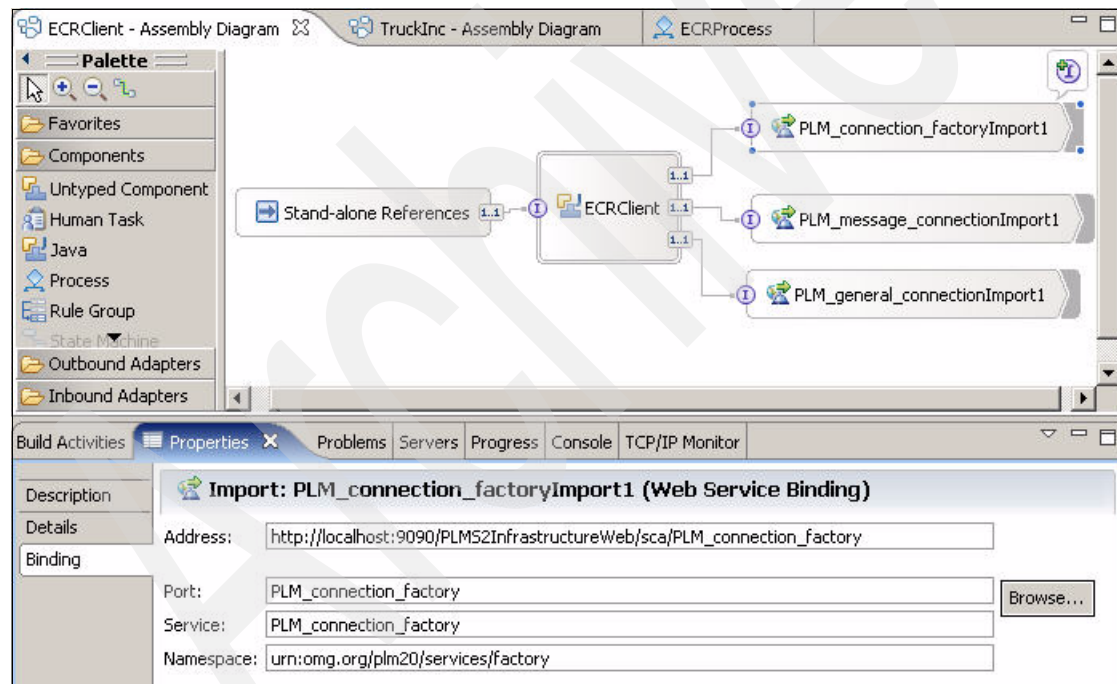


Figure 9-13 ECRClient assembly diagram

The ECRClient component connects to PLM server through the SCA import components: PLM\_connection\_factory, PLM\_message\_connection, and PLM\_general\_connection.

The PLM\_connection\_factory import component is bound to the PLM\_connection\_factory export component of the PLMS2Infrastructure using Web service binding. Review the properties of the PLM\_message\_connection and PLM\_general\_connection to review their binding configurations.

The ECR Web client code is located in the ECRWebClient project. The getPartnerList() method of the EcrHandler.java locates the ECRClient as shown in Example 9-2.

*Example 9-2*

---

```
if (ecrClientIF==null)
    ecrClientIF =
        (ECRClient)ServiceManager.INSTANCE.locateService(
            "ECRClientPartner");
```

---

The ECR Web client locates the ECRClient application through its stand-alone reference component. The connection to the PLM server is handled by the ECRClient application, which makes the client to be de-coupled from the server.

For details, double-click the ECRClient component to review its implementation. In addition have a look at the JSP pages and the truckinc.client-ECRHandler.java code in the ECRWebClient project.

## 9.3 SAP backend integration

This section describes the steps necessary to attach an SAP Enterprise Resource Planning system to the designed Engineering Change Process.

This scenario uses an SAP ECC 5.0 system that includes example data. The data is an SAP IDES (Internet Demonstration and Evaluation System) delivery that is already customized and that includes an example company, customer, and material data.

Using WebSphere Integration Developer, multiple integration modules are developed that interact with each other. The mediation modules include the logic to communicate with the SAP system and are triggered by an integration module that includes a BPEL-based business process. This process orchestrates the necessary backend invocations defining the flow based on the returned results.

The WebSphere Adapter for SAP software is the core component responsible for attaching the SAP system to the ECR process in a service-like manner. The adapter supports the developer to create SCA imports for the different SAP BAPI® modules.

Table 9-1 lists the main SAP business components that are used in this PLM scenario. In the third column, the respective BAPIs or RFC modules that have to be called to execute the desired operation in the SAP system are listed. For example, BAPI\_MATERIAL\_EXISTENCECHECK must be called to check if a specific material exists in the SAP system. For objects that do not have an existence check module in the SAP system, a read call can be used. The table lists all available operations. Although the calls Create, Update, Retrieve, Delete, and Exists are available in this scenario, this Redbooks publication do not use all of them.

Table 9-1 Main SAP business components

Business component	Description	BAPI/RFC
Engineering Change Number	This object is used within the SAP Logistics module to handle various production relevant data objects in terms of change and history tracking.  The integration logic for this object is located in mediation module SAI707SAPECN.	<b>Create</b> CALO_INIT_API CCAP_ECN_CREATE (to be called as a work unit)
		<b>Update</b> CALO_INIT_API CCAP_ECN_MAINTAIN (to be called as a work unit)
		<b>Retrieve</b> CCAP_ECN_HEADER_READ
		<b>Delete</b> CALO_INIT_API CCAP_ECN_HEADER_DELETE (to be called as a work unit)
		<b>Exists</b> n/a

Business component	Description	BAPI/RFC
Material Master	<p>The material master record contains all relevant data of a material. The material record is used by many different SAP components from Logistics to Production till Financing.</p> <p>The integration logic for this object is located in mediation module SAI707SAPMaterialMaster</p>	<b>Create</b> BAPI_MATERIAL_SAVEDATA BAPI_TRANSACTION_COMMIT (to be called as a work unit)
		<b>Update</b> BAPI_MATERIAL_SAVEDATA BAPI_TRANSACTION_COMMIT (to be called as a work unit)
		<b>Retrieve</b> BAPI_MATERIAL_GET_ALL
		<b>Delete</b> n/a
		<b>Exists</b> BAPI_MATERIAL_EXISTENCECHECK
Bill of Material	<p>A BOM can be understood as a list of objects that make up an assembly or product. Within SAP, BOMs are used in different areas, from product costing to production planning. This is why there are different BOM types available, like a material BOM or an order BOM. This Redbooks publication is using a material BOM object.</p> <p>The integration logic for this object is located in mediation module SAI707SAPMatBOM</p>	<b>Create</b> CALO_INIT_API CSAP_MAT_BOM_CREATE (to be called as a work unit)
		<b>Update</b> CALO_INIT_API CSAP_MAT_BOM_MAINTAIN (to be called as a work unit)
		<b>Retrieve</b> CSAP_MAT_BOM_READ
		<b>Delete</b> CALO_INIT_API CSAP_MAT_BOM_DELETE (to be called as a work unit)
		<b>Exists</b> n/a

Business component	Description	BAPI/RFC
Document	<p>The SAP system includes a document management capability to enable the creation of documents that can be used to describe processes in detail.</p> <p>The relevant document type for the scenario in this Redbooks publication is DES (Design / Engineering).</p> <p>This type represents a technical object that may include links to other objects like materials or BOMs. Additionally the document object can contain one or more Web links pointing to Web resources.</p> <p>The integration logic for this object is located in mediation module SAI707SAPDocument</p>	<b>Create</b> BAPI_DOCUMENT_CREATE2 BAPI_TRANSACTION_COMMIT (to be called as a work unit)
		<b>Update</b> BAPI_DOCUMENT_CHANGE2 BAPI_TRANSACTION_COMMIT (to be called as a work unit)
		<b>Retrieve</b> BAPI_DOCUMENT_GETDETAIL2
		<b>Delete</b> BAPI_DOCUMENT_DELETE BAPI_TRANSACTION_COMMIT (to be called as a work unit)
		<b>Exists</b> BAPI_DOCUMENT_EXISTENCECHECK

### 9.3.1 Configuration

In this section, we perform configurations specific to the SAP system.

#### SAP ERP customization

Within this PLM scenario, specific documents are created in the SAP system that are linked to material master objects. Additionally, materials, BOMs, and engineering change number objects are created. The selections in the steps below are required to ensure that this scenario can be executed successfully.

Follow these steps to check if your SAP system is set up correctly.

1. Log in to the SAP system and enter transaction code spro, as shown in Figure 9-14.

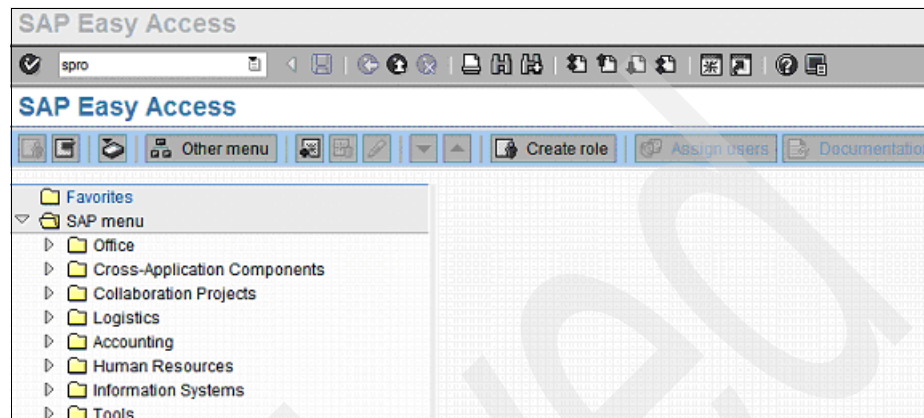


Figure 9-14 SAP transaction

2. Click the SAP Reference IMG tab as shown in Figure 9-15, or press F5.

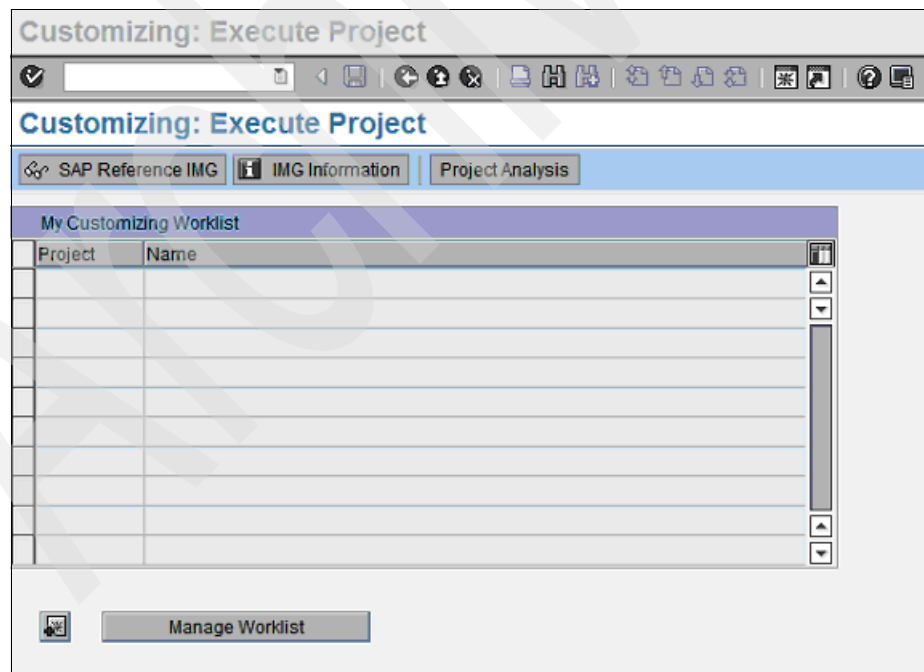


Figure 9-15 SAP customizing



- Go to **Cross application component** → **Document management** → **Control Data** and select **Define document types**. See Figure 9-16.

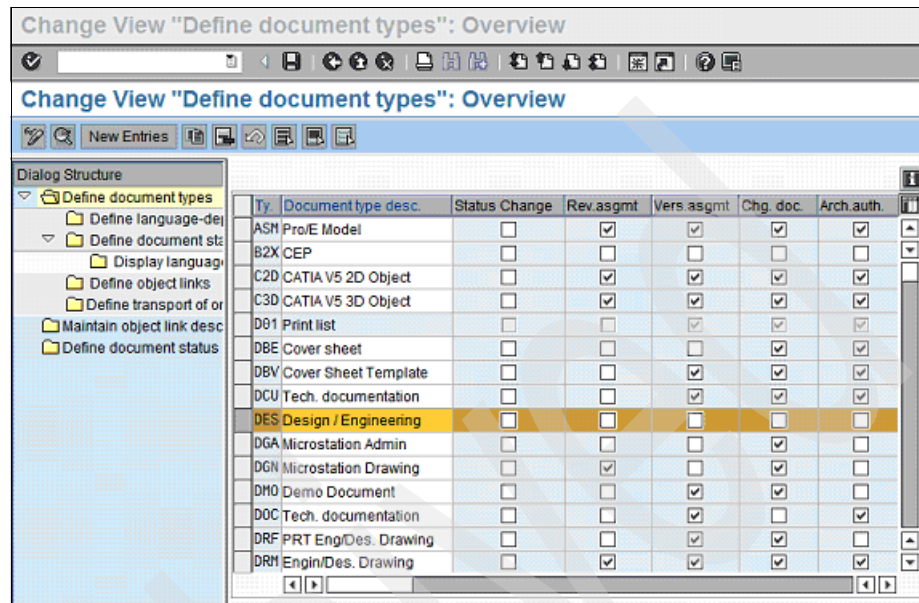


Figure 9-16 SAP document type selection

- Clear all the indicators for document type DES. Save these settings.
- When a change request window is displayed, select an appropriate change request number and save the change request.
- Select **Document type DES** then click **Define object links**.

7. Select the row displaying object DES with MARA and enter the screen number 201 as in Figure 9-17. Save these settings.

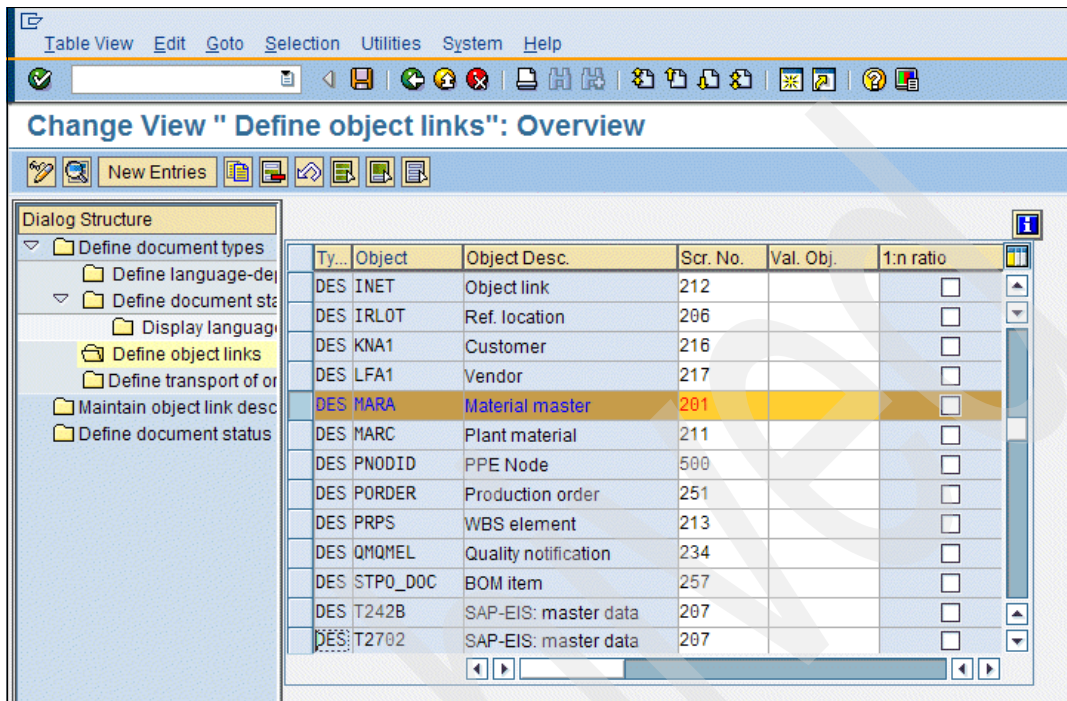


Figure 9-17 SAP document type DES object link

8. When a change request window is displayed, select an appropriate change request number and save the change request.
9. Click the **Define document status** entry in the left tree.

10. Select the row for document status IA (Figure 9-18) and ensure that the check marks circled in Figure 9-18 are set. Save these settings.

**Change View " Define document status": Details**

Dialog Structure

- Define document types
  - Define language-dep
  - Define document st
  - Display language
  - Define object links
  - Define transport of o
  - Maintain object link desc
  - Define document status

Document Type: DES

Document Status: IA

Status: IW

Status text: In work

Attributes

- ☒ Object Check
- ☒ Release Flag
- ☐ Content version
- ☐ Check in
- ☐ Complete for ECM
- ☐ Distr. lock
- ☐ Check-In Required

Fid sel. -

Status type

Prev. 1

Prev. 2

Prev. 3

Prev. 4

Prev. 5

Prev. 6

Workflow Task

Object Type

Object ID

Program exit

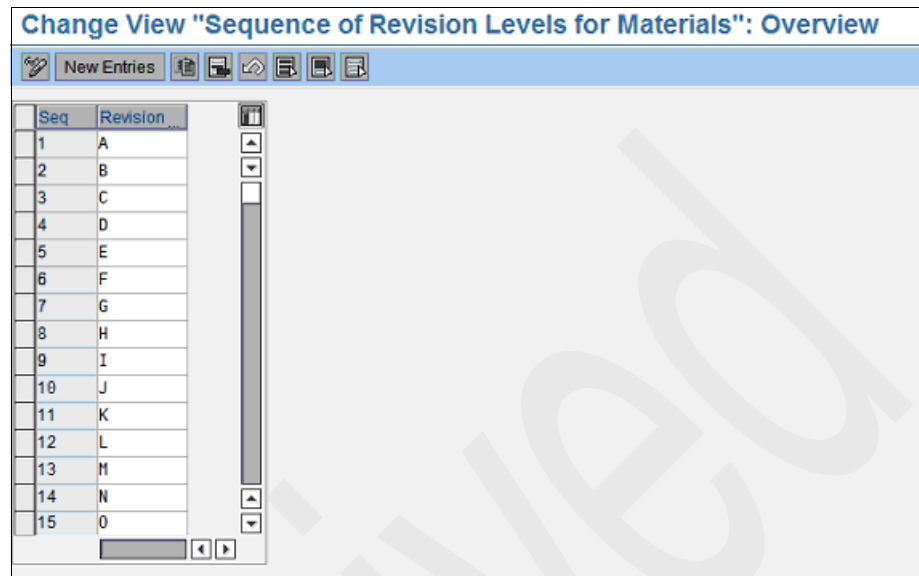
Form routine

SignStrat

Figure 9-18 SAP document type DES document status IA

11. When a change request window is displayed, select an appropriate change request number and save the change request.
12. Return to the SAP Reference IMG tab you accessed in step 2 of this procedure (Figure 9-15 on page 274). This may require logging in to the SAP system again, as done in step 1.
13. Go to **Logistic general** → **Engineering Change Management** → **Revision Levels** and select **Define Revision Levels for Materials**.

14. Verify that the sequences are maintained as shown in Figure 9-19.



Change View "Sequence of Revision Levels for Materials": Overview

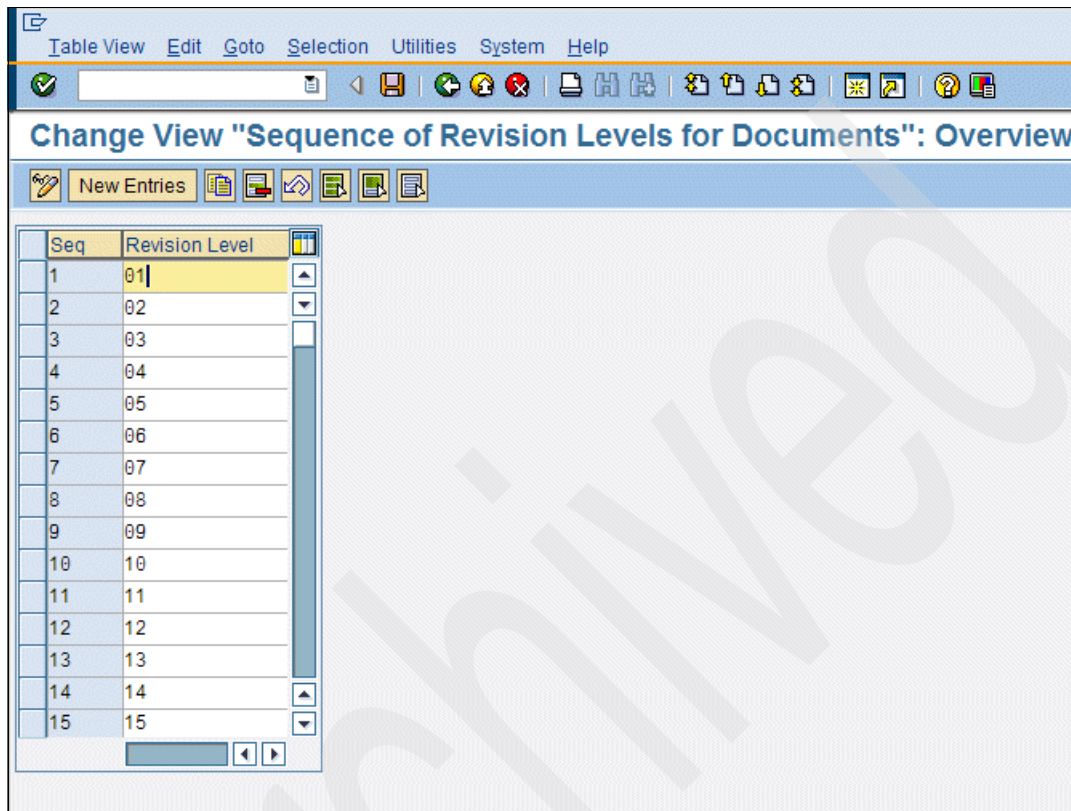
New Entries

Seq	Revision
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O

Figure 9-19 SAP Revision Levels for Materials

15. Go to **Logistic general** → **Engineering Change Management** → **Revision Levels** and select **Define Revision Levels for Documents**.

16. Verify that the sequences are maintained as shown in Figure 9-20. Save these settings.



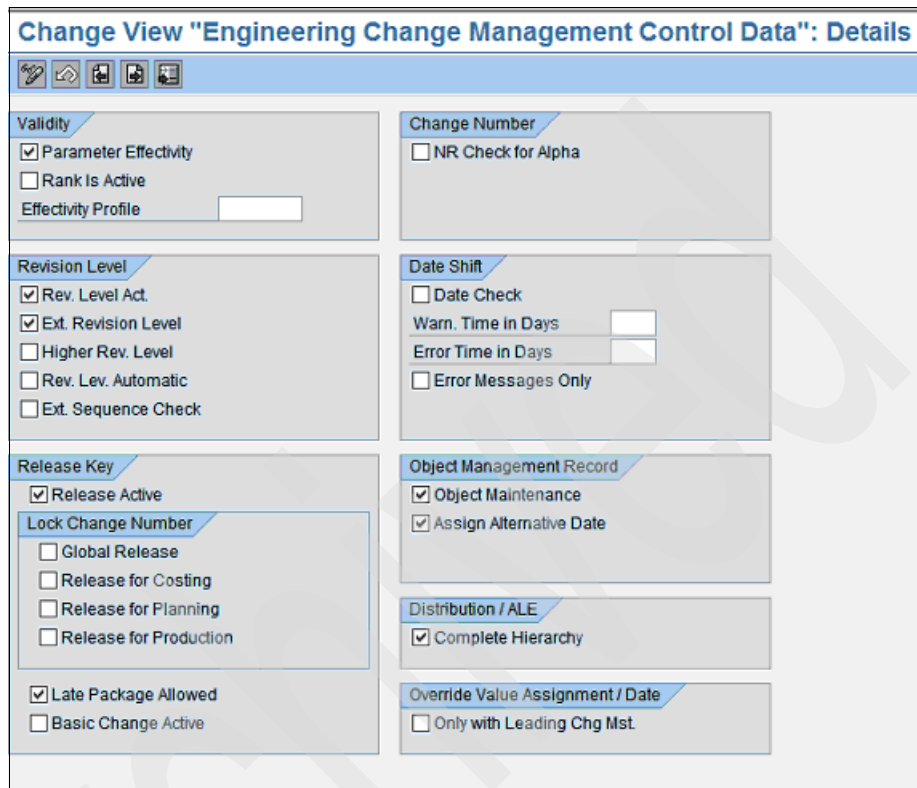
The screenshot displays the SAP Change View 'Sequence of Revision Levels for Documents': Overview. The interface includes a menu bar (Table View, Edit, Goto, Selection, Utilities, System, Help) and a toolbar with various icons. Below the title bar, there is a 'New Entries' button and a list of icons. The main area contains a table with two columns: 'Seq' and 'Revision Level'. The table lists 15 entries, with the first row (Seq 1, Revision Level 01) highlighted in yellow. The table has a vertical scrollbar on the right and a horizontal scrollbar at the bottom.

Seq	Revision Level
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
10	10
11	11
12	12
13	13
14	14
15	15

Figure 9-20 SAP Revision Levels for Documents

17. When a change request window is displayed, select an appropriate change request number and save the change request.
18. Go to **Logistic-general** → **Engineering Change Management** and select **Set Control Data**.

19. Verify that the settings are maintained as shown in Figure 9-21. Save these settings.



**Change View "Engineering Change Management Control Data": Details**

**Validity**

- ☒ Parameter Effectivity
- ☐ Rank Is Active
- Effectivity Profile

**Change Number**

- ☐ NR Check for Alpha

**Revision Level**

- ☒ Rev. Level Act.
- ☒ Ext. Revision Level
- ☐ Higher Rev. Level
- ☐ Rev. Lev. Automatic
- ☐ Ext. Sequence Check

**Date Shift**

- ☐ Date Check
- Warn. Time in Days
- Error Time in Days
- ☐ Error Messages Only

**Release Key**

- ☒ Release Active
- Lock Change Number**
  - ☐ Global Release
  - ☐ Release for Costing
  - ☐ Release for Planning
  - ☐ Release for Production
- ☒ Late Package Allowed
- ☐ Basic Change Active

**Object Management Record**

- ☒ Object Maintenance
- ☒ Assign Alternative Date

**Distribution / ALE**

- ☒ Complete Hierarchy

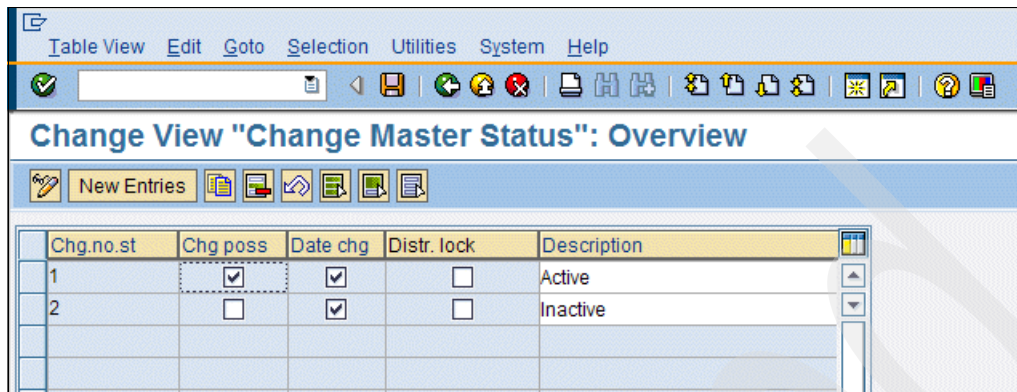
**Override Value Assignment / Date**

- ☐ Only with Leading Chg Mst.

Figure 9-21 SAP Engineering Change Management Control Data

20. When a change request window is displayed, select an appropriate change request number and save the change request.
21. Go to **Logistic-general → Engineering Change Management** and select **Define Statuses of Change Master Records**.

22. Verify that the settings are maintained as in Figure 9-22. Save these settings.



The screenshot shows the SAP Engineering Change Management (ECM) Change Master Record. The title bar reads "Change View 'Change Master Status': Overview". Below the title bar is a menu bar with "Table View", "Edit", "Goto", "Selection", "Utilities", "System", and "Help". A toolbar contains various icons for actions like save, delete, and print. Below the toolbar is a sub-toolbar with "New Entries" and other icons. The main area is a table with the following columns: "Chg.no.st", "Chg poss", "Date chg", "Distr. lock", and "Description". The table contains two rows: Row 1 has "1" in "Chg.no.st", a checked checkbox in "Chg poss", a checked checkbox in "Date chg", an unchecked checkbox in "Distr. lock", and "Active" in "Description". Row 2 has "2" in "Chg.no.st", an unchecked checkbox in "Chg poss", a checked checkbox in "Date chg", an unchecked checkbox in "Distr. lock", and "Inactive" in "Description".

Chg.no.st	Chg poss	Date chg	Distr. lock	Description
1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Active
2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Inactive

Figure 9-22 SAP Engineering Change Management Change Master Record

23. When a change request window is displayed, select an appropriate change request number and save the change request.
24. Go to **Production** → **Basic data** → **Bill of material** → **Control Data for Bills of Material** and select **Define Modification Parameters**.



25. Verify that the settings are maintained as in Figure 9-23. Save these settings.

**Change View "Modification Parameters for BOMs": Details**

**Validity**

- ☐ Low Date
- ☒ BOM validity maint.
- ☒ EC management active
- ☐ History reqmt
- Hist. reqmt. variant
- ☐ Sub-item documentn
- ☐ Header
- Handling Del. Flag ☐

**Technical Type**

- ☐ Mult. BOM inactive
- ☐ Variant BOM inactive

**BOM Item**

- Item explosion type ☐
- Desc. var. size item ☐
- ☐ Rptd itm effctvty via external item ID

**General Settings**

- Unit 'piece'
- ☒ CAD active

Figure 9-23 SAP Modification Parameters for BOMs

26. When a change request window is displayed, select an appropriate change request number and save the change request.

## WebSphere Process Server configuration

The developed artifact is deployed on a WebSphere Process Server instance. The modules expect the WebSphere Adapter for SAP software to be installed as a standalone Resource Adapter. Within this standalone adapter a Connection Factory is defined that points to the target SAP that was customized in the prior section. A standalone adapter has the advantage to administer the connection



parameters centrally. It is not necessary here to include the adapter in each integration module. The following sections describe the following procedures using the Web-based administration console of WebSphere Process Server:

- ▶ “Installing the WebSphere Adapter for SAP software standalone” on page 283
- ▶ “Creating a J2C authentication alias” on page 284
- ▶ “Defining a J2C connection factory” on page 287

### ***Installing the WebSphere Adapter for SAP software standalone***

Perform the following steps to install the WebSphere Adapter for SAP software standalone using the Web-based administration console of WebSphere Process Server.

1. Open the Web-based admin console using a Web browser. For example:  
`https://localhost:9043/ibm/console/`
2. Enter the appropriate credentials to login as user with administrative rights.
3. Click **Resources** → **Resource Adapters** → **Resource adapters**.
4. From the Resource adapters page (Figure 9-24), click **Install RAR**.

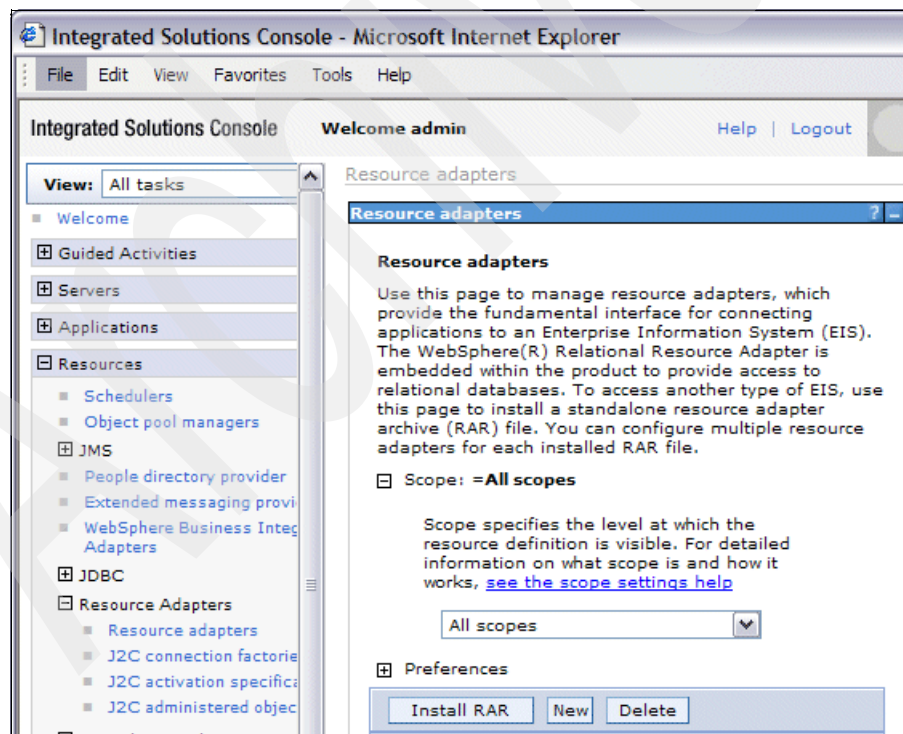


Figure 9-24 WebSphere Process Server Admin console: Install resource adapter

5. From the Install RAR file page, click **Browse**, and navigate to the appropriate RAR file for your adapter.

The SAP adapter is typically located in the following path:

```
WID_installation_directory\ResourceAdapters\SAP_6.1.0.0_IF03\deploy\
CWYAP_SAPAdapter.rar
```

6. Click **Next**.
7. (Optional) From the Resource adapters page, change the name of the adapter and add a description.
8. Click **OK**.
9. Click **Save** in the Messages box to make your changes persistent.

The WebSphere Adapter for SAP software requires SAP native libraries to work properly. These libraries have to be available on the target operating system and accessible by the WebSphere Process Server. One option is to copy the libraries into the WebSphere Process Server installation directory.

The following procedure details how this can be done for a Windows-based installation.

1. Obtain the SAP Java Connector (JCo) libraries from SAP.
2. Copy the files `librfc32.dll` & `sapjcorfc.dll` into the `<WPS_INST>\bin` directory.
3. Copy the files `sapjco.jar` into the `<WPS_INST>\lib` directory.
4. Restart the WebSphere Process Server to finish the installation.

### ***Creating a J2C authentication alias***

The next step is to create a J2C authentication alias. Perform the following steps:

1. Open the Web-based admin console using a Web browser. For example:  
`https://localhost:9043/ibm/console/`
2. Enter the appropriate credentials to login as user with administrative rights.
3. Click **Security** → **Secure administration, applications, and infrastructure**.

4. In the Authentication window (Figure 9-25), click **Java Authentication and Authorization Service** → **J2C authentication data**.

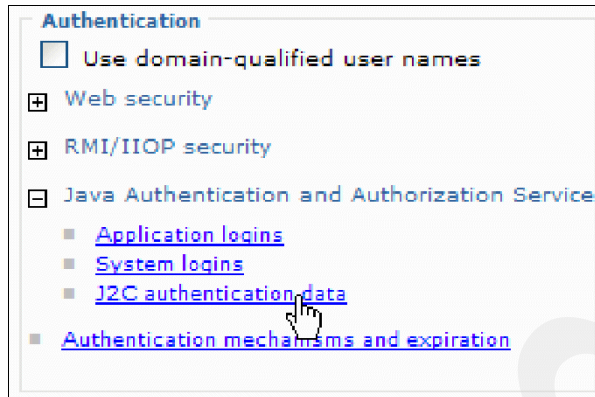


Figure 9-25 WebSphere Process Server Admin console: J2C authentication data

5. In the list of J2C authentication aliases, click **New**.
6. In the Configuration tab, type the name of the authentication alias in the Alias field.

7. Type the user ID and password (Figure 9-26) that are required to establish a connection to the SAP server. You have the option of entering a description of the alias. Click **OK**.

The screenshot shows a configuration window titled "Secure administration, applications, and infrastructure". The breadcrumb path is "Secure administration, applications, and infrastructure > JAAS - J2C authentication data > widNode/wccpc11800". Below the breadcrumb, it says "Specifies a list of user identities and passwords for Java(TM) 2 connector security to use." The "Configuration" tab is selected. Under "General Properties", there are four fields: "Alias" (widNode/wccpc11800), "User ID" (ktw), "Password" (masked with dots), and "Description" (Admin User in SAP System). At the bottom are buttons for "Apply", "OK", "Reset", and "Cancel".

Figure 9-26 J2C authentication entry for SAP server

The newly created alias is displayed. Note the full name of the alias as it is needed in subsequent configuration windows.

8. Click **Save**, then click **Save in the Messages box** at the top of the page to make your changes persistent.

At this point the J2C authentication alias is created and can be referenced by the adapter during runtime or the external service wizard during development.

## Defining a J2C connection factory

The next step is to define a J2C connection factory within the standalone adapter. Perform the following steps:

1. Open the Web-based admin console using a Web browser. For example:  
`https://localhost:9043/ibm/console/`
2. Enter the appropriate credentials to login as user with administrative rights.
3. Click **Resources** → **Resource Adapters** → **Resource adapters**.
4. From the Resource adapters page, click **IBM WebSphere Adapter for SAP Software**.
5. From the Additional Properties list, click **J2C connection factories**. See Figure 9-27.

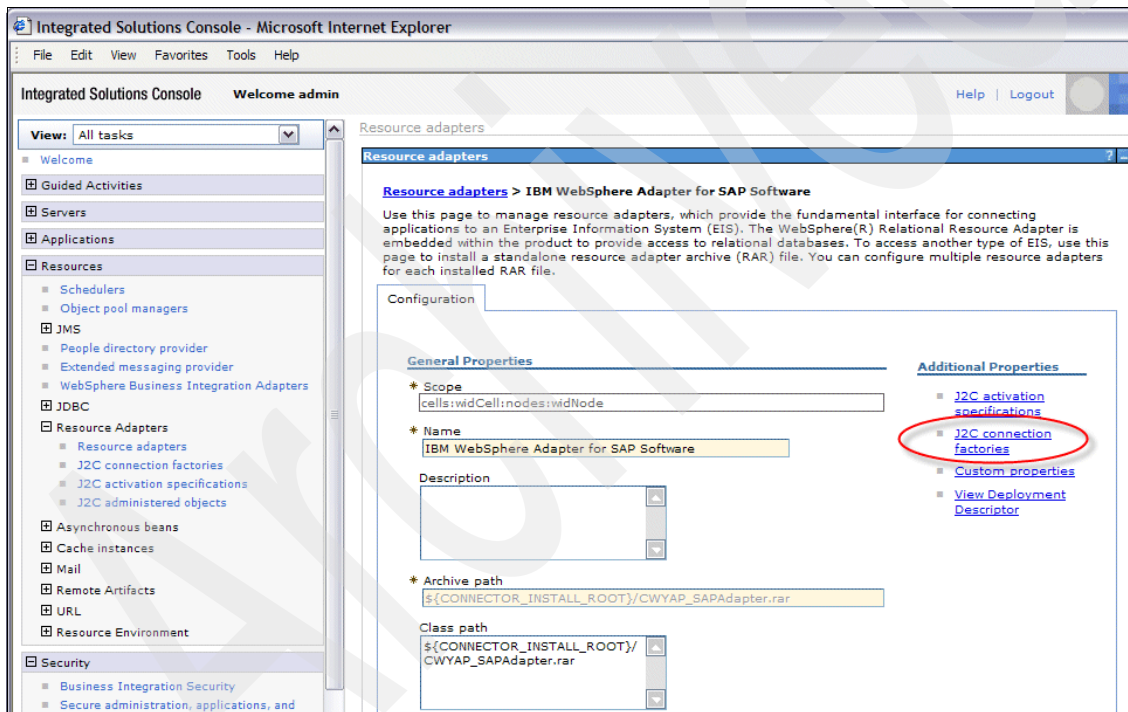


Figure 9-27 J2C connection factory

6. Click **New**.

7. In the General Properties section of the Configuration tab, enter a name for the connection factory. For example, you could enter `wccpc11800CF`. See Figure 9-28 on page 289.
8. Enter a value for JNDI name. For example, you could enter `com/eis/wccpc11800AdapterCF`.
9. Select an authentication alias from the Component-managed authentication alias list. For example, use the alias created in the prior section `widNode/wccpc11800`. Click **OK**.
10. Click **Save in the Messages box** at the top of the page. The newly created connection factory is displayed.
11. Click the newly created connection factory to adjust the custom properties.

Resource adapters

Resource adapters > IBM WebSphere Adapter for SAP Software > J2C connection factories > wccpc11800CF

Use this page to create a connection factory for use with the resource adapter. The connection factory is a collection of configuration values that define a WebSphere(R) Application Server connection to your Enterprise Information System (EIS). The connection pool manager uses these properties as directions for allocating connections during runtime. You can configure multiple connection factories for each resource adapter.

Configuration

General Properties

\* Scope  
cells:widCell:nodes:widNode

\* Provider  
IBM WebSphere Adapter for SAP Software

\* Name  
wccpc11800CF

JNDI name  
com/eis/wccpc11800AdapterCF

Description  
Connection factory pointing to wccpc11 SAP system Client 800

\* Connection factory interface  
javax.resource.cci.ConnectionFactory

Category

Component-managed authentication alias  
Component-managed authentication alias  
widNode/wccpc11800

Container-managed authentication  
Container-managed authentication alias (deprecated in V6.0, use resource reference authentication settings instead)  
(none)

Authentication preference (deprecated in V6.0, use resource reference authentication settings instead)  
None

Mapping-configuration alias (deprecated in V6.0, use resource reference authentication settings instead)  
DefaultPrincipalMapping

Additional Properties

Connection pool properties

Advanced connection factory properties

Custom properties

Related Items

Apply OK Reset Cancel

Figure 9-28 J2C connection factory general properties

12. From the Additional Properties list, click **Custom properties**. See Figure 9-29.

**Resource adapters**

Messages  
 Your preferences have been changed.

[Resource adapters](#) > [IBM WebSphere Adapter for SAP Software](#) > [J2C connection factories](#) > [wccpc11800CF](#) > Custom properties

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

Preferences

Name ▾	Value ▾	Description ▾	Required
<a href="#">ignoreBAPIReturn</a>	false		false
<a href="#">SystemNumber</a>	00		false
<a href="#">adapterID</a>	ResourceAdapter		false
<a href="#">Language</a>	E		false
<a href="#">ApplicationServerHost</a>	wccpc11.pqh.ibm.com		false
<a href="#">SAPSystemID</a>			false
<a href="#">partnerCharset</a>			false
<a href="#">traceFileSize</a>	0		false
<a href="#">RFCTraceOn</a>	false		false
<a href="#">logFilename</a>			false
<a href="#">RFCTraceLevel</a>			false
<a href="#">logonGroup</a>			false
<a href="#">logFileSize</a>	0		false
<a href="#">RFCTracePath</a>			false
<a href="#">userName</a>			false
<a href="#">traceFilename</a>			false
<a href="#">ABAPDebug</a>	false		false
<a href="#">GatewayHost</a>			false
<a href="#">MessageServerHost</a>			false
<a href="#">GatewayService</a>			false
<a href="#">logNumberOfFiles</a>	1		false
<a href="#">codepage</a>	1100		false
<a href="#">traceNumberOfFiles</a>	1		false
<a href="#">Password</a>			false
<a href="#">biDiContextEIS</a>			false
<a href="#">Client</a>	800		false
Total 26			

Figure 9-29 J2C connection factory custom properties



**Note:** Custom properties are those J2C connection factory properties that are unique to Adapter for SAP Software.

13. For each property you want to change, perform the following steps:

- a. Click the name of the property.
- b. Change the contents of the Value field or type a value, if the field is empty.
- c. Click **OK**.

In this scenario, the following properties have been adjusted:

- ApplicationServerHost
- SystemNumber
- Client
- Language

14. After you have finished setting properties, click **Apply**.

15. Click **Save in the Messages box** at the top of the window.

At this point, all WebSphere Process Server-specific configurations are complete. This is a prerequisite for the following sections. The WebSphere Adapter for SAP software is installed standalone on the WebSphere Process Server and can be used by multiple integration modules. Additionally, a J2C connection factory has been created that points to the SAP server plus. A J2C authentication alias that contains the regarding credentials for the SAP server has also been created.

### 9.3.2 Development

This section describes in detail the various steps to develop the integration and mediation modules that make up this scenario.

#### Scenario definition

The SAP integration scenario consists of four mediation modules, one integration module and one common library.

The mediation modules include the imports that are created using the WebSphere Adapter for SAP software and that represent the required BAPI and RFC calls. Additionally, in each mediation module, a specific mediation flow is used to map the SAP system (whereas, a fairly complex interface is mapped to a generic and simpler interface).

The integration module points to the four mediation modules using SCA binding. These imports are referenced and invoked by a BPEL process named TransferECRtoERPSystem that orchestrates the order in which different SAP

system invocations are executed. Beside the BPEL process, a Java component is included that is responsible to map the data object derived from the PLM container to an array format required by the BPEL process.

The common integration library includes the simplified interfaces provided by the mediation modules through SCA exports. The library also contains the interface describing the TransferECRtoERPSytem BPEL process and the PLM services interface.

### **Describe the common integration library**

The common integration library is named ITSOPLMLibrary and contains following interfaces:

- ▶ BillOfMaterialInterface
- ▶ DocumentSystemInterface
- ▶ EngineeringChangeManagementInterface
- ▶ MaterialMasterInterface
- ▶ PLMERPInterface
- ▶ PLMServicesToSAP

The interfaces use business objects that are also included in the common integration library. These business objects represent a generic representation of exchanged artifacts like parts and documents. The following business objects are included in the common integration library:

- ▶ Attachment.xsd
- ▶ BOM.xsd
- ▶ BOMBG.xsd
- ▶ BOMItem.xsd
- ▶ ComponentLink.xsd
- ▶ Document.xsd
- ▶ DocumentBG.xsd
- ▶ ECN.xsd
- ▶ ECNBG.xsd
- ▶ Effectivity.xsd
- ▶ Message.xsd
- ▶ MultiBOMs.xsd
- ▶ MultiDocuments.xsd
- ▶ MultiParts.xsd
- ▶ Part.xsd
- ▶ PartBG.xsd
- ▶ Properties.xsd
- ▶ Result.xsd
- ▶ ResultBG.xsd
- ▶ StandardFaultType.xsd

Figure 9-30 shows a window of the Engineering Change Management interface.

Operations		
Operations and their parameters		
	Name	Type
createECN		
Input(s)	ecn	ECN BG
Output(s)	result	Result BG
Fault	ApplicationFault	StandardFaultType
changeECN		
Input(s)	ecn	ECN BG
Output(s)	result	Result BG
Fault	ApplicationFault	StandardFaultType
readECN		
Input(s)	ecnnumber	string
Output(s)	result	ECN BG
Fault	ApplicationFault	StandardFaultType
deleteECN		
Input(s)	ecnnumber	string
Output(s)	result	Result BG
Fault	ApplicationFault	StandardFaultType

Figure 9-30 Engineering Change Management interface Operations window

Not all operations displayed in this interface are used within this scenario.

Figure 9-31 shows a Material Master interface window.

Operations		
Operations and their parameters		
	Name	Type
createMaterial		
Input(s)	material	PartBG
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
changeMaterial		
Input(s)	material	PartBG
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
readMaterial		
Input(s)	materialID	string
	plant	string
Output(s)	material	PartBG
Fault	ApplicationFault	StandardFaultType
isExisting		
Input(s)	materialID	string
	plant	string
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
deleteMaterial		
Input(s)	materialID	string
	plant	string
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType

Figure 9-31 Material Master interface Operations window

Not all operations displayed in this interface are used within this scenario.

Figure 9-32 shows a Bill of Material interface window.

<b>Operations</b>		
Operations and their parameters		
	Name	Type
▼ createBOM		
Input(s)	bom	BOMBG
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
▼ changeBOM		
Input(s)	bom	BOMBG
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
▼ retrieveBOM		
Input(s)	bomID	string
	plant	string
	usage	string
Output(s)	result	BOMBG
Fault	ApplicationFault	StandardFaultType
▼ deletBOM		
Input(s)	bomID	string
	plant	string
	usage	string
	changenumber	string
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType

Figure 9-32 Bill of Material interface Operations window

Not all operations displayed in this interface are used within this scenario.

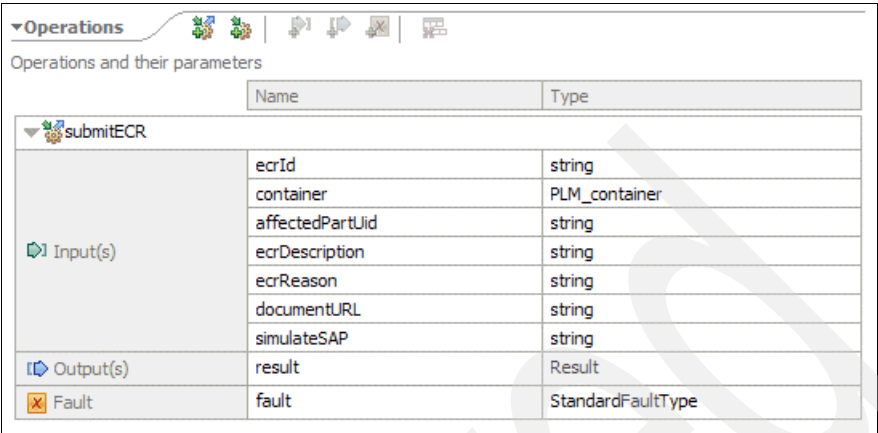
Figure 9-33 shows a Document interface window.

Operations		
Operations and their parameters		
	Name	Type
▼ createDocument		
Input(s)	document	DocumentBG
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
▼ changeDocument		
Input(s)	document	DocumentBG
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
▼ retrieveDocument		
Input(s)	documentnumber	string
	documenttype	string
	documentpart	string
	documentversion	string
Output(s)	document	DocumentBG
Fault	ApplicationFault	StandardFaultType
▼ deleteDocument		
Input(s)	documentnumber	string
	documenttype	string
	documentpart	string
	documentversion	string
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType
▼ isExisting		
Input(s)	documentnumber	string
	documenttype	string
	documentpart	string
	documentversion	string
Output(s)	result	boolean
Fault	ApplicationFault	StandardFaultType

Figure 9-33 Document interface Operations window

Not all operations displayed in this interface are used within this scenario.

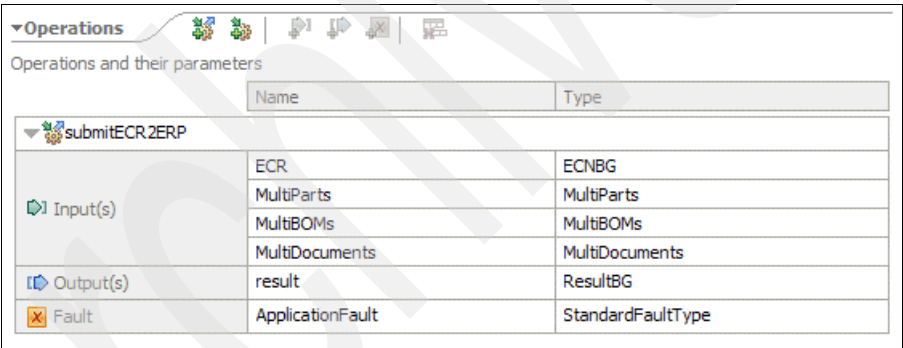
Figure 9-34 shows a PLM services to SAP interface window.



	Name	Type
submitECR		
Input(s)	ecrId	string
	container	PLM_container
	affectedPartUId	string
	ecrDescription	string
	ecrReason	string
	documentURL	string
	simulateSAP	string
Output(s)	result	Result
Fault	fault	StandardFaultType

Figure 9-34 PLM services to SAP interface Operationswindow

Figure 9-35 shows a window of the PLM ERP interface window.



	Name	Type
submitECR2ERP		
Input(s)	ECR	ECN BG
	MultiParts	MultiParts
	MultiBOMs	MultiBOMs
	MultiDocuments	MultiDocuments
Output(s)	result	ResultBG
Fault	ApplicationFault	StandardFaultType

Figure 9-35 PLM ERP interface Operations window

The common integration library is referenced as a dependency by the four mediation modules and the integration module.

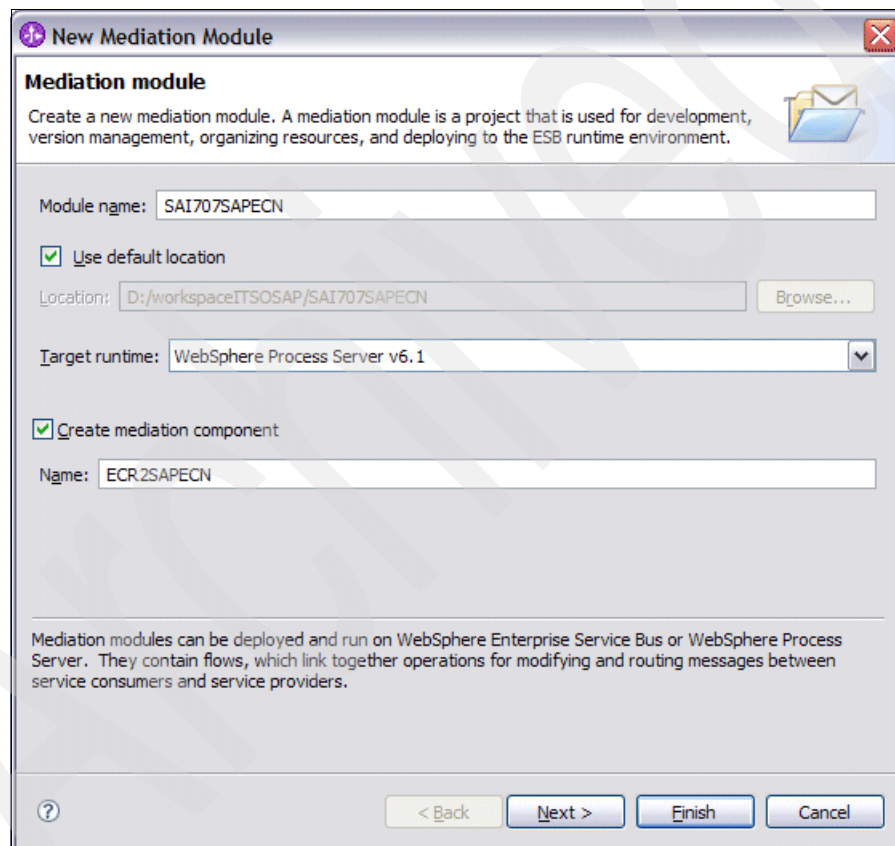
## Develop ECM mediation module

This mediation module is responsible to create a Engineering Change Number in the SAP system. In this scenario, only the Create operation is described.

### ***Creating the ECM mediation module***

Complete the following steps to create the mediation module:

1. Go to **File → New → Mediation Module**. Name the mediation module SAI707SAPECN (see Figure 9-36) and the mediation component ECR2SAPECN. Select **Next**.



**New Mediation Module**

**Mediation module**

Create a new mediation module. A mediation module is a project that is used for development, version management, organizing resources, and deploying to the ESB runtime environment.

Module name: SAI707SAPECN

☒ Use default location

Location: D:/workspace/ITSOSAP/SAI707SAPECN Browse...

Target runtime: WebSphere Process Server v6.1

☒ Create mediation component

Name: ECR2SAPECN

Mediation modules can be deployed and run on WebSphere Enterprise Service Bus or WebSphere Process Server. They contain flows, which link together operations for modifying and routing messages between service consumers and service providers.

? < Back Next > Finish Cancel

Figure 9-36 Create mediation module SAI707SAPECN



2. Select the check box to reference the ITSOPMLLibrary. Click **Finish**.
3. In the Assembly Diagram, choose the SAP Outbound adapter in the Palette and place it on a free area on the canvas. See Figure 9-37.

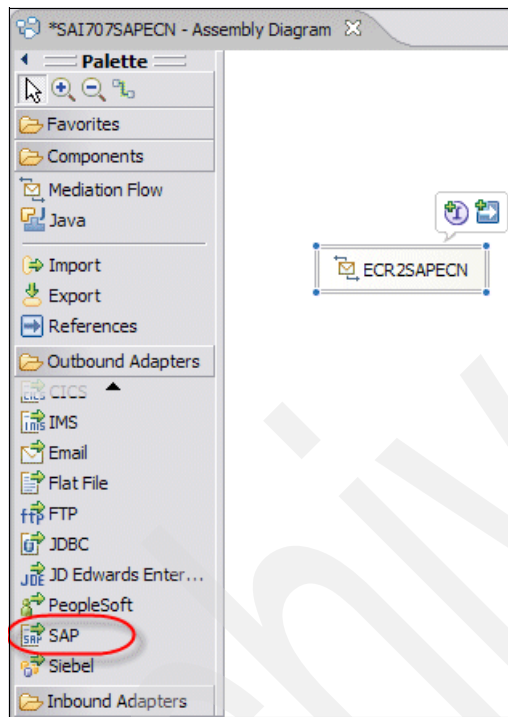


Figure 9-37 Select SAP outbound adapter

4. Enter the connection data for the SAP system and choose **BAPI work unit** (Figure 9-38). Click **Next**.

**Note:** Depending on if the WebSphere Integration Developer tooling is used the first time with Adapters the next windows may differ. In this scenario, select the IBM WebSphere Adapter for SAP software. The adapter with transactional support is not required here.

The screenshot shows a window titled 'External Service' with a sub-header 'Discovery Configuration'. Below the sub-header is the instruction 'Specify properties to begin discovery.' The main section is 'Connection properties' with a sub-section 'SAP system connection information'. This section contains several input fields: 'Host name' with the value 'wccpc11.pgh.ibm.com', 'System number' with '00', 'Client' with '800', 'Language code' with 'EN (English)' and a 'Select...' button, 'Code page' with '1100' and a dropdown arrow, 'User name' with 'ktw', 'Password' with masked characters '\*\*\*\*\*', and 'SAP interface name' with 'BAPI work unit' and a dropdown arrow. There is an 'Advanced >>' button and a checkbox labeled 'Change logging properties for wizard' which is currently unchecked. At the bottom are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Figure 9-38 SAP adapter Discovery configuration

5. Select the RFC tree (Figure 9-39) and click the Filter icon.

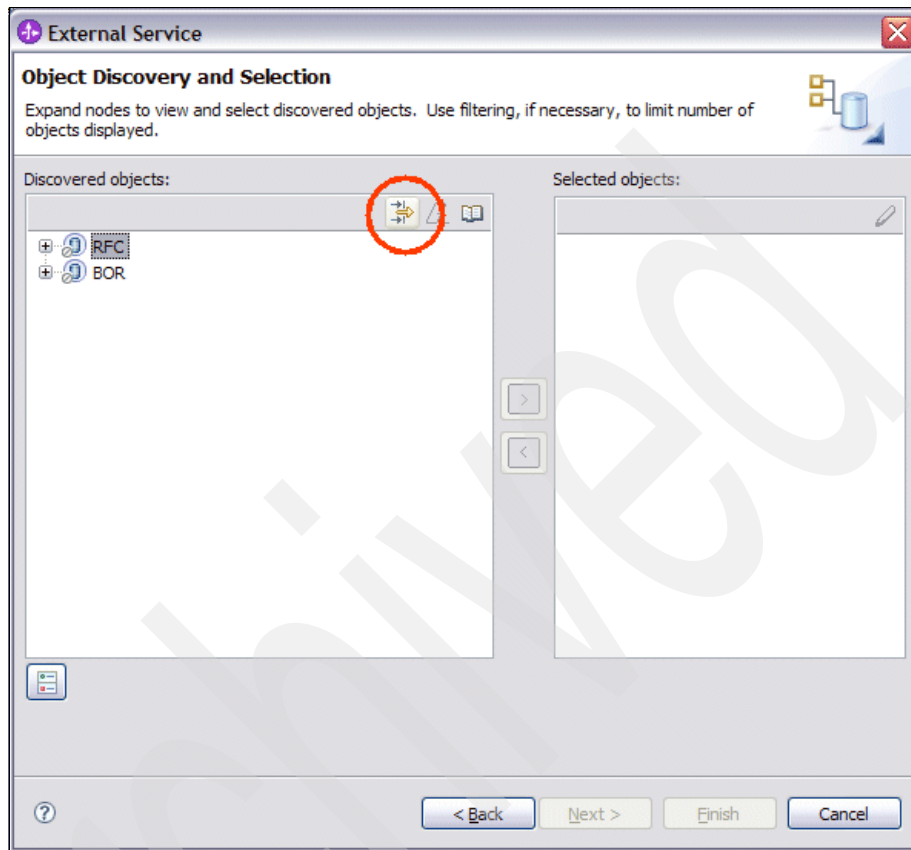


Figure 9-39 Object Discovery and Selection

6. Enter CALO\* in the find field (Figure 9-40), click **OK**.

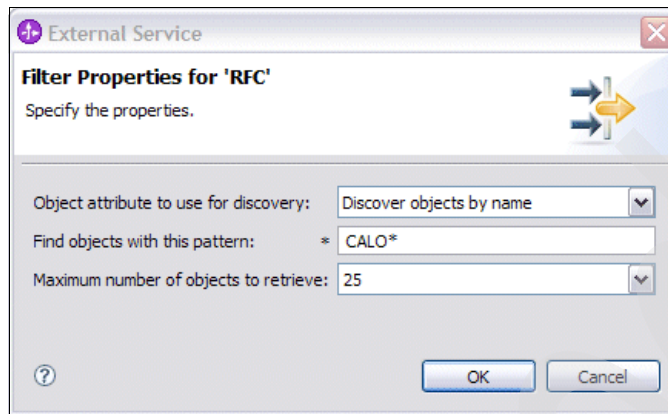


Figure 9-40 Filter properties for RFC

7. Choose the list entry CALO\_INIT\_API (Figure 9-41) and click > to select this function module.

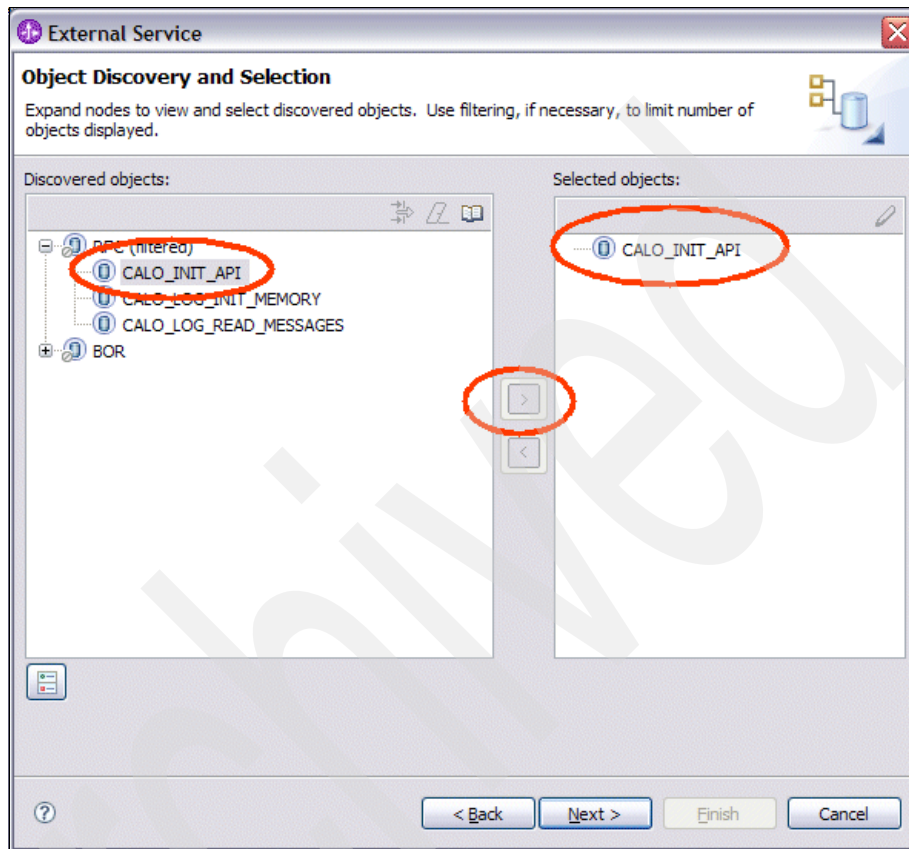


Figure 9-41 Select first function module

8. In the External Service pop-up window, select **Use SAP field names to generate attributes names**. See Figure 9-42.

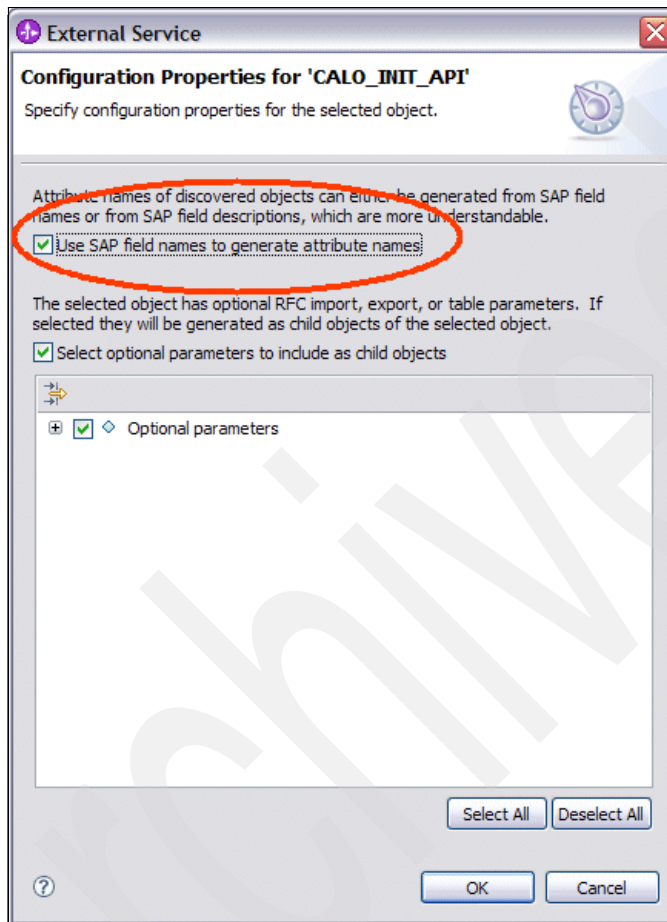


Figure 9-42 Use SAP field names as attribute names

9. Change the filter option to search for the pattern CCAP\*.
10. Select the list entry CCAP\_ECN\_CREATE (Figure 9-43) and add it to the selected object list.

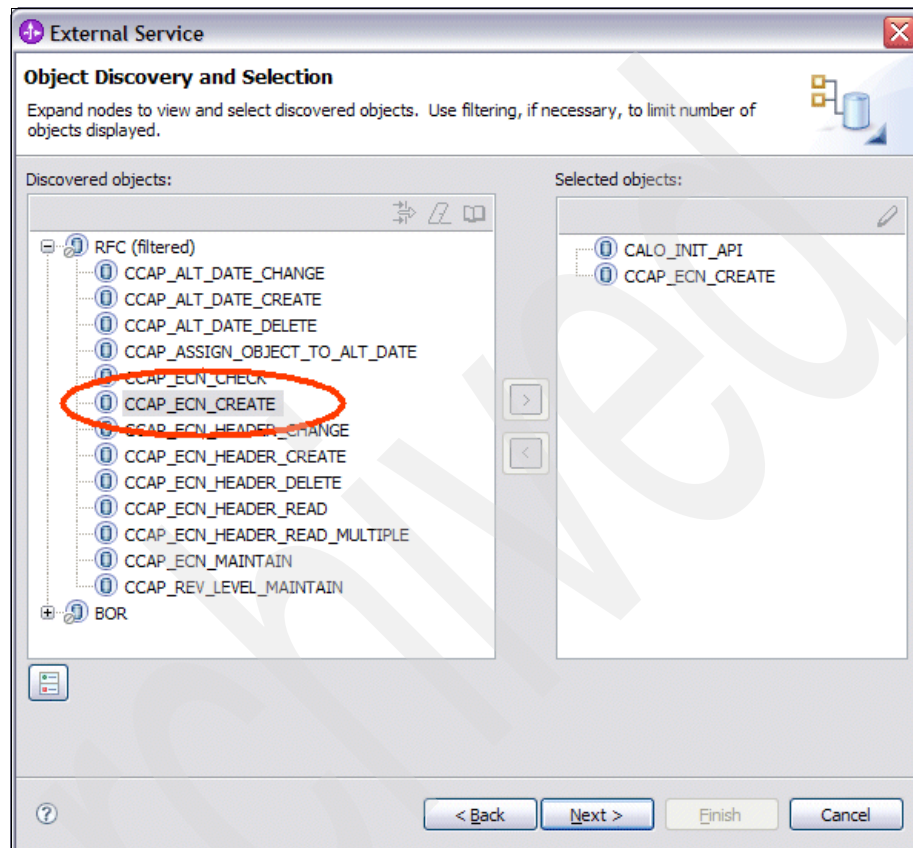


Figure 9-43 Select second function module

11. In the External Service pop-up window, select **Use SAP field names to generate attributes names**.
12. Click **Next** to proceed to the subsequent window.

13. Enter the values as shown in Figure 9-44.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations:\* ECN

Service operations:\*

Create

Add...

Remove

Sequence of RFC functions for the selected operation:\*

CALO\_INIT\_API  
CCAP\_ECN\_CREATE

Add...

Remove

Business object namespace: \* http://www.ibm.com/xmlns/prod/websphere/j2ca/sa

Specify the relative folder for generated business object:

Folder: bo/sapecn

☒ Generate a business graph for each business object

☒ Ignore errors in BAPI Return object

? < Back Next > Finish Cancel

Figure 9-44 Configure Composite Properties



14. Enter the values as shown in Figure 9-45.

The screenshot shows a dialog box titled "External Service" with a sub-header "Service Generation and Deployment Configuration". Below the sub-header is a description: "Specify properties for generating the service and running it on the server." There is a "Service operations" section with a note and an "Edit Operations..." button. The "Deployment properties" section includes a checked checkbox for "Specify a Java Authentication and Authorization Services (JAAS) alias security credential.", a text field for "J2C Authentication Data Entry:" containing "widNode/wccpc11800", a dropdown for "Deploy connector project:" set to "On server for use by multiple applications", a section for "Specify the settings used to connect to SAP Software at runtime:" with a dropdown for "Connection properties:" set to "Use predefined connection properties", and a text field for "JNDI Lookup Name:" containing "com/eis/wccpc11800AdapterCF". At the bottom are buttons for "< Back", "Next >", "Finish", and "Cancel".

Figure 9-45 Service Generation and Deployment Configuration

15. Enter `SAPEngineeringChangeInterface` as the interface name, leave all other values unchanged, and click **Finish**.
16. Add a reference to the mediation flow component and wire it to the `SAPEngineeringChangeInterface` import.
17. Select **OK** to add matching references when prompted.
18. Right-click the mediation flow and add the interface `EngineeringChangeManagementInterface`.
19. Right-click the mediation flow and generate an export of type SCA binding.

20. Right-click the mediation flow and select Regenerate Implementation.

The Assembly diagram must look similar to Figure 9-46.

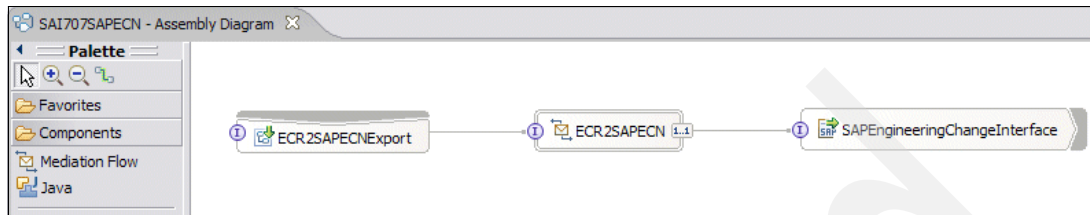


Figure 9-46 SAI707SAPECN Assembly diagram

### ***Implementing the ECM mediation flow***

The next steps describe how to implement a suitable mediation flow that maps from the generic interface to the regarding SAP interface.

1. Double-click the mediation flow.
2. Create an operation connection between operation createECN and createSapECNTxn (Figure 9-47 on page 309).
3. Place an XSL transformation node in the Request diagram and name it transformSAPECNRequest
4. Wire the Input node to the in terminal of the XSL transformation.
5. Wire the out terminal of the XSL transformation to the Callout node.

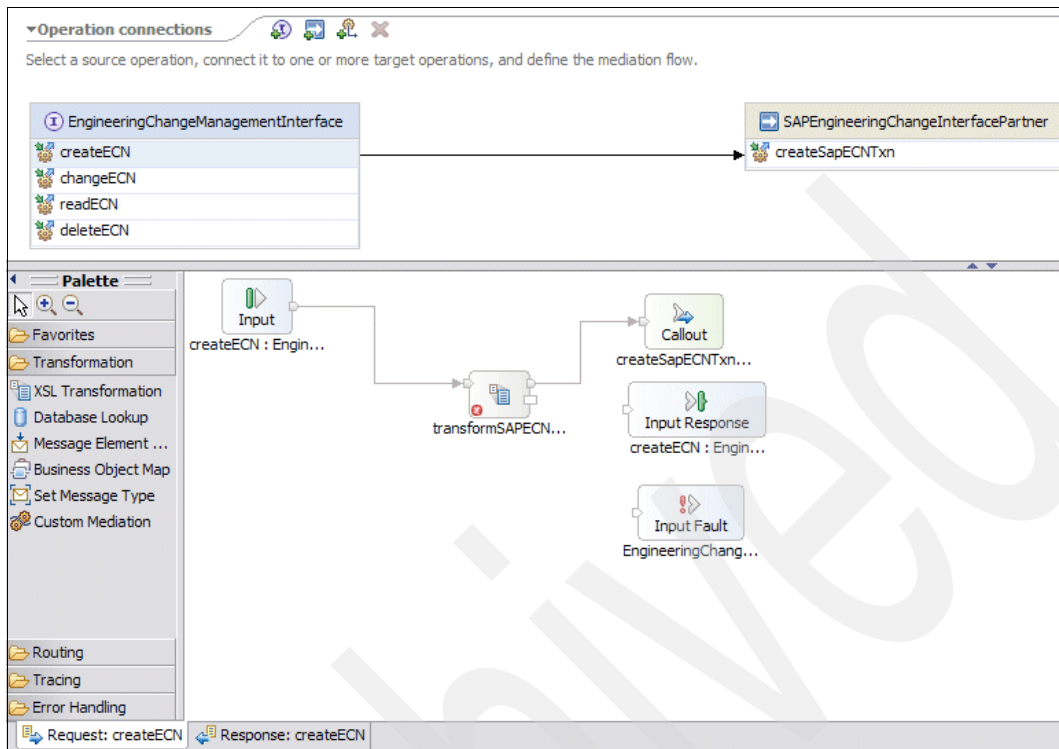


Figure 9-47 SAP ECN mediation flow: Request diagram

**Note:** The error mark is correct as there is no mapping file defined yet. This is done in the next steps.

6. Double-click the XSL transformation node transformSAPECNRequest.
7. Enter a suitable name for the mapping file, for example, CreateECN-Request and click **Finish**.
8. Expand the target object.
9. Add an Assign transformation to following attributes:
  - SapCaloInitApi/DataResetSign → !
  - SapCaloInitApi/DelLogAfterDays → 10
  - SapCaloInitApi/ExternlLogNo → API
  - SapCaloInitApi/FlagDbLogNo → X
  - SapCaloInitApi/FlagMsgOn → X

The map must look similar to Figure 9-48 on page 310.

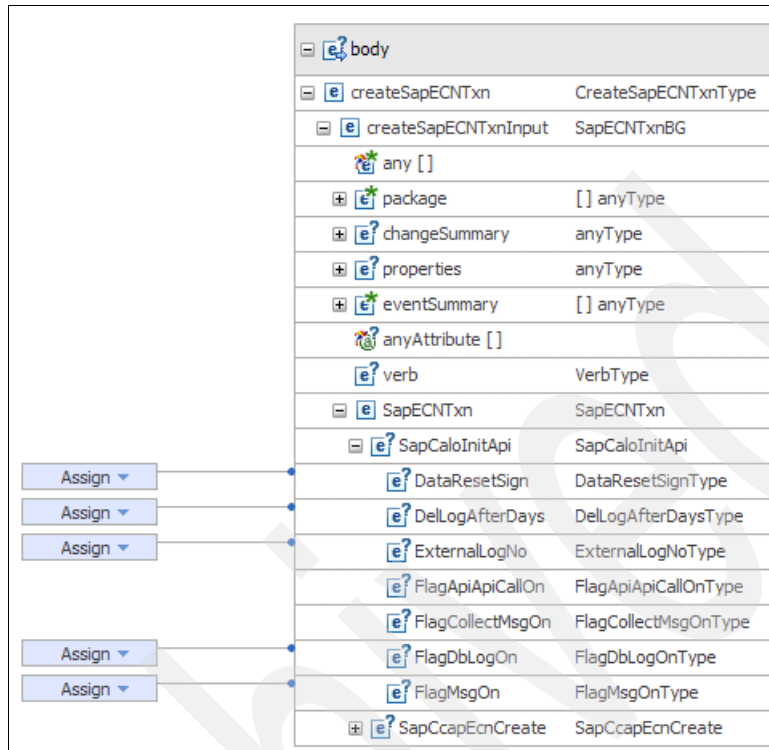


Figure 9-48 XSL mapping editor: SapCaloInitApi

10. Add a Move transformation between following attributes:

- Source ChangeNumber → Target SapHeader/ChangeNo
- Source Description → Target SapHeader/Descript
- Source ReasonforChange → Target SapHeader/ReasonChg
- Source ECN/Effectivity/ReasonforChange → Target SapHeader/ValidFrom

11. Add an Assign transformation to following target attributes:

- SapHeader/Status → 01
- SapHeader/Rank → 00
- SapHeader/ReleaseKey → 00
- SapObjectBom/Active → X
- SapObjectBom/ObjRequ → X
- SapObjectBom/MgtrecGen → X
- SapObjectDoc/Active → X
- SapObjectDoc/ObjRequ → X
- SapObjectDoc/MgtrecGen → X

- SapObjectMat/Active → X
- SapObjectMat/ObjRequ → X
- SapObjectMat/MgtrecGen → X

12. Save the mapping file.

13. Switch to the Response diagram.

14. Place an XSL transformation node in the Response diagram and name it transformSAPECNResponse.

15. Place an XSL transformation node in the Response diagram and name it processException.

16. Wire the out terminal of the Callout Response node with the input of the XSL transformation transformSAPECNResponse.

17. Wire the fail terminal of the Callout Response node with the input of the XSL transformation processException.

18. Wire the output terminal of the XSL transformation transformSAPECNResponse to the Input Response node.

19. Wire the output terminal of the XSL transformation processException to the Input Fault node.

The Response diagram must look similar to Figure 9-49:

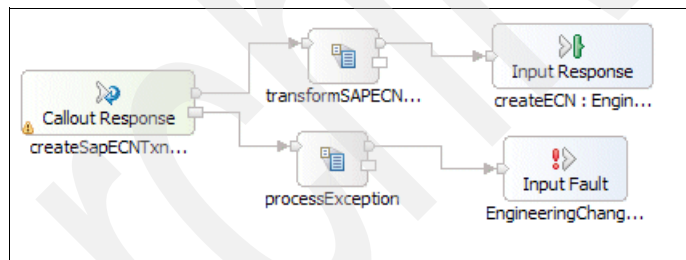


Figure 9-49 SAP ECN mediation flow Response diagram

20. Double-click the XSL transformation node transformSAPECNResponse.

21. Enter a suitable name for the mapping file, for example, CreateECN-Response and click **Finish**.

22. Expand the target object.

23. Add a Move transformation in the Source ChangeNo → Target Result/MessageV1 attributes.

24. Add an Assign transformation to the following target attributes:

- Result/Type → S
- Result/Message → SAP ECN successfully created.

25. Save the mapping file.
26. Double-click **XSL transformation node processException**.
27. Enter a suitable name for the mapping file, for example, createException and click **Finish**.
28. Expand the target object.
29. Add an Assign transformation to the following target attributes:
  - FaultName → ECNCreationException
  - MessageText → SAP ECN creation failed.
  - RootException → SAP BAPI call failed. Likely duplicated ECN.
30. Save the mapping file.
31. Save the mediation flow.

**Note:** The handling of the SAP system response is not typical. The used RFC module doesn't provide any error details in a return structure but throws ABAP™ exceptions that are not passed to the SCA layer.

The mediation module finished and can be deployed on the runtime server for testing.

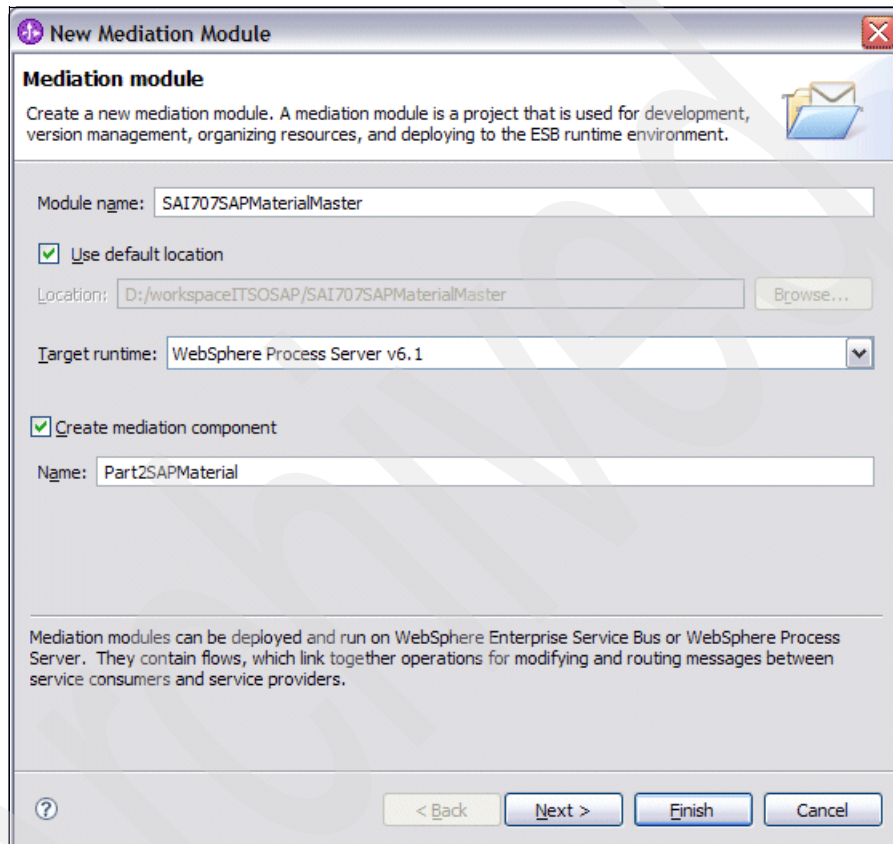
### **Develop Material Master mediation module**

This mediation module is responsible for creating a Material master record in the SAP system and to check if a Material master record is already available.

### ***Creating a Material Master mediation module***

Follow the steps below to create the Material Master mediation module.

1. Go to **File** → **New** → **Mediation Module**. Name the mediation module SAI707SAPMaterialMaster and the mediation component Part2SAPMaterial (Figure 9-50). Select **Next**.



*Figure 9-50 Create mediation module SAI707SAPMaterialMaster*

2. Select the check box to reference the ITSOPMLLibrary. Click **Finish**.

3. In the Assembly Diagram, choose the SAP Outbound adapter (Figure 9-51) in the Palette and place it on a free area on the canvas.

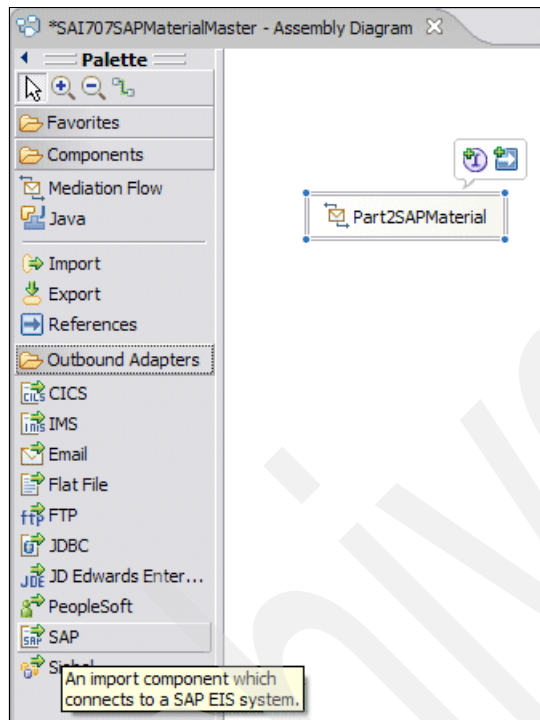


Figure 9-51 Select SAP outbound adapter



4. Enter the connection data for the SAP system and choose BAPI work unit (Figure 9-52). Click **Next**.

The screenshot shows a Windows-style dialog box titled "External Service" with a close button in the top right corner. Below the title bar, the text "Discovery Configuration" is displayed, followed by the instruction "Specify properties to begin discovery." To the right of this text is a small icon of a document with a magnifying glass. The main area of the dialog is titled "Connection properties" and contains a section for "SAP system connection information". This section includes several input fields: "Host name:" with the value "wccpc11.pgh.ibm.com", "System number:" with "00", "Client:" with "800", "Language code:" with "EN (English)" and a "Select..." button to its right, "Code page:" with "1100" and a dropdown arrow, "User name:" with "ktw", and "Password:" with "\*\*\*\*\*". Below these fields is a "SAP interface name:" dropdown menu showing "BAPI work unit". There is an "Advanced >>" button below the interface name field. At the bottom of the dialog, there is a checkbox labeled "Change logging properties for wizard" which is currently unchecked. The bottom right corner of the dialog contains four buttons: "< Back", "Next >", "Finish", and "Cancel".

Figure 9-52 SAP adapter Discovery configuration

5. Select the RFC tree and click the Filter icon.
6. Use BAPI\_MATERIAL\* as search pattern.

7. Choose BAPI\_MATERIAL\_SAVEDATA and BAPI\_MATERIAL\_EXISTENCECHECK (Figure 9-53). Click **Next**.

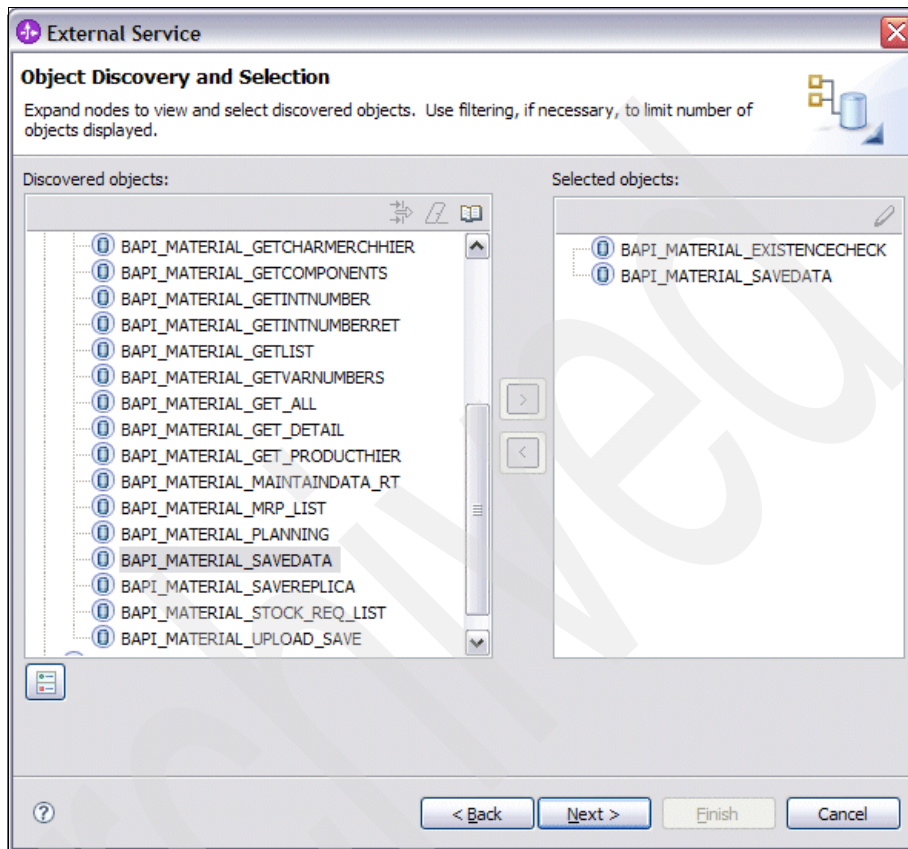


Figure 9-53 Object Discover and Selection: Material master mediation module

8. In the pop-up window, click **Generate the attributes based on the SAP field names** check box.
9. Click **Next** to proceed to the subsequent window.

10. Enter the values as shown in Figure 9-54 for the Create operation.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations: \*

Service operations: \*

Create
Retrieve

Sequence of RFC functions for the selected operation: \*

BAPI_MATERIAL_SAVEDATA
COMMIT

Business object namespace: \*

Specify the relative folder for generated business object:

Folder:

☒ Generate a business graph for each business object

☒ Ignore errors in BAPI Return object

Figure 9-54 Configure Composite Properties: Create operation

11. Enter the values as shown in Figure 9-55 for the Retrieve operation.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations: \*

Service operations: \*

Sequence of RFC functions for the selected operation: \*

Business object namespace: \*

Specify the relative folder for generated business object:

Folder:

☒ Generate a business graph for each business object  
☒ Ignore errors in BAPI Return object

Figure 9-55 Configure Composite Properties: Retrieve operation

12. Enter the values as shown in Figure 9-56.

**External Service**

**Service Generation and Deployment Configuration**

Specify properties for generating the service and running it on the server.

**Service operations**

If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.

**Deployment properties**

☒ Specify a Java Authentication and Authorization Services (JAAS) alias security credential.

J2C Authentication Data Entry:\* widNode/wccpc11800

Deploy connector project: On server for use by multiple applications

Specify the settings used to connect to SAP Software at runtime:

Connection properties: Use predefined connection properties

JNDI Lookup Name: \* com/eis/wccpc11800AdapterCF

< Back Next > Finish Cancel

Figure 9-56 Service Generation and Deployment Configuration

13. Click **Edit Operations...**
14. As we make an existence check using the Retrieve operation we adjust the operation name to fit better.
15. Change operation name retrieveSapMaterialTxn to existsSapMaterialTxn.
16. Enter SAPMaterialMasterInterface as the interface name, leave all other values unchanged, and click **Finish**.
17. Add a reference to the mediation flow component and wire it to the SAPMaterialMasterInterface import.

18. Select **OK** to add matching references when prompted to do so.
19. Right-click the mediation flow and add the interface `MaterialMasterInterface`.
20. Right-click the mediation flow and generate an export of type SCA binding.
21. Right-click the mediation flow and select **Regenerate Implementation**.

The Assembly diagram must look similar to Figure 9-57:

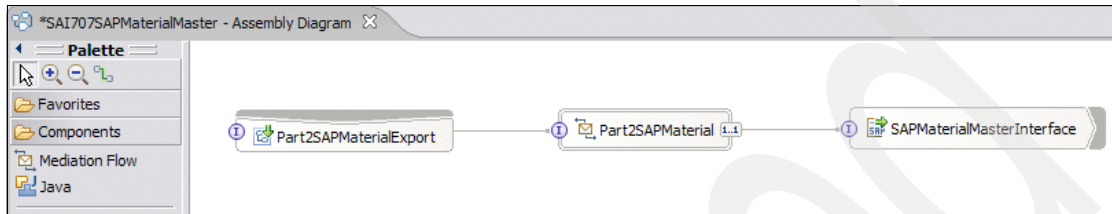


Figure 9-57 SAI707SAPMaterialMaster Assembly diagram

### Implementing the Material Master mediation flow

The next steps describe how to implement a suitable mediation flow that maps from the generic interface to the regarding SAP interface.

1. Double-click the mediation flow.
2. Create an operation connection between operation `createMaterial` and `createSapMaterialTxn`. See Figure 9-58.

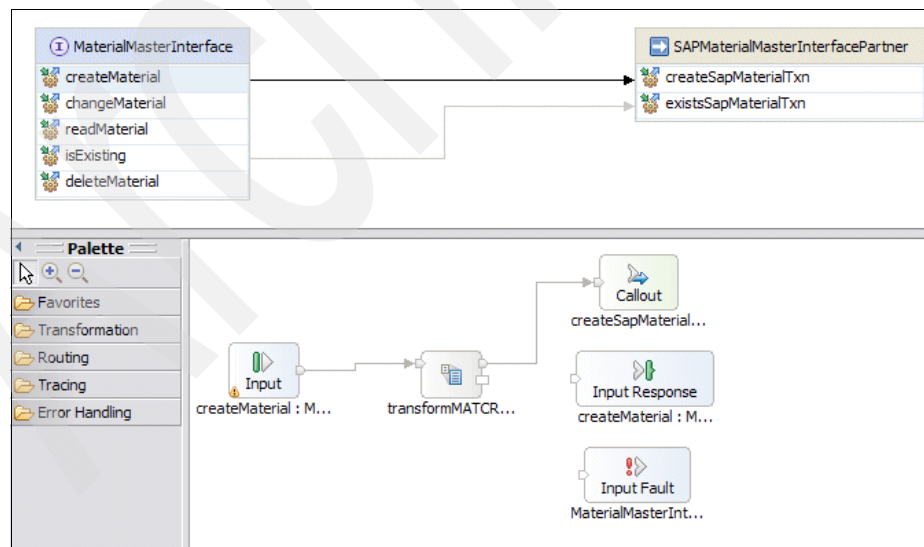


Figure 9-58 Mediation flow diagram request: Create operation

3. Create an operation connection between operation `isExisting` and `existsSapMaterialTxn`.
4. Place a XSL transformation node in the Request diagram for the create operation and name it `transformMATCREATEReq`.
5. Place a XSL transformation node in the Response diagram for the create operation and name it `transformMATCREATEResp`.
6. Place a XSL transformation node in the Request diagram for the create operation and name it `transformMATEXISTReq`.
7. Place a XSL transformation node in the Response diagram for the create operation and name it `transformMATEXISTResp`.
8. Wire the Input, XSL transformation and Callout nodes together.

The mediation flow diagram for the create operation is similar to Figure 9-58 on page 320.

The mediation flow diagram for the create operation must look similar to Figure 9-59.

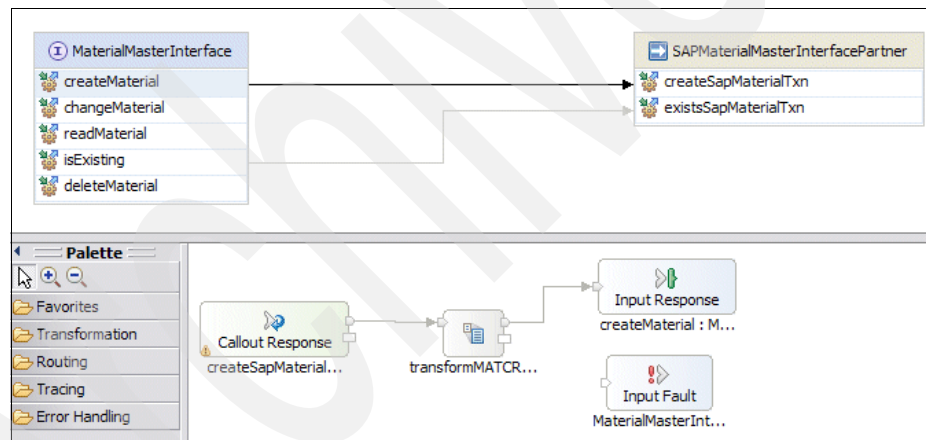


Figure 9-59 Mediation flow diagram response: Create operation

The mediation flow diagram for the exists operation looks similar and is not displayed here in more detail.

9. Double-click the XSL transformation node `transformMATCREATEReq`.
10. Enter a suitable name for the mapping file (for example, `CreateMaterial-Request`). Click **Finish**.
11. Expand the target object.

12. Add a Move transformation between the following attributes:

- Source PartNumber → Target SapHeaddata/Material
- Source Description → Target SapMaterialdescription/MatlDesc

13. Add an Assign transformation to the following target attributes:

- SapClientdata/MatlGroup → 00305
- SapClientdata/BaseUom → EA
- SapClientdata/Type → X
- SapClientdata/Message → X
- SapHeaddata/Indsector → M
- SapHeaddata/MatlType → HALB
- SapHeaddata/BasicView → X
- SapHeaddata/MRPView → X
- SapPlantdata/Plant → 1000
- SapPlantdata/MrpType → ND
- SapPlantdata/Plant → X
- SapPlantdata/MrpType → X
- SapMaterialdescription/Langu → EN

14. Save the mapping file.

15. Double-click the XSL transformation node transformMATCREATEResp.

16. Enter a suitable name for the mapping file, for example, CreateMaterial-Response and click **Finish**.

17. Expand the target object.

18. Add a Move transformation between the following attributes:

- Source SapReturn/Type → Target Result/Type
- Source SapReturn/Id → Target Result/Id
- Source SapReturn/Number → Target Result/Number
- Source SapReturn/Message → Target Result/Message
- Source SapReturn/LogNo → Target Result/LogNo
- Source SapReturn/MessageV1 → Target Result/MessageV1
- Source SapReturn/MessageV2 → Target Result/MessageV2
- Source SapReturn/MessageV3 → Target Result/MessageV3
- Source SapReturn/MessageV4 → Target Result/MessageV4

19. Save the mapping file.

20. Double-click the XSL transformation node transformMATExistReq.

21. Enter a suitable name for the mapping file (for example, ExistMaterial-Request) and click **Finish**.

22. Expand the target object.

23. Add a Move transformation between the Source materialID → Target SapBapiMaterialExistencecheck/Material attributes.



24. Save the mapping file.
25. Double-click the XSL transformation node transformMATEXISTResp.
26. Enter a suitable name for the mapping file (for example, ExistMaterial-Response) and click **Finish**.
27. Expand the target object.
28. Add a Move transformation between the following attributes:
  - Source SapReturn/Type → Target Result/Type
  - Source SapReturn/Id → Target Result/Id
  - Source SapReturn/Number → Target Result/Number
  - Source SapReturn/Message → Target Result/Message
  - Source SapReturn/LogNo → Target Result/LogNo
  - Source SapReturn/MessageV1 → Target Result/MessageV1
  - Source SapReturn/MessageV2 → Target Result/MessageV2
  - Source SapReturn/MessageV3 → Target Result/MessageV3
  - Source SapReturn/MessageV4 → Target Result/MessageV4
29. Save the mapping file.

**Note:** This component provides BAPI modules that deliver all responses in an SAP Return structure. Within the mediation flow, additional exception handling can be included. For the scenario in the Redbooks publication, the processing of the standard return structure is sufficient.

The development of this mediation module is finished and the module can be deployed on the runtime server for testing.

## Develop BOM mediation module

This mediation module is responsible for creating a material BOM entry in the SAP system. There is no function module available in the SAP system to check the availability of a BOM. Therefore, a read function module is used and when this call returns an error, it is assumed that the BOM does not exist yet.

### Creating a BOM mediation module

Complete the following steps to create the mediation module.

1. Go to **File** → **New** → **Mediation Module** and name the mediation module SAI707SAPMatBOM and the mediation component PLMBOM2SAPBOM (Figure 9-60). Select **Next**.

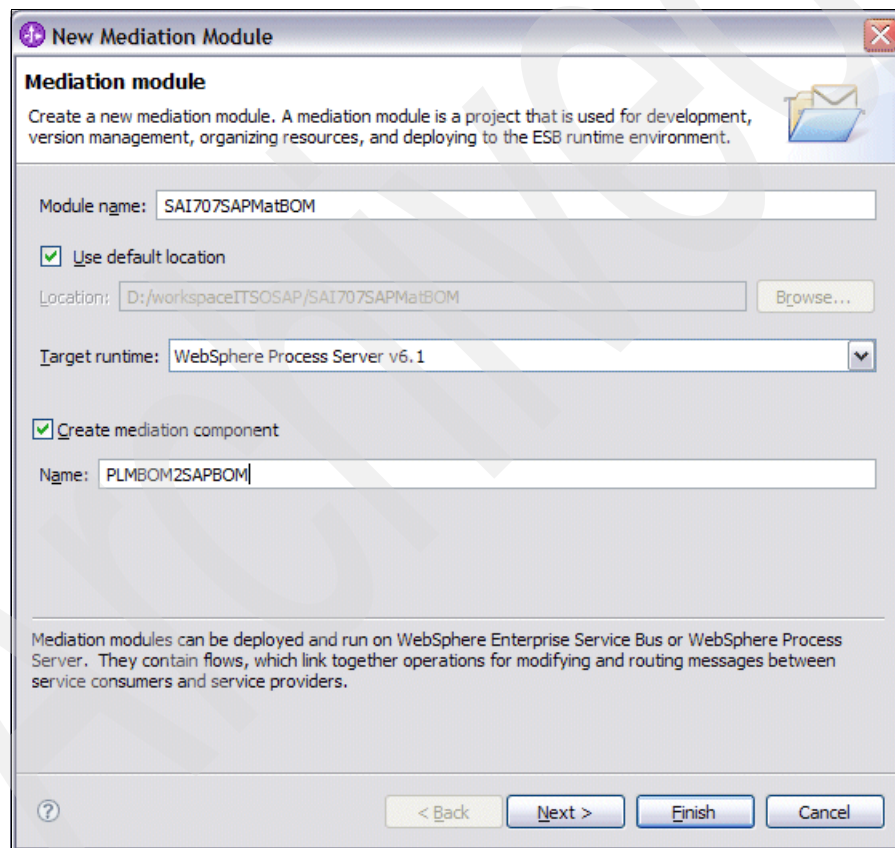


Figure 9-60 Create mediation module SAI707SAPMatBOM

2. Select the check box to reference the ITSOPMLLibrary, click **Finish**.

3. In the Assembly Diagram (Figure 9-61), choose the SAP Outbound adapter in the Palette and place it on a free area on the canvas.

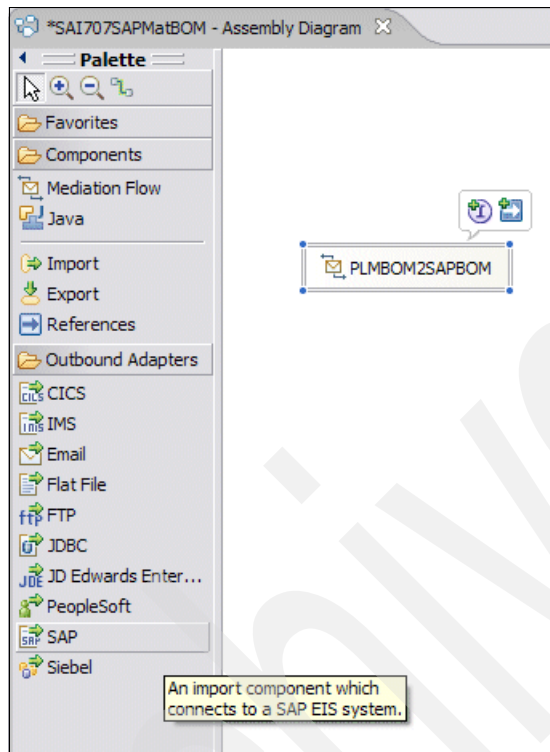


Figure 9-61 Select SAP outbound adapter

4. Enter the connection data for the SAP system and choose BAPI work unit (Figure 9-62). Click **Next**.

**External Service**

**Discovery Configuration**  
Specify properties to begin discovery.

Connection properties

SAP system connection information

Host name: \* wccpc11.pgh.ibm.com

System number: 00

Client: 800

Language code: EN (English) Select...

Code page: 1100

User name: \* ktw

Password: \* \*\*\*\*\*

SAP interface name: BAPI work unit

Advanced >>

☐ Change logging properties for wizard

? < Back Next > Finish Cancel

Figure 9-62 SAP adapter Discovery configuration

5. Select the RFC tree and click the Filter icon.
6. Use CALO\* as search pattern.

7. Choose CALO\_API\_INIT (see Figure 9-63) from the discovered object list and add it to the selected object list.

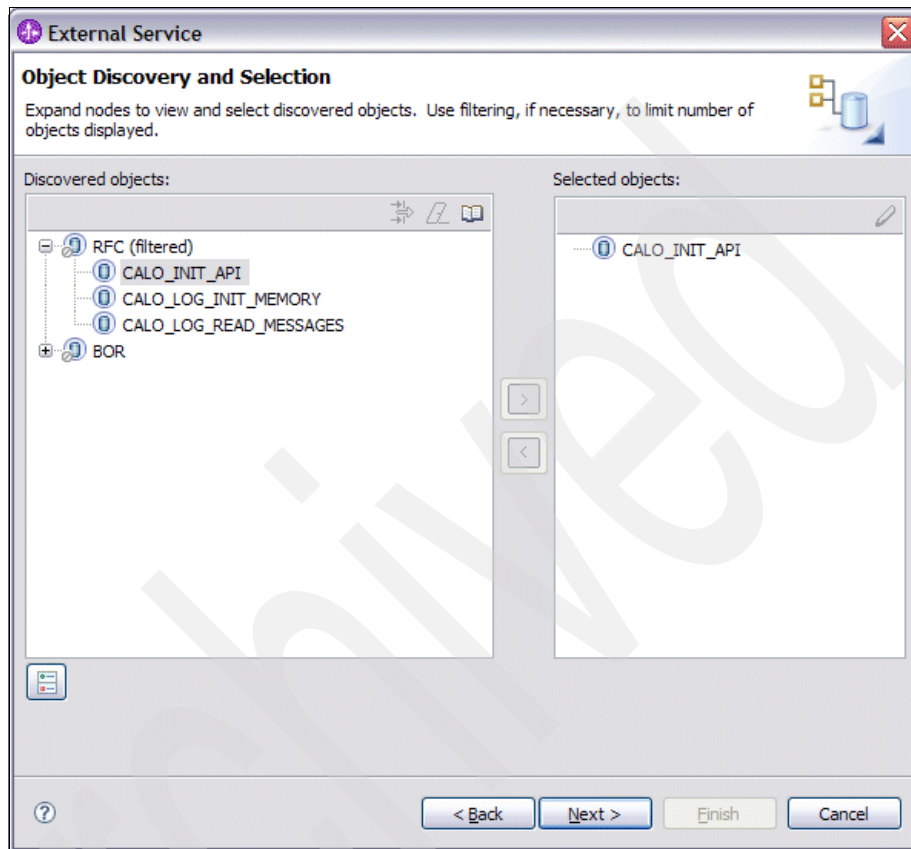


Figure 9-63 Object Discovery and Selection: CALO\_INIT\_API

8. Select in the pop-up window to generate the attributes based on the SAP field names.
9. Use CSAP\_MAT\_BOM\* as search pattern.

10. Choose CSAP\_MAT\_BOM\_CREATE and CSAP\_MAT\_BOM\_READ (see Figure 9-64) and click **Next**.

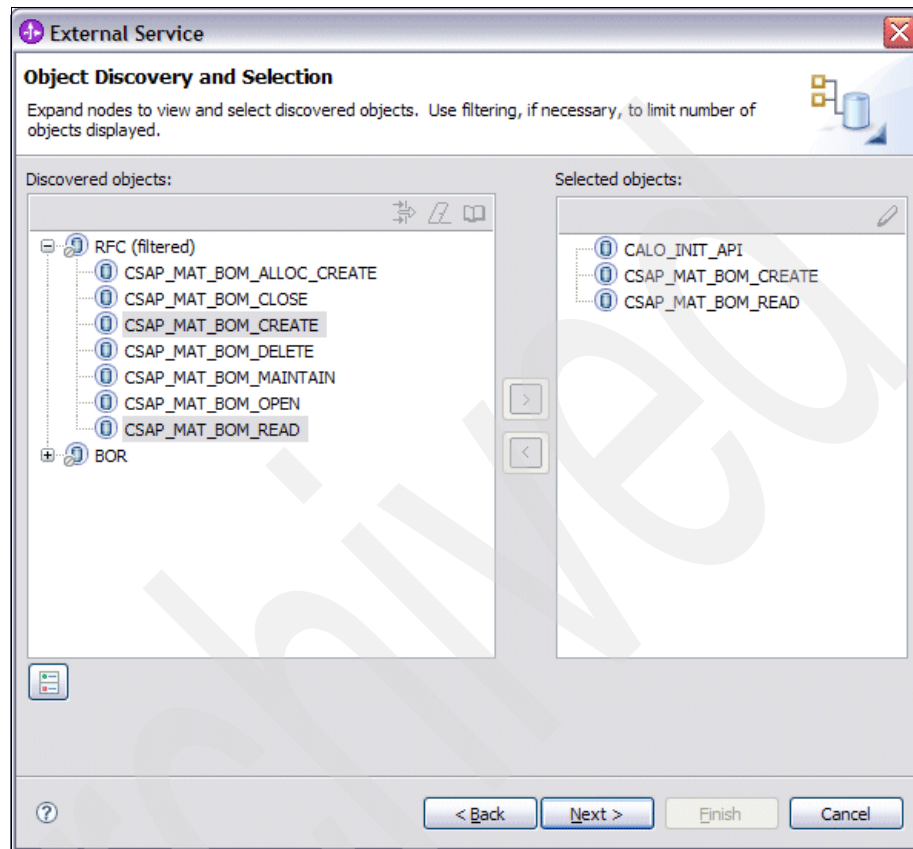


Figure 9-64 Object Discover and Selection: BOM create and read

11. Select in the pop-up window to generate the attributes based on the SAP field names. Click **Next**

12. Enter the values as shown in Figure 9-65 for the Create operation.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations: \*

Service operations: \*

Sequence of RFC functions for the selected operation: \*

Business object namespace: \*

Specify the relative folder for generated business object:

Folder:

☒ Generate a business graph for each business object

☒ Ignore errors in BAPI Return object

Figure 9-65 Configure Composite Properties: Create operation

13. Enter the values as shown in Figure 9-66 for the Retrieve operation.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations: \*

Service operations: \*

Sequence of RFC functions for the selected operation: \*

Business object namespace: \*

Specify the relative folder for generated business object:

Folder:

☒ Generate a business graph for each business object  
☒ Ignore errors in BAPI Return object

Figure 9-66 Configure Composite Properties: Retrieve operation



14. Enter the values as shown in Figure 9-67.

**External Service**

**Service Generation and Deployment Configuration**

Specify properties for generating the service and running it on the server.

**Service operations**

If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.

**Deployment properties**

☒ Specify a Java Authentication and Authorization Services (JAAS) alias security credential.

J2C Authentication Data Entry: \* widNode/wccpc11800

Deploy connector project: On server for use by multiple applications

Specify the settings used to connect to SAP Software at runtime:

Connection properties: Use predefined connection properties

JNDI Lookup Name: \* com/eis/wccpc11800AdapterCF

< Back Next > Finish Cancel

Figure 9-67 Service Generation and Deployment Configuration

15. Enter `SAPBillOfMaterialInterface` as the interface name, leave all other values unchanged, and click **Finish**.
16. Add a reference to the mediation flow component and wire it to the `SAPBillOfMaterialInterface` import.
17. Select **OK** to add matching references when prompted.
18. Right-click the mediation flow and add the interface `BillOfMaterialInterface`.
19. Right-click the mediation flow and generate an export of type SCA binding.

20. Right-click the mediation flow and select **Regenerate Implementation**.

The Assembly diagram must look similar to Figure 9-68:

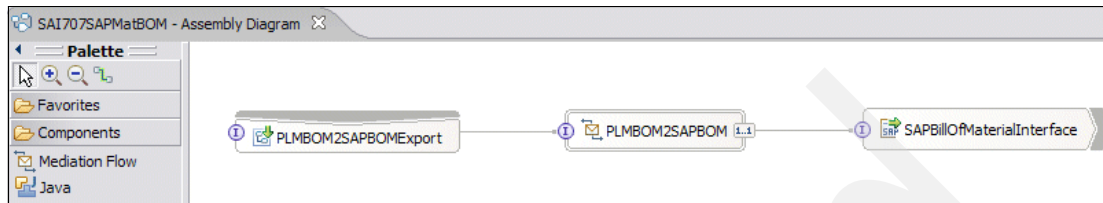


Figure 9-68 SAI707SAPMatBOM Assembly diagram

### ***Implementing the BOM mediation flow***

The next steps describe how to implement a suitable mediation flow that maps from the generic interface to the regarding SAP interface.

1. Double-click the mediation flow.
2. Create an operation connection between operation createBOM and createSapMaterialBOMTxn.
3. Create an operation connection between operation retrieveBOM and retrieveSapMaterialBOMTxn.
4. Place a XSL transformation node in the Request diagram for the create operation and name it transformMatBOMCREATEReq.
5. Place a XSL transformation node in the Response diagram for the create operation and name it transformMatBOMCREATEResp.
6. Place a XSL transformation node in the Response diagram and name it processCreateException.

7. The mediation flow diagram for the create operation must look similar to Figure 9-69.

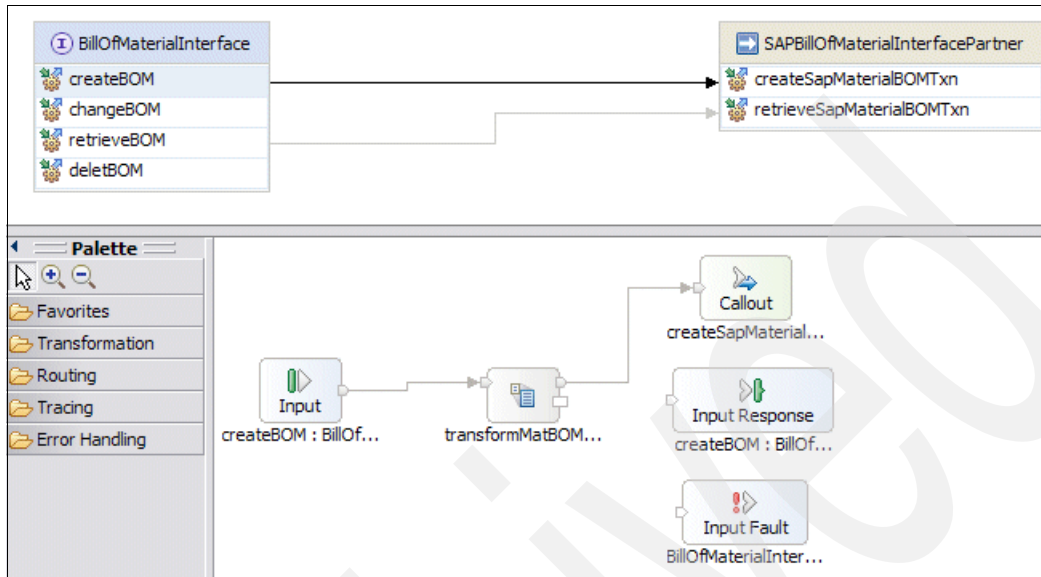


Figure 9-69 Mediation flow diagram reqs: Create operation

8. The mediation flow diagram for the create operation must look similar to Figure 9-70.

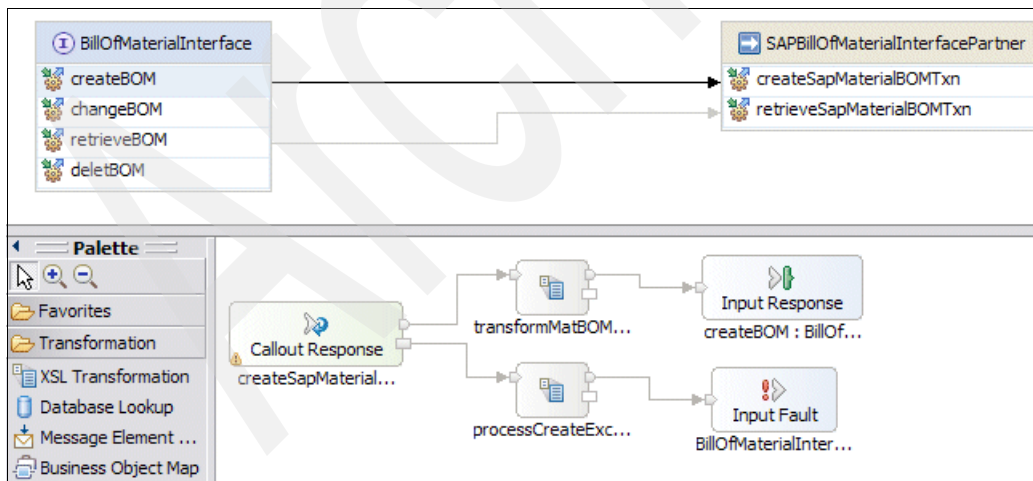


Figure 9-70 Mediation flow diagram response: Create operation

9. Double-click the XSL transformation node transformMatBOMCREATEReq.

10. Enter a suitable name for the mapping file, for example, CreateMatBOM-Request and click **Finish**.

11. Expand the target object.

12. Add an Assign transformation to the following attributes:

- SapCaloInitApi/DataResetSign → !
- SapCaloInitApi/DelLogAfterDays → 10
- SapCaloInitApi/ExternalLogNo → API
- SapCaloInitApi/FlagDbLogNo → X
- SapCaloInitApi/FlagMsgOn → X

The map must look similar to Figure 9-71 in this stage:

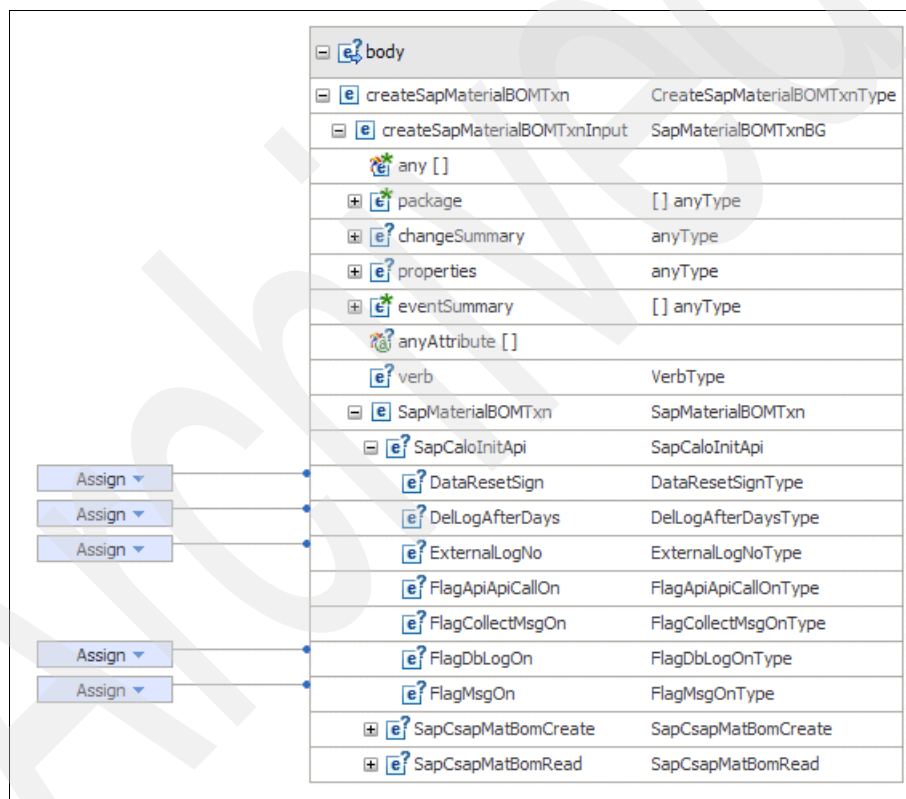


Figure 9-71 XSL mapping editor: SapCaloInitApi

13. Add a Move transformation between the following attributes:

- Source AssemblyPartNumber → Target SapCsapMatBOMCreate/Material
- Source ChangeNumber → Target SapCsapMatBOMCreate/ChangeNo

14. Add an Assign transformation to the following target attributes:
  - SapCsapMatBOMCreate/BOMUsage → 1
  - SapCsapMatBOMCreate/Plant → 1000
  - SapCsapMatBOMCreate/SapIStko/BaseQuan → 1
  - SapCsapMatBOMCreate/SapIStko/BaseUnit → EA
  - SapCsapMatBOMCreate/SapIStko/BOMStatus → 1
15. Add a Submap named BOMItem2SAPBOMItem between the Source BOMItems → Target SapCsapMatBOMCreate/SapTStpo attributes:
  - a. Within the submap add an Assign transformation to the following target attributes.
    - SapTStpo/ItemCateg → L
    - SapTStpo/RelCost → X
    - SapTStpo/RelPmaint → X
  - b. Within the submap add an Move transformation to the following target attributes.
    - Source BOMItem/LineNumber → Target SapTStpo/ItemNo
    - Source BOMItem/ComponentPartNumber → Target SapTStpo/Component
    - Source BOMItem/Quantity → Target SapTStpo/CompQty
    - Source BOMItem/Unit → Target SapTStpo/CompUnit
16. Save the submap file.
17. Save the mapping file.
18. Double-click the XSL transformation node transformMatBOMCREATEResp.
19. Enter a suitable name for the mapping file (for example, CreateMatBOM-Response) and click **Finish**.
20. Expand the target object.
21. Add an Assign transformation to the following target attributes:
  - Result/Type → S
  - Result/Message → SAP BOM created successfully.
22. Save the mapping file.
23. Double-click the XSL transformation node processCreateException.
24. Enter a suitable name for the mapping file, for example, createException and click **Finish**.
25. Expand the target object.

26. Add an Assign transformation to the following target attributes:

- FaultName → BOMCreationException
- MessageText → SAP BOM creation failed.
- RootException → SAP BAPI call failed.

27. Save the mapping file.

28. Save the mediation flow.

**Note:** The handling of the SAP system response is not typical. The used RFC module doesn't provide any error details in a return structure but throws ABAP exceptions that are not passed to the SCA layer.

Switch to the connection of the retrieve operation and open the request flow.

1. Place a XSL transformation node in the Request diagram for the retrieve operation and name it transformMatBOMRETRIEVEReq.
2. Place a Message Element Setter node in the Request diagram for the retrieve operation and name it populateContext.
3. Place a XSL transformation node in the Response diagram for the retrieve operation and name it transformMatBOMRETRIEVEResp.
4. Place a Message Element Setter node in the Request diagram for the retrieve operation and name it readfromContext.
5. Place a XSL transformation node in the Response diagram and name it processRetrieveException.
6. Wire the Input, XSL transformation Message Element Setter and Callout nodes together.

7. Wire the Callout Response out and fail terminals with the according XSL transformations, Message Element Setter, Callout and Fault nodes.

The mediation flow diagram for the create operation must look similar to Figure 9-72.

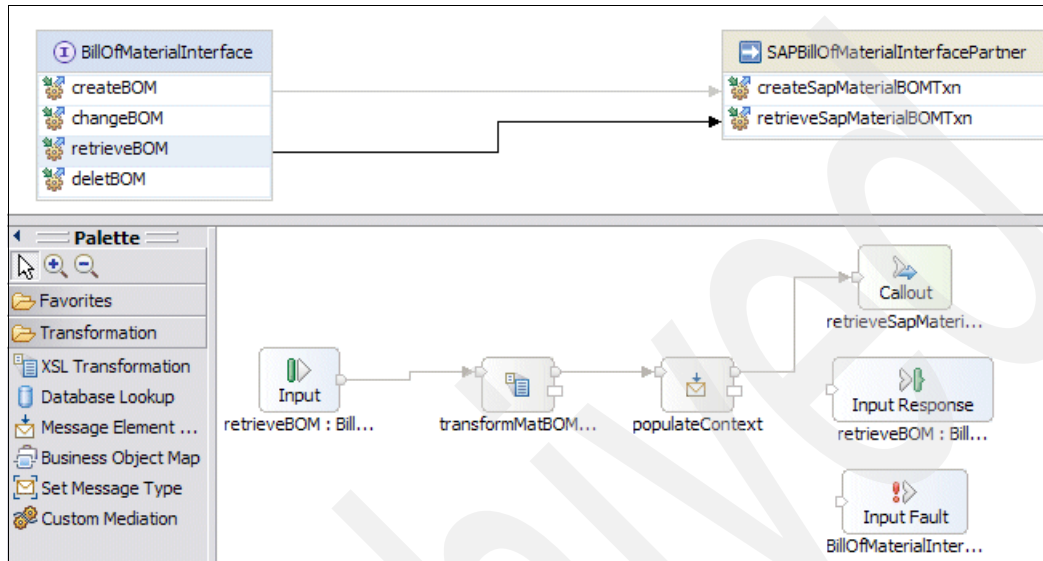


Figure 9-72 Mediation flow diagram request: Retrieve operation

The mediation flow diagram for the create operation must look similar to Figure 9-73.

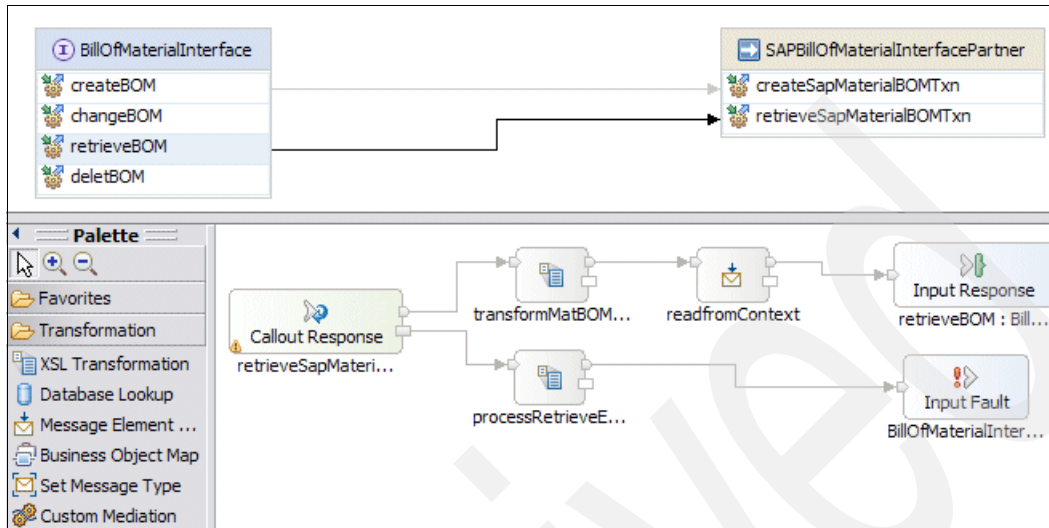


Figure 9-73 Mediation flow diagram response: Retrieve operation

8. Add a business object to the context by clicking the right side on the + icon. See Figure 9-74.

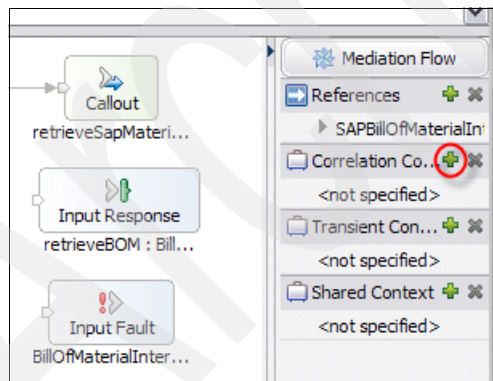


Figure 9-74 Mediation flow: Add context object



9. Select the business object SapCsapMatBOMRead as a context object. See Figure 9-75.

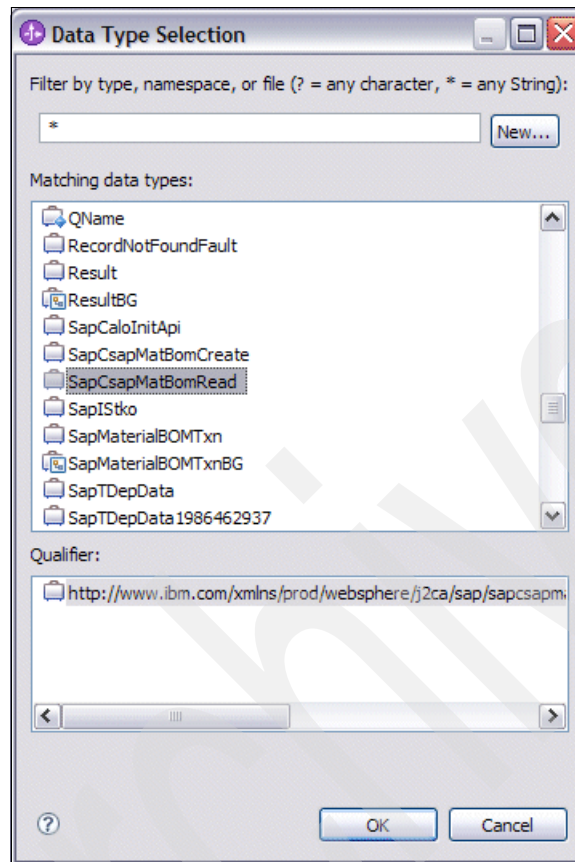


Figure 9-75 Mediation flow: Select object SAPCcsapMatBOMRead

10. Double-click the XSL transformation node transformMatBOMRETRIEVEReq.
11. Enter a suitable name for the mapping file (for example, RetrieveMatBOM-Request) and click **Finish**.
12. Expand the target object.
13. Add an Assign transformation to the following attributes:
- SapCaloInitApi/DataResetSign → !
  - SapCaloInitApi/DelLogAfterDays → 10
  - SapCaloInitApi/ExternlLogNo → API
  - SapCaloInitApi/FlagDbLogNo → X
  - SapCaloInitApi/FlagMsgOn → X

The map must look similar to Figure 9-76 in this stage.

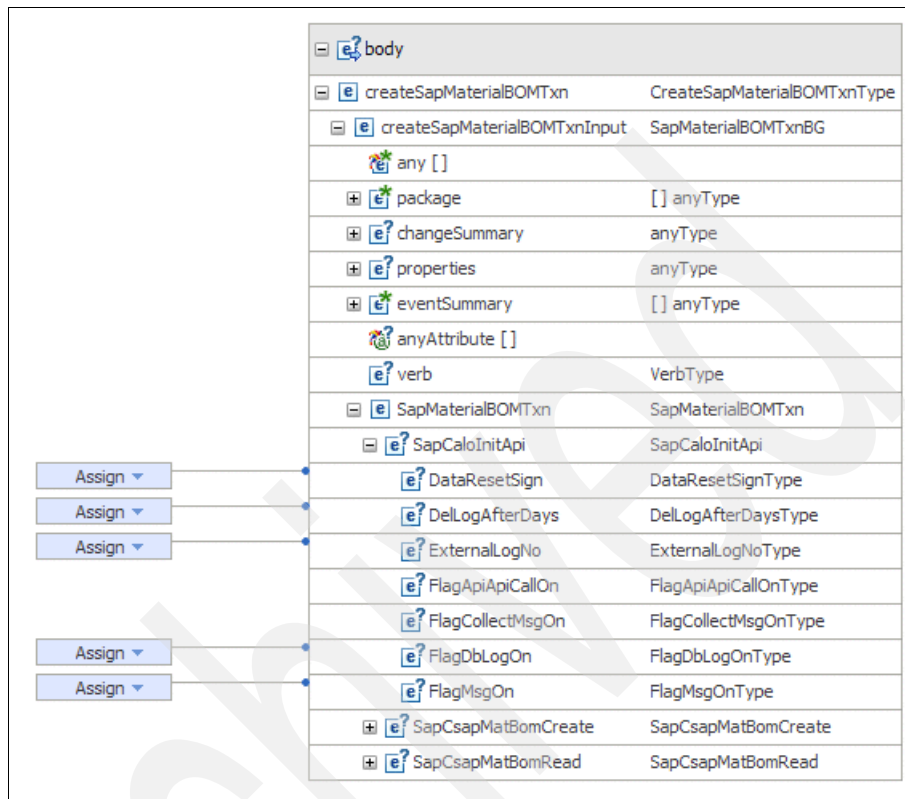


Figure 9-76 XSL mapping editor: SapCaloInitApi

14. Add a Move transformation between the following attributes:

- Source BOMID → Target SapCsapMatBOMRead/Material
- Source plant → Target SapCsapMatBOMRead/Plant
- Source usage → Target SapCsapMatBOMRead/BOMUsage

15. Save the mapping file.

16. Double-click the Message Element Setter node populateContext.

17. Add the following entry in the Setter node:

- Target /context/correlation/Plant
- Type copy
- Source  
/body/retrieveSapMaterialBOMTxn/retrieveSapMaterialBOMTxnInput/SapMaterialBOMTxn/SapCsapMatBOMRead/Plant

18. Add the following entry in the Setter node:
  - Target /context/correlation/Material
  - Type copy
  - Source  
/body/retrieveSapMaterialBOMTxn/retrieveSapMaterialBOMTxnInput/SapMaterialBOMTxn/SapCsapMatBOMRead/Material
19. Add the following entry in the Setter node:
  - Target /context/correlation/BOMUsage
  - Type copy
  - Source  
/body/retrieveSapMaterialBOMTxn/retrieveSapMaterialBOMTxnInput/SapMaterialBOMTxn/SapCsapMatBOMRead/BOMUsage
20. Double-click the XSL transformation node transformMatBOMRETRIEVEResp.
21. Enter a suitable name for the mapping file (for example, RetrieveMatBOM-Response) and click Finish.
22. Add a Move transformation between the following attributes:
  - Source ValidFrom → Target BOM/Effectivity/StartDateEffectivity
  - Source ChangeNo → Target BOM/ChangeNumber
23. Add a Submap named SAPBOMItem2BOMItem between the Source SapCsapMatBOMCreate/SapTStpo → Target BOMItems attributes.
24. Within the submap, add a Move transformation to the following target attributes:
  - Source SapTStpo/ItemCateg → Target BOMItem/ItemCategory
  - Source SapTStpo/ItemNo → Target BOMItem/LineNumber
  - Source SapTStpo/Component → Target BOMItem/ComponentPartNumber
  - Source SapTStpo/CompQty → Target BOMItem/Quantity
  - Source SapTStpo/CompUnit → Target BOMItem/Unit
  - Source SapTStpo/ValidFrom → Target BOMItem/Effectivity/StartDateEffectivity
25. Save the submap file.
26. Save the mapping file.
27. Double-click the XSL transformation node processRetrieveException.
28. Enter a suitable name for the mapping file, for example, retrieveException and click **Finish**.

29. Expand the target object.

30. Add an Assign transformation to the following target attributes:

- FaultName → BOMRetrievalException
- MessageText → SAP BOM retrieval failed.
- RootException → SAP BAPI call failed.

The development of this mediation module is finished and the module can be deployed on the runtime server for testing.

## Develop Document mediation module

This mediation module is responsible to create a Document record in the SAP system and to check if a Document is already existing in the system.

### Creating a Document mediation module

Perform the following steps to create the mediation module.

1. Go to **File**→**New** → **Mediation Module** and name the mediation module SAI707SAPDocument (Figure 9-77) and the mediation component Doc2SAPDocument. Select **Next**.

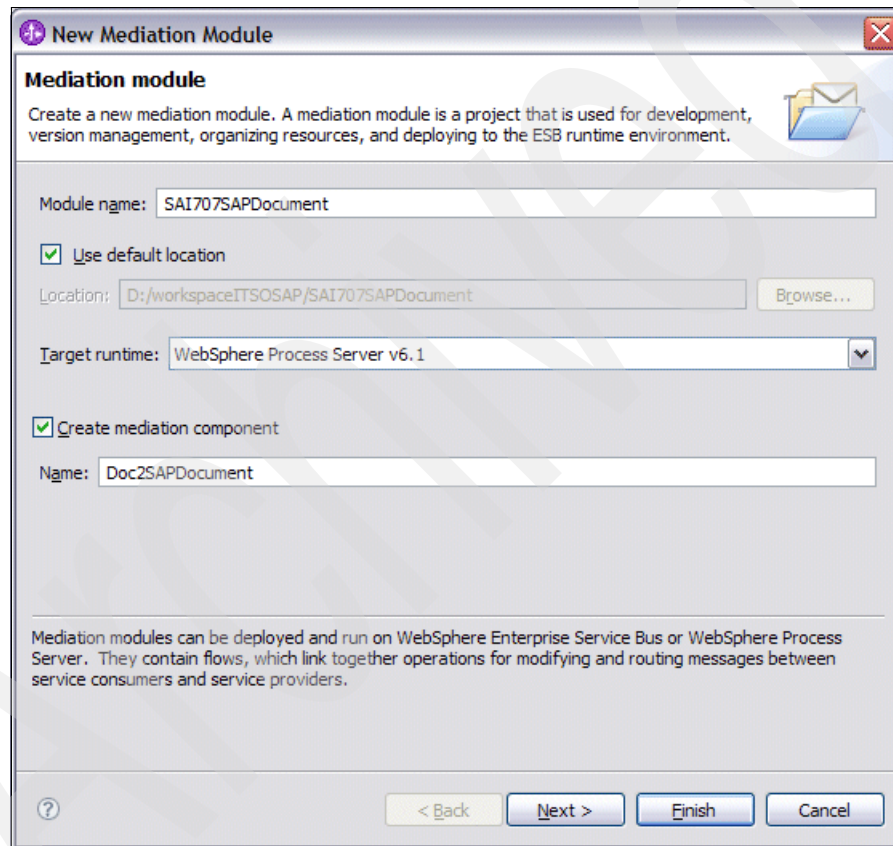


Figure 9-77 Create mediation module SAI707SAPDocument

2. Select the check box to reference the ITSOPMLLibrary, click **Finish**.

3. In the Assembly Diagram (Figure 9-78), choose the SAP Outbound adapter in the Palette and place it on a free area on the canvas.

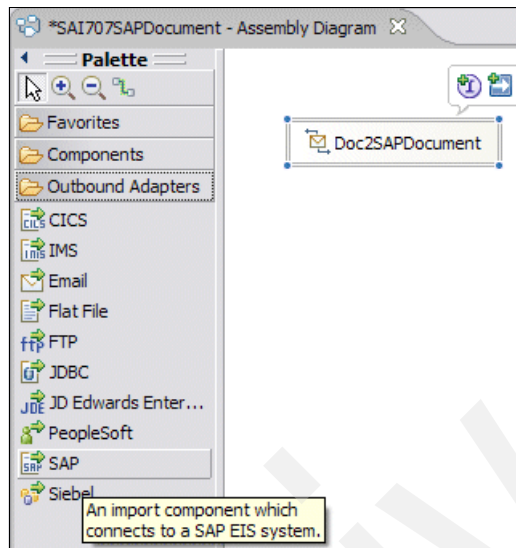


Figure 9-78 Select SAP outbound adapter

4. Enter the connection data for the SAP system (Figure 9-79) and choose BAPI work unit, click **Next**.

**External Service**

**Discovery Configuration**  
Specify properties to begin discovery.

Connection properties

SAP system connection information

Host name: \* wccpc11.pgh.ibm.com

System number: 00

Client: 800

Language code: EN (English) Select...

Code page: 1100

User name: \* ktw

Password: \* \*\*\*\*\*

SAP interface name: BAPI work unit

Advanced >>

☐ Change logging properties for wizard

? < Back Next > Finish Cancel

Figure 9-79 SAP adapter Discovery configuration

5. Select the RFC tree and click the Filter icon.
6. Use BAPI\_DOCUMENT\* as search pattern.

7. Add BAPI\_DOCUMENT\_CREATE2 and BAPI\_DOCUMENT\_EXISTENCECHECK to the selected objects list (Figure 9-80) and click **Next**.

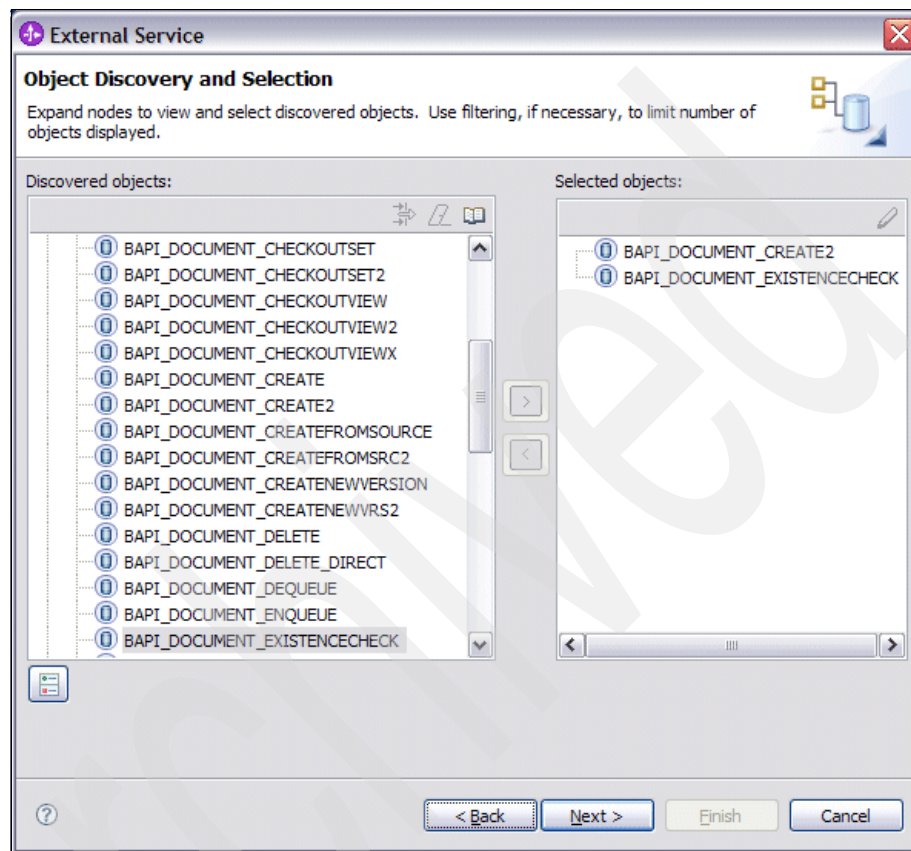


Figure 9-80 Object Discover and Selection:Material master mediation module

8. In the pop-up window, click check box to Generate the attributes based on the SAP field names. Click **Next**.



9. Enter the values as shown in Figure 9-81 for the Create operation.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations: \* Document

Service operations: \*

Create  
Retrieve

Add...  
Remove

Sequence of RFC functions for the selected operation: \*

BAPI\_DOCUMENT\_CREATE2  
COMMIT

Add...  
Remove

Business object namespace: \* http://www.ibm.com/xmlns/prod/websphere/j2ca/sa

Specify the relative folder for generated business object:

Folder: bo/sapdocument

☒ Generate a business graph for each business object

☒ Ignore errors in BAPI Return object

? < Back Next > Finish Cancel

Figure 9-81 Configure Composite Properties: Create operation

10. Enter the values as shown in Figure 9-82 for the Retrieve operation.

**External Service**

**Configure Composite Properties**  
Specify properties that apply to all selected objects.

Map service operations to RFC functions

Business object name for service operations: \* Document

Service operations: \*

Create  
Retrieve

Add...  
Remove

Sequence of RFC functions for the selected operation: \*

BAPI\_DOCUMENT\_EXISTENCECHECK

Add...  
Remove

Business object namespace: \* http://www.ibm.com/xmlns/prod/websphere/j2ca/sa

Specify the relative folder for generated business object:

Folder: bo/sapdocument

☒ Generate a business graph for each business object  
☒ Ignore errors in BAPI Return object

? < Back Next > Finish Cancel

Figure 9-82 Configure Composite Properties: Retrieve operation

11. Enter the values as shown in Figure 9-83.

The screenshot shows a dialog box titled "External Service" with a sub-tab "Service Generation and Deployment Configuration". The main instruction is "Specify properties for generating the service and running it on the server." The dialog is divided into two sections: "Service operations" and "Deployment properties".

**Service operations:** A text box explains that users can modify names or add descriptions to operations generated in the interface file by clicking the "Edit Operations..." button.

**Deployment properties:**

- A checked checkbox labeled "Specify a Java Authentication and Authorization Services (JAAS) alias security credential."
- A text field for "J2C Authentication Data Entry: \*" containing the value "widNode/wccpc11800".
- A dropdown menu for "Deploy connector project:" set to "On server for use by multiple applications".
- A text box for "Specify the settings used to connect to SAP Software at runtime:".
- A dropdown menu for "Connection properties:" set to "Use predefined connection properties".
- A text field for "JNDI Lookup Name: \*" containing the value "com/eis/wccpc11800AdapterCF".

At the bottom, there are navigation buttons: "< Back", "Next >", "Finish", and "Cancel". A help icon (?) is also present.

Figure 9-83 Service Generation and Deployment Configuration

12. Click **Edit Operations**.
13. As we make an existence check using the Retrieve operation, we adjust the operation name to fit better.
14. Change operation name retrieveSapDocumentTxn to existsSapMaterialTxn.
15. Enter SAPDocumentInterface as the interface name, leave all other values unchanged and click **Finish**.
16. Add a reference to the mediation flow component and wire it to the SAPDocumentInterface import.

17. Select **OK** to add matching references when prompted to do so.
18. Right-click the mediation flow and add the interface `DocumentSystemInterface`.
19. Right-click the mediation flow and generate an export of type SCA binding.
20. Right-click the mediation flow and select **Regenerate Implementation**.

The Assembly diagram must look similar to Figure 9-84:

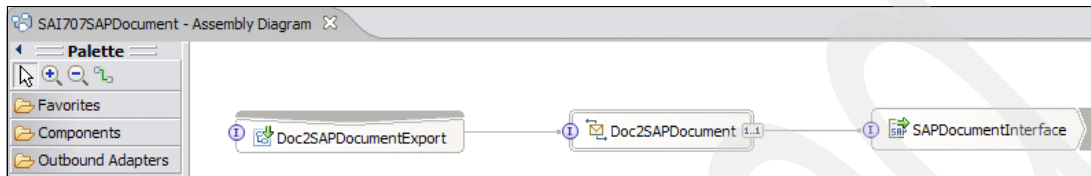


Figure 9-84 SAI707SAPDocument Assembly diagram

### ***Implementing the Document mediation module***

The next steps describe how to implement a suitable mediation flow that maps from the generic interface to the regarding SAP interface.

1. Double-click the mediation flow.
2. Create an operation connection between operation `createDocument` and `createSapDocumentTxn`.
3. Create an operation connection between operation `isExisting` and `existsSapDocumentTxn`.
4. Select the `createDocument` connection.
5. Place a XSL transformation node in the Request diagram and name it `transformSAPDOCUMENTRequest` (Figure 9-85 on page 351).
6. Wire the Input node to the in terminal of the XSL transformation.
7. Wire the out terminal of the XSL transformation to the Callout node.

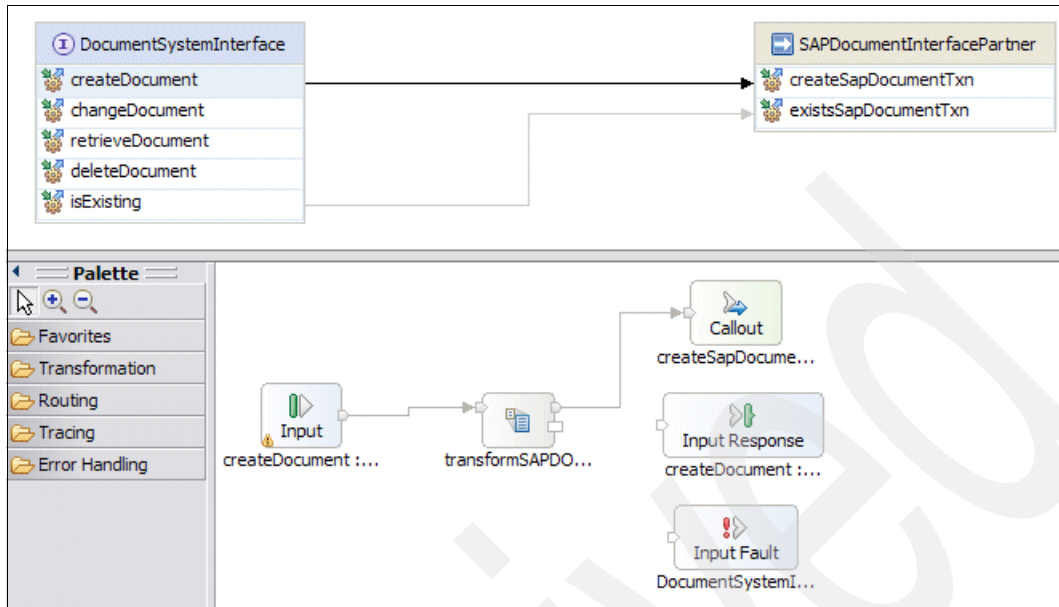


Figure 9-85 SAP Document mediation flow: Request diagram

8. Double-click the XSL transformation node transformSAPDOCUMENTRequest.
9. Enter a suitable name for the mapping file (for example, CreateDocument-Request) and click **Finish**.
10. Expand the target object.
11. Add a Move transformation between the following attributes:
  - Source DocumentNumber → Target SapDocumentdata/Documentnumber
  - Source ChangeNumber → Target SapDocumentdata/Ecnumber
  - Source Description → Target SapDocumentdata/Description
  - Source Effectivity/StartDateEffectivity → Target SapDocumentdata/ValidFrom
  - Source Attachment/Type → Target SapDocumentdata/Wsapapplication1
  - Source Attachment/File → Target SapDocumentdata/Docfile1

12. Add a conditional Move transformation between the following attributes. The condition is added in brackets [ ] :

a. Source Type → Target SapDocumentdata/Documenttype

```
[not((/body/createDocument/document/Document/Type = null) or  
(/body/createDocument/document/Document/Type = '' ) or  
(not(/body/createDocument/document/Document/Type)))]
```

b. Source Version → Target SapDocumentdata/Documentversion

```
[not((/body/createDocument/document/Document/Version = null) or  
(/body/createDocument/document/Document/Version = '' ) or  
(not(/body/createDocument/document/Document/Version)))]
```

c. Source Iteration → Target SapDocumentdata/Documentpart

```
[not((/body/createDocument/document/Document/Iteration = null) or  
(/body/createDocument/document/Document/Iteration = '' ) or  
(not(/body/createDocument/document/Document/Iteration)))]
```

d. Source Attachement/UserDefined → Target  
SapDocumentdata/Datacarrier1

```
[not((/body/createDocument/document/Document/Attachments/UserDefi  
ned = null) or  
(/body/createDocument/document/Document/Attachments/UserDefined =  
'') or  
(not(/body/createDocument/document/Document/Attachments/UserDefin  
ed)))]
```

13. Add a conditional Assign transformation (the condition is added in brackets [ ]) to the following target attributes:

a. SapDocumentdata/Documenttype → DES

```
[(/body/createDocument/document/Document/Type = null) or  
(/body/createDocument/document/Document/Type = '' ) or  
(not(/body/createDocument/document/Document/Type)))]
```

b. SapDocumentdata/Documentversion → -

```
[(/body/createDocument/document/Document/Version = null) or  
(/body/createDocument/document/Document/Version = '' ) or  
(not(/body/createDocument/document/Document/Version)))]
```

c. SapDocumentdata/Documentpart → 000

```
[(/body/createDocument/document/Document/Iteration = null) or  
(/body/createDocument/document/Document/Iteration = '' ) or  
(not(/body/createDocument/document/Document/Iteration)))]
```

d. SapDocumentdata/Datacarrier1 → INTERNET

```
[(/body/createDocument/document/Document/Attachments/UserDefined  
= null) or  
(/body/createDocument/document/Document/Attachments/UserDefined =  
'') or  
(not(/body/createDocument/document/Document/Attachments/UserDefin  
ed))]
```

**Note:** The pattern above is used to create conditional mappings and to define default values for cases when no values are entered in the source interface.

14. Add a Submap named Attachment2SAPDOC between the Source Document/Attachments → Target SapBapiDocumentCreate2/SapDocumentfiles attributes:

a. Within the submap add Move transformations to the following target attributes:

- Source Type → Target Wsapplication
- Source Name → Target Description
- Source File → Target Docfile

b. Within the submap add Assign transformations to the following target attributes:

- Originaltype → 01
- Activeversion → X

c. Within the submap add a conditional Move transformation to the Source UserDefined → Target Sourcedatacarrier attribute. The condition is added in brackets [ ].

```
[not((/Attachment/UserDefined = null) or (/Attachment/UserDefined  
= '') or (not(/Attachment/UserDefined)))]
```

d. Within the submap, add an Assign transformation to the Sourcedatacarrier. This is an INTERNET target attribute. The condition is added in brackets [ ].

```
[(/Attachment/UserDefined = null) or (/Attachment/UserDefined =  
'') or (not(/Attachment/UserDefined))]
```

15. Save the submap file.

16. Add a Submap named CompLink2SAPOBJ between the Source Document/ComponentLink → Target SapBapiDocumentCreate2/SapObjectlinksattributes:
  - a. Within the submap add Move transformations to the following target attributes:
    - Source ID → Target Objectkey
    - Source Description → Target Objetdescription
    - Source UserDefined → Target Objectlinkid
  - b. Within the submap add a conditional Move transformation to the Source Type → Target Objecttype attribute. The condition is added in brackets [ ] :  
`[not((/ComponentLink/Type = null) or (/ComponentLink/Type = '' ) or (not(/ComponentLink/Type)))]`
  - c. Within the submap add an Assign transformation to the Objecttype → MARA target attribute. The condition is added in brackets [ ] :  
`[(/ComponentLink/Type = null) or (/ComponentLink/Type = '' ) or (not(/ComponentLink/Type))]`
17. Save the submap file.
18. Save the mapping file.
19. Switch to the Response diagram.
20. Place a XSL transformation node in the Response diagram and name it transformDocCreateResp.
21. Place a XSL transformation node in the Response diagram and name it processException.
22. Wire the out terminal of the Callout Response node with the input of the XSL transformation transformDocCreateResp.
23. Wire the fail terminal of the Callout Response node with the input of the XSL processException.
24. Wire the output terminal of the XSL transformation transformDocCreateResp to the Input Response node.
25. Wire the output terminal of the XSL transformation processException to the Input Fault node.



The Response diagram must look similar to Figure 9-86:

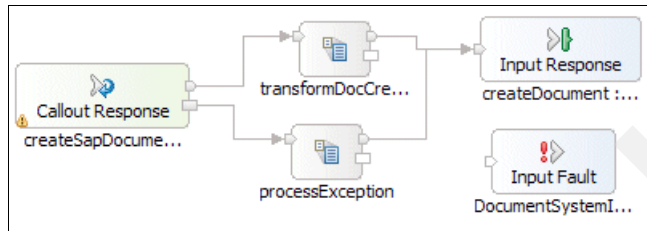


Figure 9-86 SAP Document mediation flow: Response diagram

26. Double-click the XSL transformation node transformDocCreateResp.

27. Enter a suitable name for the mapping file, for example, CreatedOCUMENT-Response and click **Finish**.

28. Expand the target object.

29. Add a Move transformation between the following attributes:

- Source SapReturn/Type → Target Result/Type
- Source SapReturn/Id → Target Result/Id
- Source SapReturn/Number → Target Result/Number
- Source SapReturn/Message → Target Result/Message
- Source SapReturn/LogNo → Target Result/LogNo
- Source SapReturn/MessageV1 → Target Result/MessageV1
- Source SapReturn/MessageV2 → Target Result/MessageV2
- Source SapReturn/MessageV3 → Target Result/MessageV3
- Source SapReturn/MessageV4 → Target Result/MessageV4

30. Save the mapping file.

**Note:** This component provides BAPI modules that deliver all response in an SAP Return structure. Within the mediation flow additional exception handling can be included. For the scenario in the Redbooks publication the processing of the standard return structure is sufficient.

31. Double-click the XSL transformation node processException.

32. Enter a suitable name for the mapping file (for example, createException) and click **Finish**.

33. Expand the target object.

34. Add a Move transformation between the following attributes:

- Source SapReturn/Type → Target Result/Type
- Source SapReturn/Id → Target Result/Id
- Source SapReturn/Number → Target Result/Number
- Source SapReturn/Message → Target Result/Message
- Source SapReturn/Message → Target ErrorMessage/ERPPrimary
- Source SapReturn/LogNo → Target Result/LogNo
- Source SapReturn/MessageV1 → Target Result/MessageV1
- Source SapReturn/MessageV1 → Target ErrorMessage/ERPPrimary
- Source SapReturn/MessageV2 → Target Result/MessageV2
- Source SapReturn/MessageV3 → Target Result/MessageV3
- Source SapReturn/MessageV4 → Target Result/MessageV4

35. Save the mapping file.

36. Save the mediation flow.

**Note:** The response flow in this mediation is not generating a Fault as the BAPI return structure contains sufficient error message to handle errors in the backend appropriately.

### ***Implementing the isExisting operation***

Perform the following steps to implement the isExisting operation (see Figure 9-87 on page 357).

1. Select the createDocument connection.
2. Place a XSL transformation node in the Request diagram and name it transformExistenceReq.
3. Wire the Input node to the in terminal of the XSL transformation.
4. Wire the out terminal of the XSL transformation to the Callout node.

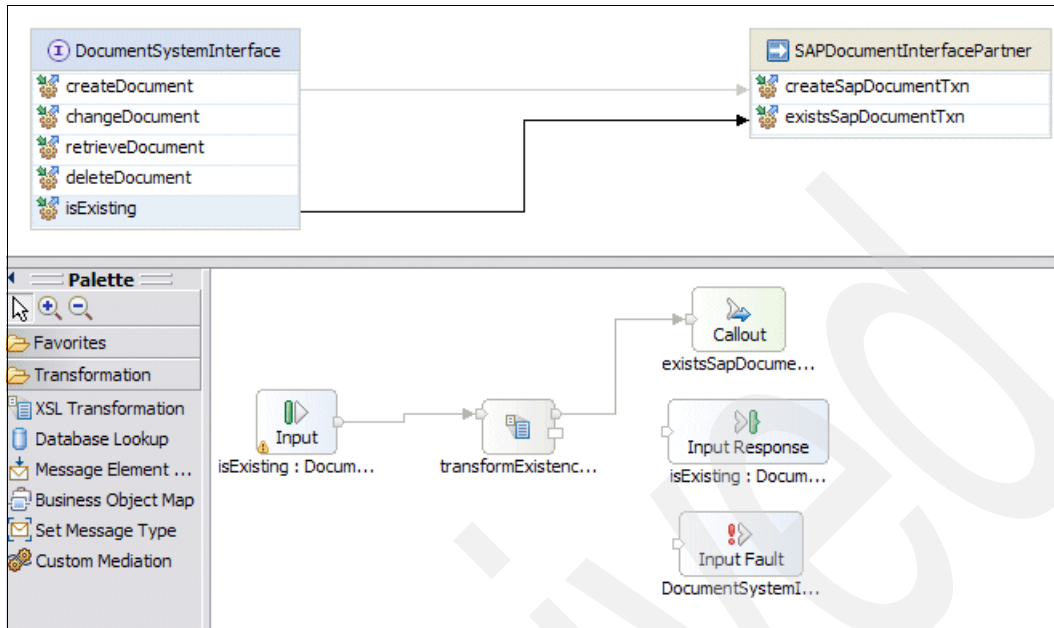


Figure 9-87 SAP Document mediation flow: Request diagram isExisting operation

5. Double-click the XSL transformation node transformExistenceReq.
6. Enter a suitable name for the mapping file, for example, ExistDocument-Request and click **Finish**.
7. Expand the target object.
8. Add a Move transformation between the Source documentnumber → Target SapBapiDocumentExistencecheck/Documentnumber attributes.
9. Add conditional Move transformations between the following attributes. The condition is added in brackets [ ].
  - a. Source documenttype → Target SapBapiDocumentExistencecheck/Documenttype  
`[not((/body/isExisting/documenttype = null) or (/body/isExisting/documenttype = ''))]`
  - b. Source documentpart → Target SapBapiDocumentExistencecheck/Documentversion  
`[not((/body/isExisting/documentpart = null) or (/body/isExisting/documentpart = ''))]`

- c. Source documentversion → Target  
SapBapiDocumentExistencecheck/Documentpart  
[not((/body/isExisting/documentversion = null) or  
(/body/isExisting/documentversion = ''))]
- 10. Add conditional Assign transformations to the following target attributes. The condition is added in brackets [ ].
  - a. SapBapiDocumentExistencecheck/Documenttype → DES  
[(/body/isExisting/documenttype = null) or  
(/body/isExisting/documenttype = '')]
  - b. SapBapiDocumentExistencecheck/Documentversion → -  
[(/body/isExisting/documentpart = null) or  
(/body/isExisting/documentpart = '')]
  - c. SapBapiDocumentExistencecheck/Documentpart → 000  
[(/body/isExisting/documentversion = null) or  
(/body/isExisting/documentversion = '')]

**Note:** The pattern above is used to create conditional mappings and to define default values for cases when no values are entered in the source interface.

- 11. Save the mapping file.
- 12. Switch to the Response diagram.
- 13. Place a XSL transformation node in the Response diagram and name it transformDocExistsResp.
- 14. Place a XSL transformation node in the Response diagram and name it processException.
- 15. Wire the out terminal of the Callout Response node with the input of the XSL transformation transformDocCreateResp.
- 16. Wire the fail terminal of the Callout Response node with the input of the XSL processException.
- 17. Wire the output terminal of the XSL transformation transformDocCreateResp to the Input Response node.
- 18. Wire the output terminal of the XSL transformation processException to the Input Fault node.

The Response diagram must look similar to Figure 9-88 on page 359.

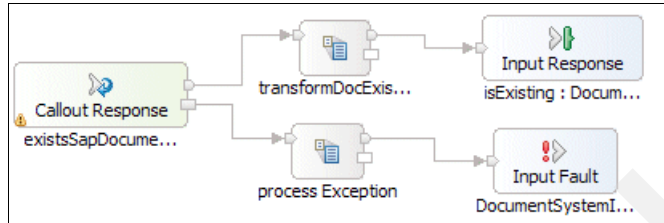


Figure 9-88 SAP Document mediation flow: Response diagram isExisting operation

19. Double-click the XSL transformation node transformDocExistsResp.
20. Enter a suitable name for the mapping file (for example, ExistsCreateDOCUMENT-Response) and click Finish.
21. Expand the target object.
22. Add two conditional Assign transformations to the following target attributes. The condition is added in brackets [ ].
  - a. isExistingResponse/result → true  
`[/body/isExistingSapDocumentWrapperResponse/isExistingSapDocumentWrapperOutput/SapDocumentWrapper/SapBapiDocumentExistencecheck/Documentexists = 'X']`
  - b. isExistingResponse/result → false  
`[(not(/body/isExistingSapDocumentWrapperResponse/isExistingSapDocumentWrapperOutput/SapDocumentWrapper/SapBapiDocumentExistencecheck/Documentexists)) or (/body/isExistingSapDocumentWrapperResponse/isExistingSapDocumentWrapperOutput/SapDocumentWrapper/SapBapiDocumentExistencecheck/Documentexists = null)]`
23. Save the mapping file.
24. Double-click the XSL transformation node processException.
25. Enter a suitable name for the mapping file (for example, existsException) and click **Finish**.
26. Expand the target object.
27. Add an Assign transformation to the following attributes:
  - FaultName → SAP exception
  - MessageText → Document existence call failed.
28. Save the mapping file.
29. Save the mediation flow.

The mediation module is now complete and can be deployed on the runtime server for testing.

## Develop the PLM to SAP integration module

This integration module contains integration artifacts that connect the PLM services assets to the SAP integrating mediation modules that were developed in prior sections of this chapter. The module exposes an SCA Export node that is linked to a Java mapping component. Within this Java component, a mapping between the PLM services object to the generic object is done.

Beside the SCA Export, four SCA Import nodes are located in the Assembly diagram that are connected to the SCA Exports in the particular SAP integrating mediation modules. The key component is the BPEL process named TransferECRtoERPSytem that is responsible to call the SAP functions in an appropriate order. The development of the BPEL process is described in a separate section of this chapter.

### Creating a PLM to SAP integration module

Follow the steps below to create the PLM to SAP integration module.

1. Go to **File** → **New** → **Module** and name the module SAI707PLM2SAP (Figure 9-89) and the mediation component Doc2SAPDocument. Select **Next**.

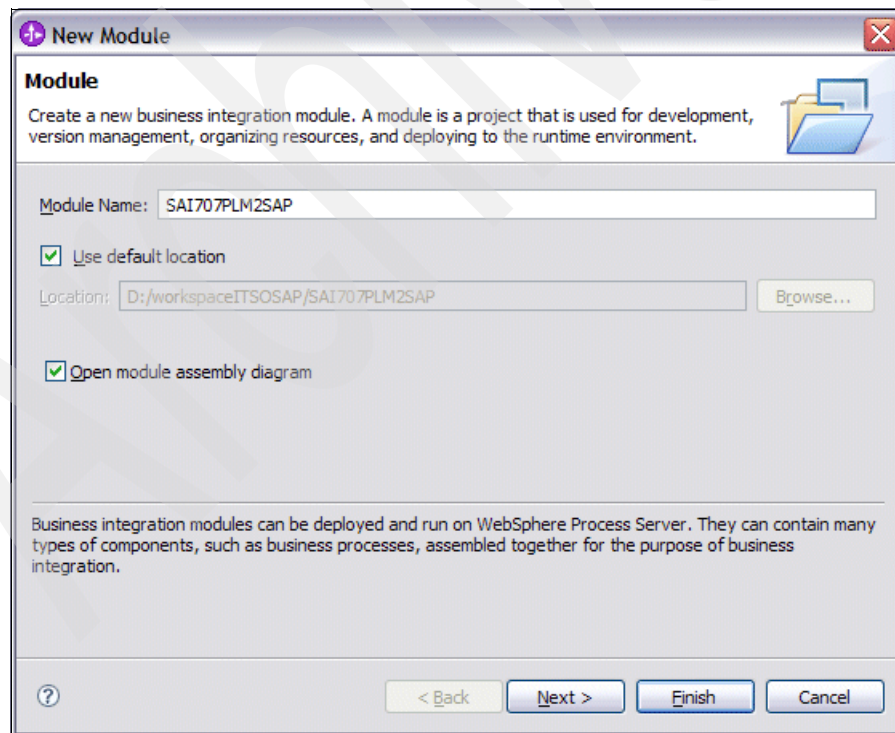


Figure 9-89 Create module SAI707PLM2SAP

2. Select the check box to reference the ITSOPMLLibrary. Click **Finish**.
3. In the Assembly Diagram, add the following nodes on the canvas:
  - Java component with the name MapDataAndCallSAP.
  - BPEL component with the name TransferECRtoERPSytem.
  - Import node with the name ECNSystem.
  - Import node with the name MaterialMasterSystem.
  - Import node with the name BOMSystem.
  - Import node with the name DocumentSystem.
4. Add the interface PLMServicesToSAP to the MapDataAndCallSAP component.
5. Add the interface PLMERPInterface to the TransferECRtoERPSytem component.
6. Add the interface EngineeringChangeManagementInterface to the ECNSystem component.
7. Add the interface MaterialMasterInterface to the MaterialMasterSystem component.
8. Add the interface BillOfMaterialInterface to the BOMSystem component.
9. Add the interface DocumentSystemInterface to the DocumentSystem component.
10. Add the reference PLMERPInterfacePartner to the MapDataAndCallSAP component.
11. Add the reference EngineeringChangeManagementInterfacePartner to the TransferECRtoERPSytem component.
12. Add the reference MaterialMasterInterfacePartner to the TransferECRtoERPSytem component.
13. Add the reference BillOfMaterialInterfacePartner to the TransferECRtoERPSytem component.
14. Add the reference DocumentSystemInterfacePartner to the TransferECRtoERPSytem component.
15. Right-click each node and use the Wire to Existing function to wire the references to the corresponding interfaces.
16. Right-click the MapDataAndCallSAP component and select to generate a SCA binding export node.

17. At this point the Assembly diagram must look similar to Figure 9-90.

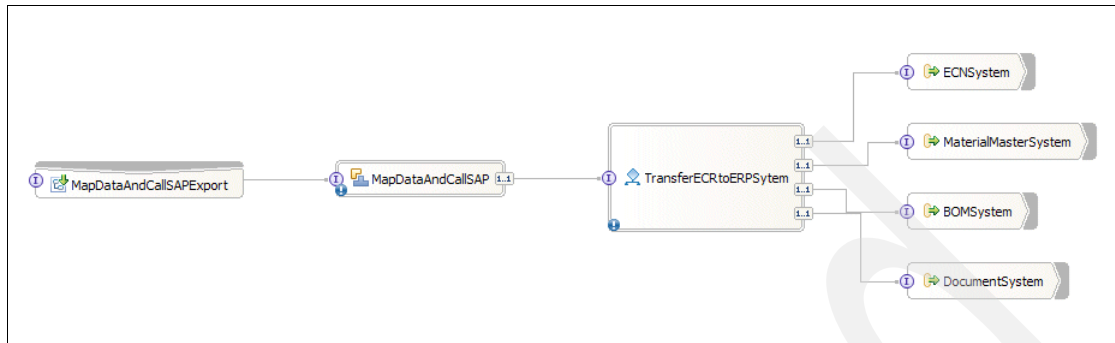


Figure 9-90 Assembly diagram SAI707PLM2SAP

18. Right-click the MapDataAndCallSAP component and select **Generate Implementation ... → Java**.

- In the Create Package window, create a new package named truckinc.sapintegration (see Figure 9-91).
- Select the new package and click **OK**.

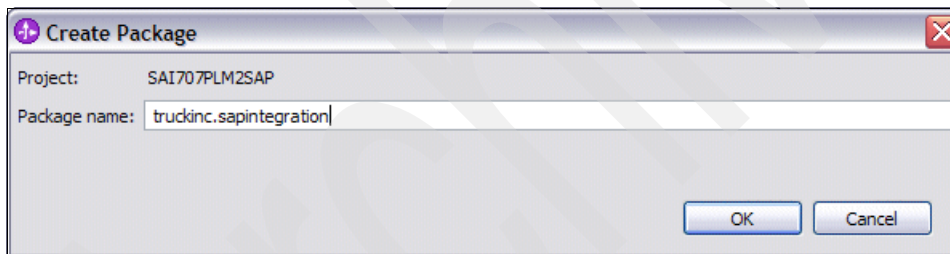


Figure 9-91 Create new package for Java mapping implementation file



19. When asked to create Java interfaces for existing WSDL interfaces (Figure 9-92), select **Yes**.

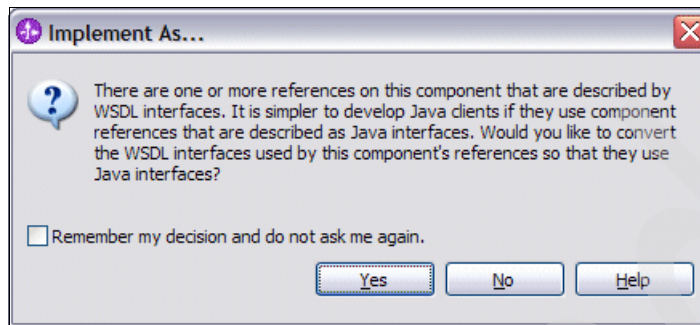


Figure 9-92 Create Java interfaces for existing WSDL interfaces

20. Replace the implementation of MapDataAndCallSAPImpl.java with the content of the MapDataAndCallSAPImpl.java file that can be found in the SAI707PLM2SAP folder in the directory where you extracted the downloaded Redbook247593\_lab.zip lab file (see Appendix A, “Additional material” on page 465).
21. Review the MapDataAndCallSAPImpl.java content. This class traverses the passed PLM Services data and creates the data structures that is passed to SAP:
- ECN
  - MultiParts
  - MultiBOMs
  - MultiDocuments
- Dependent on the simulateSAP flag, this data is written to the console or passed to the SAP system.
22. Right-click the TransferECRtoERPSytem component and select **Generate Implementation**.
- a. In the pop-up window, choose to create a new package named ecrprocess.
  - b. Select the new package and click **OK**.

23. At this point, the BPEL editor must look similar to Figure 9-93:

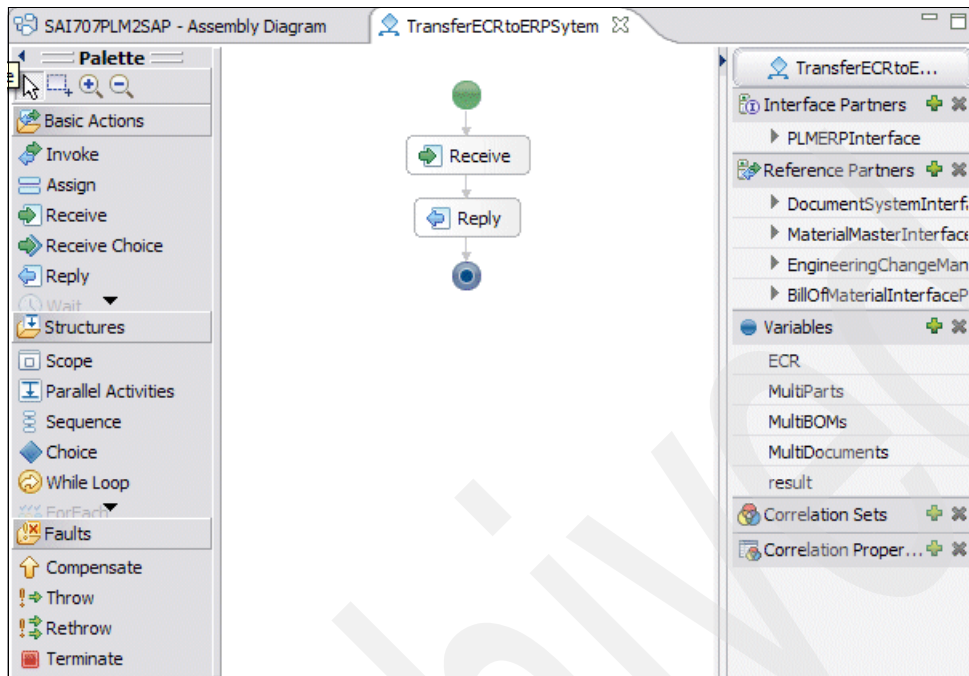


Figure 9-93 ECR BPEL process initial version

The BPEL skeleton is created including the corresponding variables, partner links and interface signatures. The next section describes in detail how to implement a custom PLM to SAP ERP process by adding more logic into the BPEL process. The final process calls various operations in a meaningful sequence and react on the response of the called SAP system approbated.

Save the BPEL file as it is filled with logic in the next section. Do the last steps to finalize the module.

1. Open the Assembly diagram of the SAI707PLMSAP module.
2. Right-click each Import node and select to generate a SCA binding.
3. In the Details window of each Import node, choose a suitable interface to bind. In the Redbook setup, there must be only one interface per binding available to select.
4. Save the mediation module.

## Develop PLM to SAP integration process

This section describes in more detail how to develop a custom ECR process that integrates with an SAP ERP system.

The BPEL process uses existing mediation modules that were created earlier and that expose their functionality through SCA bindings. The process is as follows:

1. The ECR process creates a Engineering Change Number in the SAP system.
2. If this is successful, it creates all parts that are provided as an array in an input variable.
3. After successfully finishing this section, another array holding BOM objects is processed, and the included objects are transferred to the SAP system.
4. The ECR transfers the document objects that are located in an input variable array to the SAP system.
5. A result object is used to respond back to the calling component how the SAP system interactions executed.

The process contains various exit points. For example, when an error occurs during creation of the Engineering Change Number in SAP, it doesn't make sense to process further.

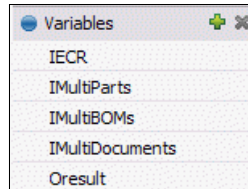
**The ECR process described above is an example:** It highlight potentials integration capabilities. Based on specific customer requirements the process is different for each customer scenario. To keep the scenario simple and understandable the material, BOM, and document change options are not implemented in this process. In cases where master data is already available in the SAP system, an additional sub-process including Human Tasks or Business Rules can be incorporated to ensure that the change is not creating inconsistencies in the SAP system.

To make the execution of the BPEL process more visible, many Log activities are included that are not necessary for the core purpose of the process. These activities can be skipped when doing productive implementations.

### ***Creating the exemplary ECR process in BPEL***

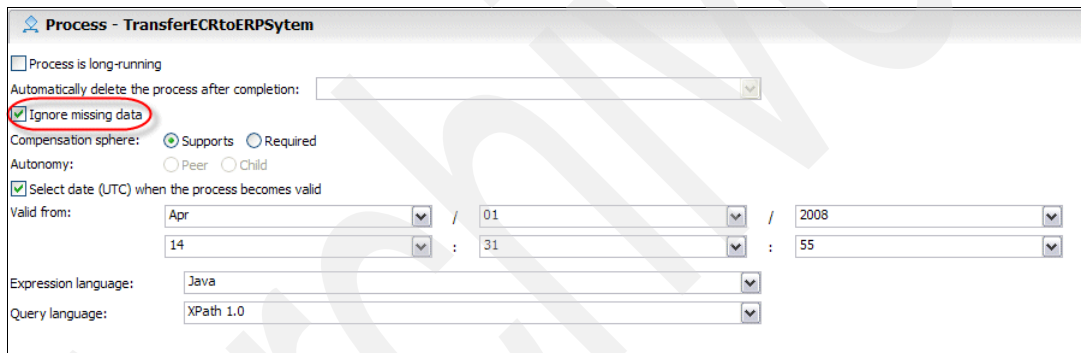
Follow the steps below to create the exemplary ECR process in BPEL:

1. Open the TransferECRtoERPSystem process in the BPEL editor.
2. Change the generated variable names and add prefixes I and O to identify input and output variables as shown in Figure 9-94.



*Figure 9-94 Add prefixes to Input and Output variables*

3. Open the properties window of the BPEL process and check the Ignore missing data check box. See Figure 9-95.



*Figure 9-95 Adjust attribute Ignore missing data*

4. Add a scope between the Receive and the Reply node and name it Process ECN.
5. Place a snippet in the scope and name it add current date in actual ECR.

6. Insert the following visual snippet in the Details window (Figure 9-96):

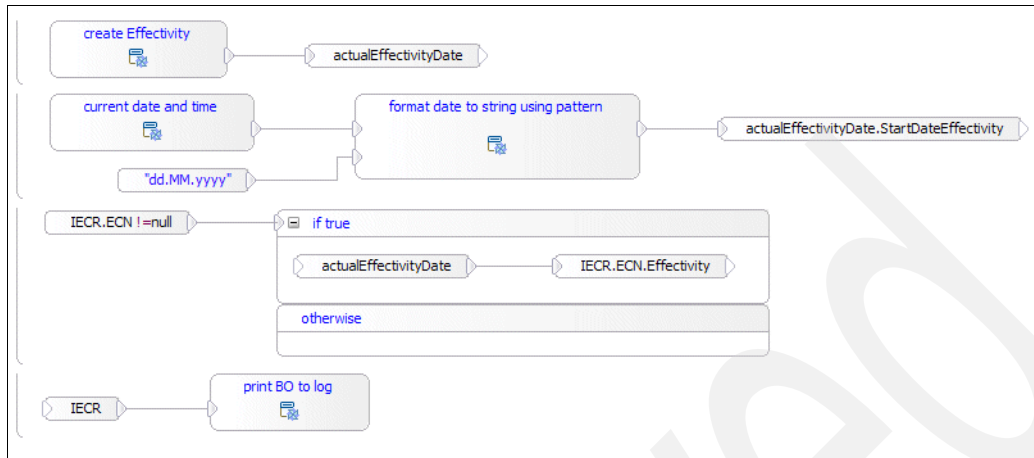


Figure 9-96 Visual snippet in the Details window: add actual date to ECR

7. Create a variable with the name ECNCreationResult and enter ResultBG.
8. Place an Invoke in the scope and name it Create ECN.
9. Set the properties of the invoke node as shown in Figure 9-97:

	Name	Variable	
Input(s)	ecn	IECR	...
Output(s)	result	ECNCreationResult	...

Figure 9-97 Properties of the invoke node: Create ECN

10. Place a snippet after the invoke and name it Log Create ECN result.

11. Insert the following visual snippet in the Details window (Figure 9-98):

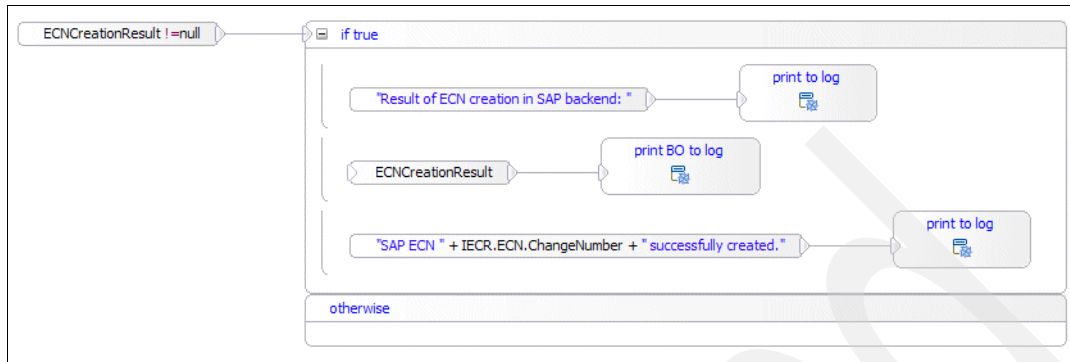


Figure 9-98 Visual snippet in the Details window: Log Create ECN result

12. Place an Assign after the snippet and name it Add to ECR Result Object.

13. Insert the following assignments (Figure 9-99) in the Details window:

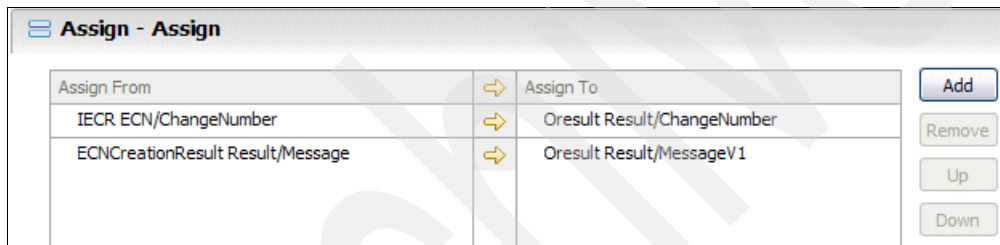


Figure 9-99 Assignments in the Details window: Add to ECR Result Object

14. Right-click the invoke node and add a fault handler.

15. Place a snippet in the fault handler and name it Error Creating ECN.

16. Insert the following visual snippet (Figure 9-100) in the Details window:

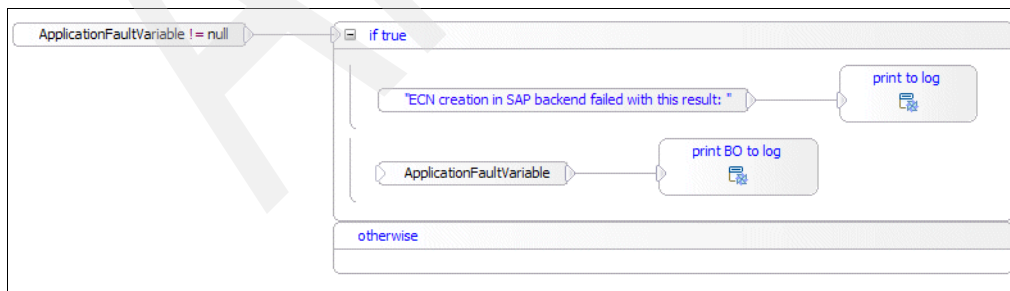


Figure 9-100 Visual snippet in the Details window: Error Creating ECN

17. Place an Assign after the snippet and name it Prepare Response.

18. Insert the following assignments (Figure 9-101) in the Details window:

Assign - Assign	
Assign From	Assign To
E	Oresult Result/Type
ECN creation in SAP failed.	Oresult Result/Message
ApplicationFaultVariable ApplicationFault FaultName	Oresult Result/MessageV1
ApplicationFaultVariable ApplicationFault FaultName	Oresult Result/MessageV2
ApplicationFaultVariable ApplicationFault MessageText	Oresult Result/MessageV3
ApplicationFaultVariable ApplicationFault RootException	Oresult Result/MessageV4

Figure 9-101 Assignments in the Details window: Prepare Response

19. Place a snippet after the Assign node in the fault handler and name it Dump Result Object.

20. Insert the following visual snippet (Figure 9-102) in the Details window:

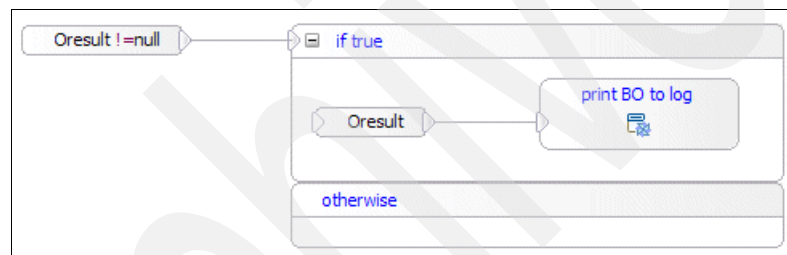


Figure 9-102 Visual snippet in the Details window: Dump Result Object

21. Place a Reply node after the snippet and set the properties as shown in Figure 9-103.

Reply - Reply1							
Partner:*	PLMERPInterface <span>Browse...</span>						
Interface:*	PLMERPInterface						
Operation:*	submitECR2ERP						
<input checked="" type="checkbox"/> Use Data Type Variables							
<span>Output(s)</span>	<table border="1"> <thead> <tr> <th>Name</th> <th>Variable</th> </tr> </thead> <tbody> <tr> <td>result</td> <td>Oresult</td> </tr> <tr> <td colspan="2"><span>...</span></td> </tr> </tbody> </table>	Name	Variable	result	Oresult	<span>...</span>	
Name	Variable						
result	Oresult						
<span>...</span>							
Reply Type: <input checked="" type="radio"/> Normal <input type="radio"/> Fault							

Figure 9-103 Place a Reply node after the snippet

22. Place a Terminate node after the Reply node.

23. At this stage the BPEL process must look similar to Figure 9-104.

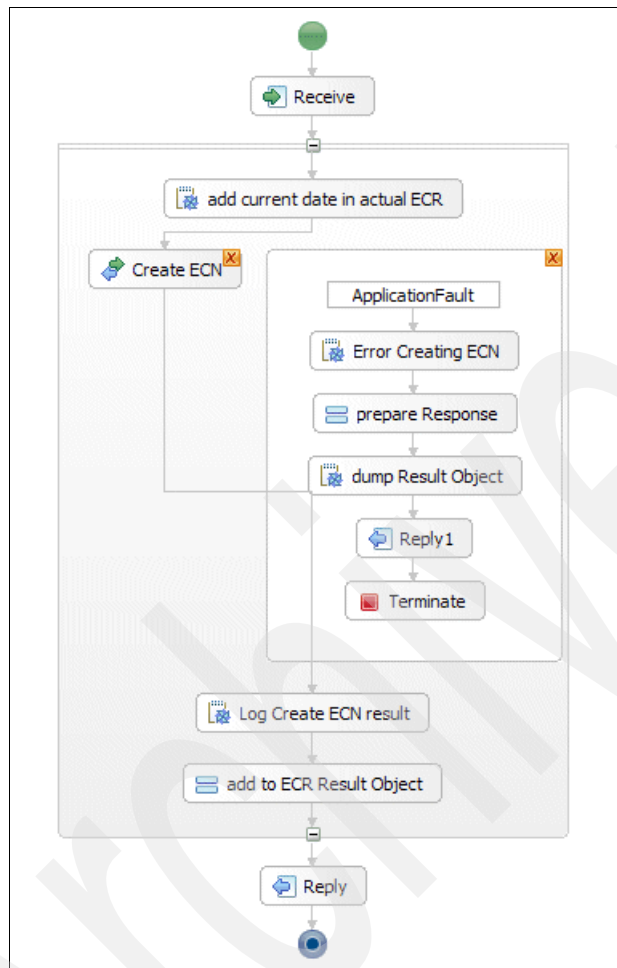


Figure 9-104 ECR BPEL process second version

At this stage, the BPEL process uses one reference partner to create a ECN in the SAP system. In case of failure, the exception is processed and a corresponding error result object is returned to the caller.



### Creating the core skeleton

The following procedure details the creation of the core skeleton. As the process receives most of the objects in arrays, the BPEL construct `forEach` is used to execute several steps for each array item. Between the `forEach` nodes, snippets are placed to create log entries for demonstration purpose only.

1. Open the `TransferECRtoERPSystem` process in the BPEL editor.
2. Minimize the scope `Process ECN`.
3. Place a snippet after the scope area and name it `Log Parts section`.
4. Insert the following visual snippet (Figure 9-105) in the `Details` window:

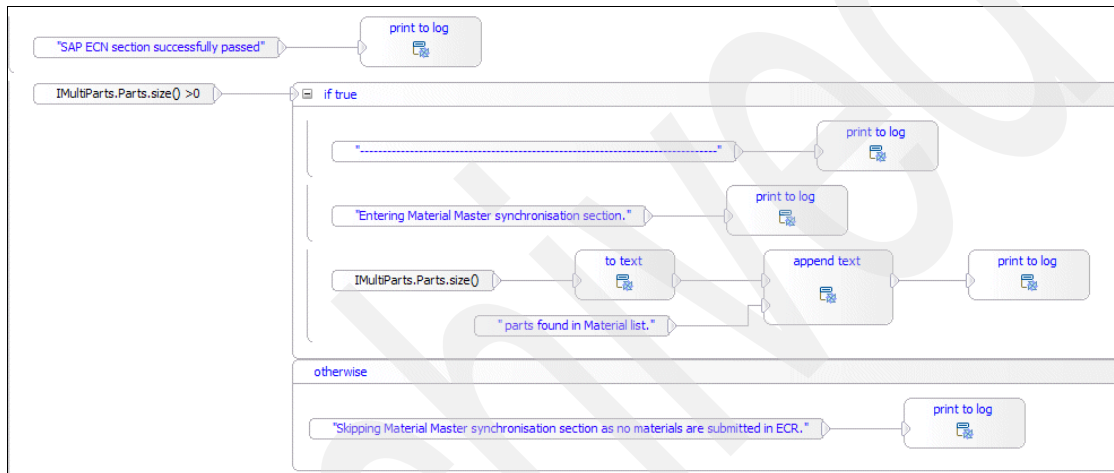


Figure 9-105 Visual snippet in the `Details` window: `Log Parts section`

5. Place a snippet after the scope area and name it `Log BOM section`.

6. Insert the following visual snippet (Figure 9-106) in the Details window:

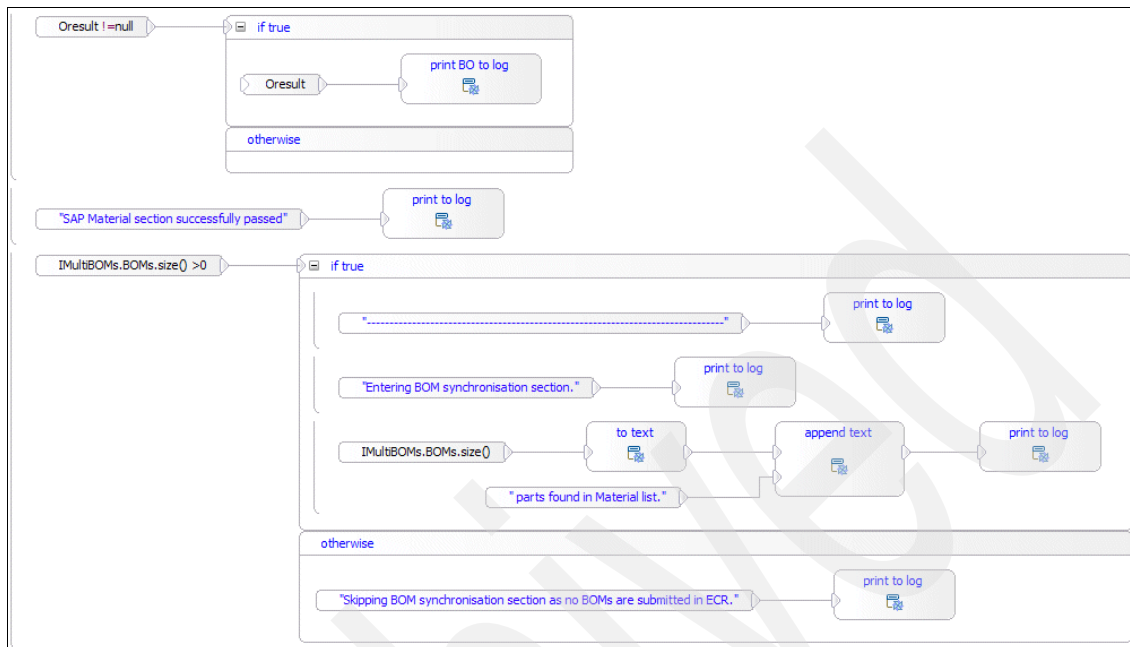


Figure 9-106 Visual snippet in the Details window: Log BOM section

7. Place a snippet after the scope area and name it Log Document section.

8. Insert the following visual snippet (Figure 9-107) in the Details window:

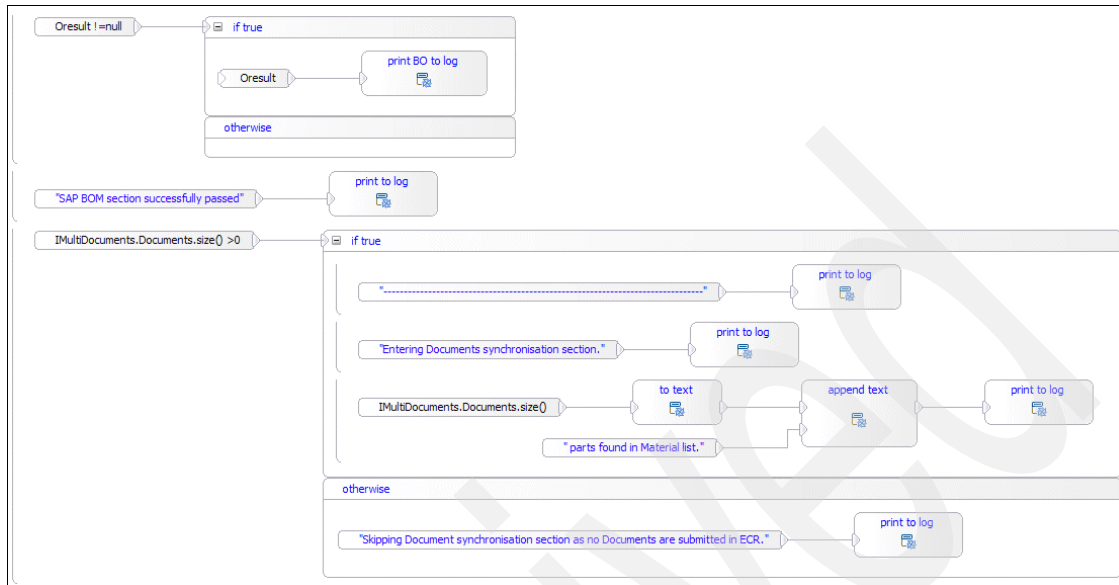


Figure 9-107 Visual snippet in the Details window: Log Document section

9. Place an Assign after the snippet Log Document section and name it Prepare Success Result.

10. Insert the following assignments (Figure 9-108) in the Details window:

Assign - Assign		
Assign From		Assign To
S	⇒	Oresult Result/Type
ECR process successfully executed.	⇒	Oresult Result/Message

Figure 9-108 Assignments in the Details window: Prepare Success Result

11. Place a snippet after the Assign node and name it Dump Result Object.

12. Insert the following visual snippet (Figure 9-109) in the Details window:

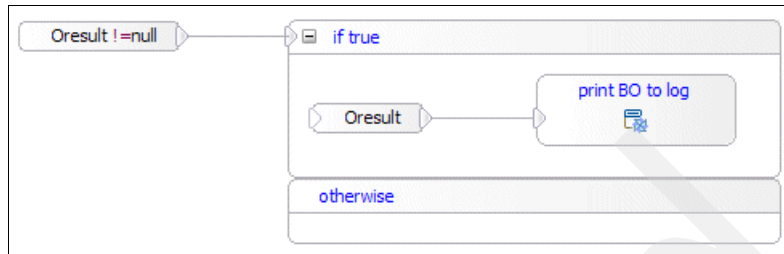


Figure 9-109 Visual snippet in the Details window: Dump Result Object

13. Place a forEach node after the snippet Log Parts section and name it Process parts.

14. Set the properties of this forEach node as shown in Figure 9-110.

A screenshot of the 'ForEach - Process Parts' configuration window. At the top, it says 'ForEach - Process Parts' with a small icon. Below this, 'Execution of iterations:' has two radio buttons: 'Sequential' (selected) and 'Parallel'. Underneath, 'Index-Variable Name:' is followed by a text box containing 'Index'. A section titled 'Iteration' with a dropdown arrow is expanded. It contains the text 'Define the bounds of the range to iterate over by specifying an iteration type.' and 'Type: Array (dynamic bounds)'. At the bottom, there is a list box containing two items: 'IMultiParts' (with a blue circle icon) and 'Parts' (with a folder icon).

Figure 9-110 Properties of the forEach node: Process Parts

15. Place a forEach node after the snippet Log BOM section and name it Process BOMs.

16. Set the properties of this forEach node as shown in Figure 9-111.

**ForEach - Process BOMs**

Execution of iterations: ☒ Sequential ☐ Parallel

Index-Variable Name:

**Iteration**

Define the bounds of the range to iterate over by specifying an iteration type.

Type: Array (dynamic bounds)

☐ IMultiBOMs  
☒ BOMs

Figure 9-111 Properties of the forEach node: Process BOMs

17. Place a forEach node after the snippet Log Documents section and name it Process Documents.

18. Set the properties of this forEach node as shown in Figure 9-112.

**ForEach - Process Documents**

Execution of iterations: ☒ Sequential ☐ Parallel

Index-Variable Name:

**Iteration**

Define the bounds of the range to iterate over by specifying an iteration type.

Type: Array (dynamic bounds)

☐ IMultiDocuments  
☒ Documents

Figure 9-112 Properties of the forEach node: Process Documents

19. Save the process.

The forEach nodes must be flagged with error signs as no content is included. The content is filled in the next section. The BPEL file in its current stage must look similar to Figure 9-113.

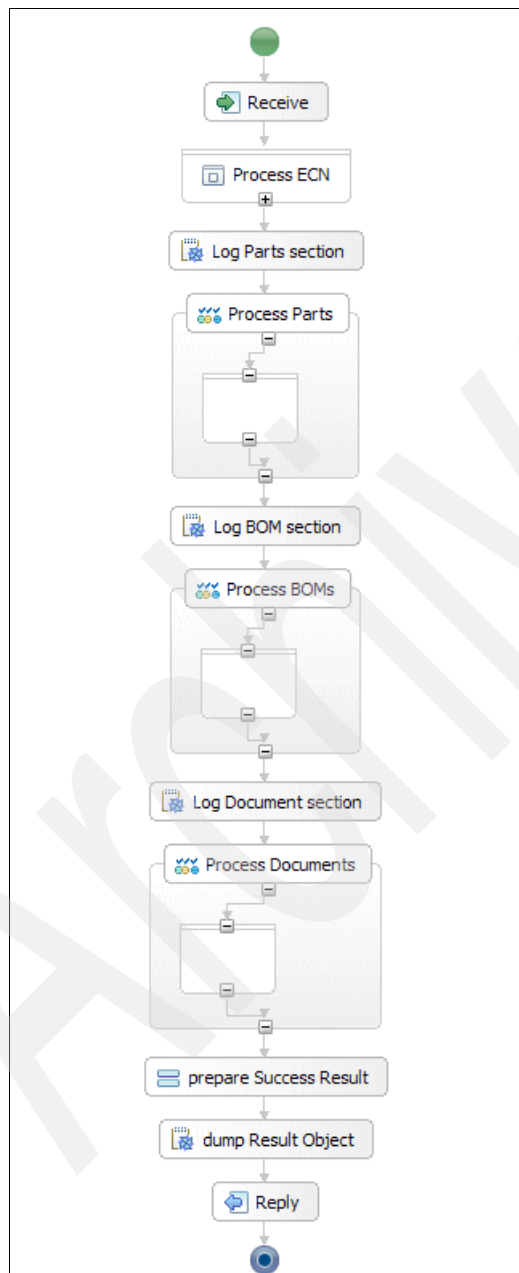


Figure 9-113 ECR BPEL process second version

### ***Processing Parts forEach section configuration***

The BPEL nodes that are described in this section all go into the Process Parts forEach section block. After adding all required nodes the Process Parts forEach section must look similar to Figure 9-114.

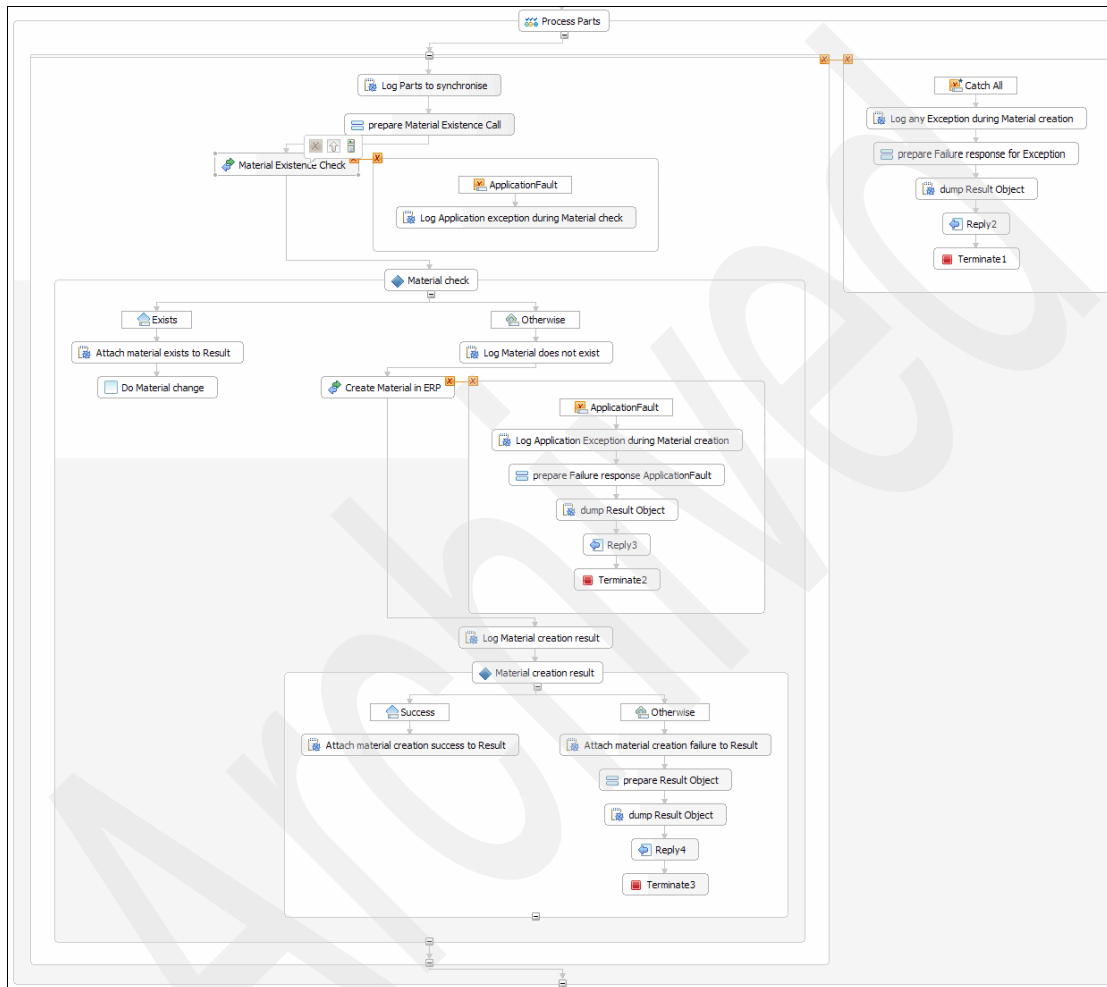


Figure 9-114 *forEach* section Process Parts

Follow these steps to get to this configuration:

1. Add a fault handler to catch all faults to top level scope of the forEach node Process Parts.
2. Add the following nodes to the CatchAll case:
  - a. Add a snippet node and name it Log any Exception during Material creation.
  - b. Insert the following logic (Figure 9-115) in the Details window of the snippet.

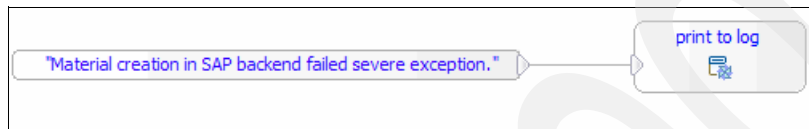


Figure 9-115 Log any Exception during Material creation snippet: Details

- c. Add a Assign node with the name Prepare Failure response for Exception.
- d. Insert the following assign rules (Figure 9-116) in the Details window of the Assign node.

Assign - prepare Failure response for Exception	
Assign From	Assign To
E	OECRResult Result/Type
Sever exception caught during material processing.	OECRResult Result/Message

Figure 9-116 Assign rules in the Details window: Prepare Failure response for Exception

- e. Insert the following logic (Figure 9-117) in the Details window of the snippet.

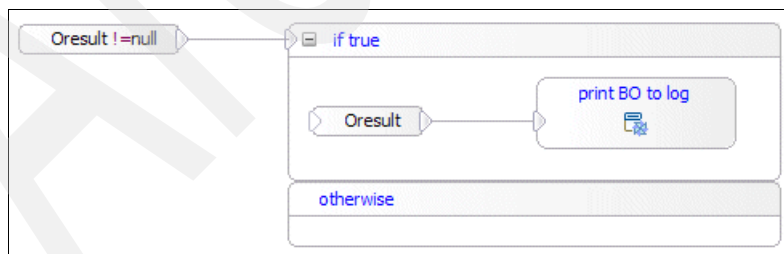


Figure 9-117 Insert logic in the Details window of the snippet: Dump Result Object



- f. Add a Reply node after the snippet and set the properties as shown in Figure 9-118:

Name	Type	Variable
Output(s)	result	ResultBG
		OECRRResult

Figure 9-118 Reply node: Details

- g. Place a Terminate node after the Reply node. The CatchAll fault handler section is done.
3. Add a snippet node with the name Log Parts to Synchronize into the Process Parts scope canvas.
4. Insert this logic (Figure 9-119) in the Details window of the snippet.

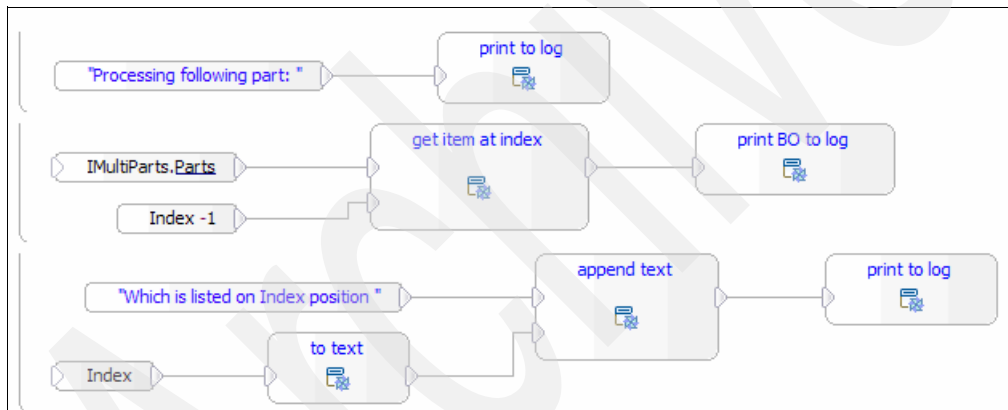


Figure 9-119 Insert logic in the Details window of the snippet: Log Parts to Synchronize

5. Add the following variables to the BPEL process:
  - ActuaPartNr from type string
  - ActualPlant from type string
  - ActualPart from type PartBG
  - MaterialExistenceResult from type ResultBG.
  - MaterialCreationResult from type ResultBG.
6. Add a Assign node with the name Prepare Material Existence Call.

- Insert the following assign rules (Figure 9-120) in the Details window of the Assign node.

Assign From	Assign To
1000	ActualPlant
IMultiParts Parts[bpws:getVariableData('Index')]/PartNumber	ActualPartNr
IMultiParts Parts[bpws:getVariableData('Index')]	ActualPart Part

Figure 9-120 Assign rules in the Details window of the Assign node: Prepare Material Existence Call

- Add an Invoke node with the name Material Existence Check.
- Insert the content shown in Figure 9-121 in the Details window of the invoke node.

Partner: \* MaterialMasterInterfacePartner Browse...

Interface: \* MaterialMasterInterface

Operation: \* isExisting

☒ Use data type variables mapping

	Name	Type	Variable
Input(s)	materialID	string	ActualPartNr
	plant	string	ActualPlant
Output(s)	result	ResultBG	MaterialExistenceResult

Figure 9-121 Invoke node Material Existence Check: Details

- Add a fault handler for the ApplicationFault to the invoke node Material Existence check.

11. Add the following nodes to the ApplicationFault case:

- a. Add a snippet node and name it Log Application exception during Material check.
- b. Insert the following logic (Figure 9-122) in the Details window of the snippet.

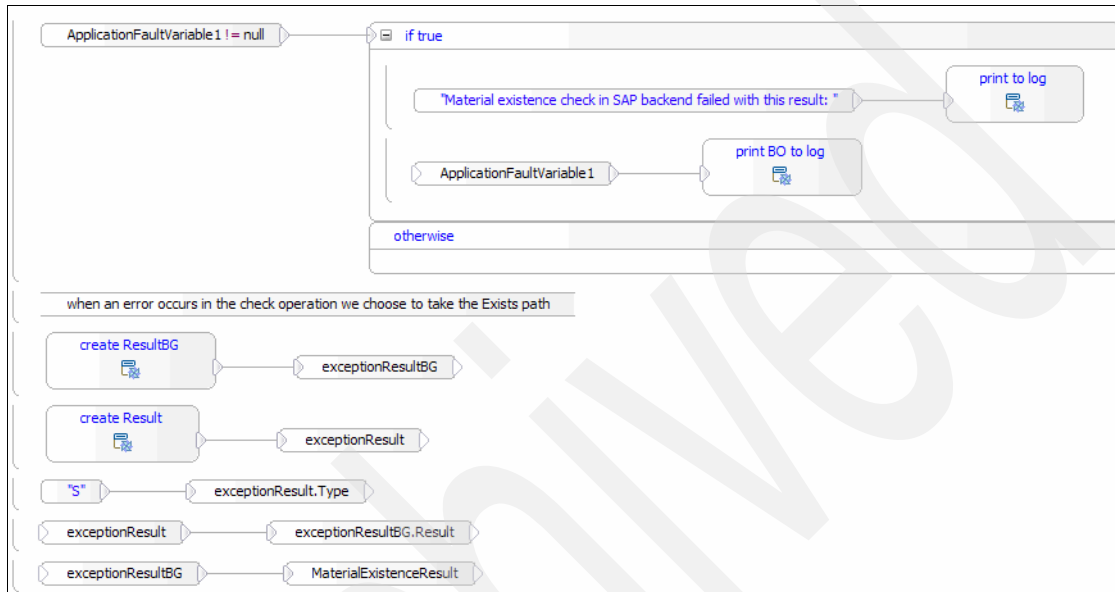


Figure 9-122 Insert logic in the Details window of the snippet: Log Application exception during Material check

12. Add a Choice node and name it Material check.

13. Add a new case to the choice with the name Exists.

14. Insert the following expression (Figure 9-123) for this new case in the Details window.

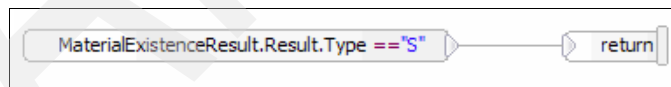


Figure 9-123 Insert expression in the Details window: Material check choice node: Exists case

15. Add an otherwise case to the choice node,

16. Add a snippet node to the Exists case and name it Attach material exists to Result.

17. Insert the following logic (Figure 9-124) in the Details window of the snippet.

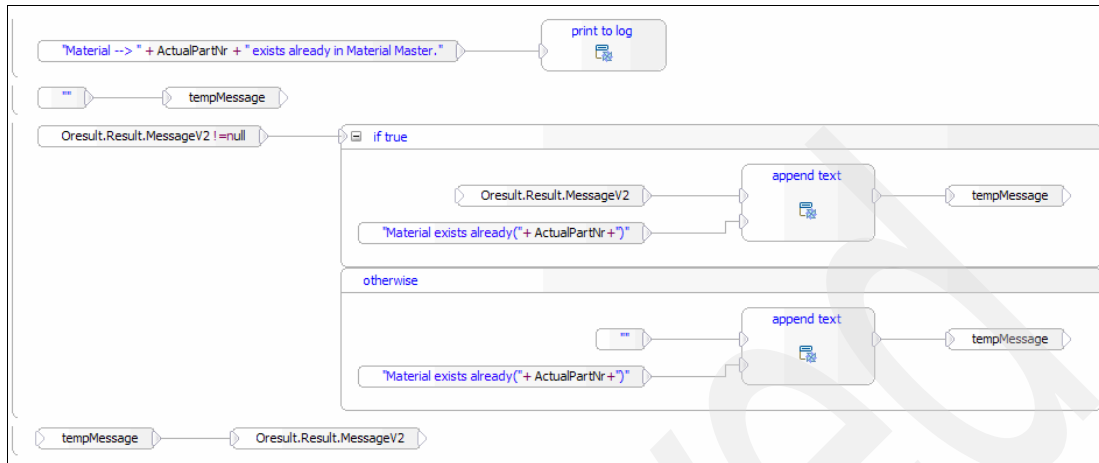


Figure 9-124 Insert logic in the Details window of the snippet: Attach material exists to Result

18. Add an Empty Action node after the snippet and name it Do Material change.

**Note:** This empty action represents the change business logic that can be attached here. Depending on what must be changed when a material already exists, a customized process can be triggered to enable a flexible and consistent integration of the material master SAP system. It is likely that such a dedicated change process includes business rules or human tasks to define exactly which attributes can be changed and to verify this using human approvals or automated business rules.

19. Add a snippet node to the Otherwise case and name it Log Material does not exist.

20. Insert the following logic (Figure 9-125) in the Details window of the snippet.

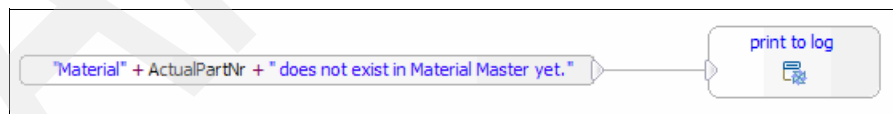


Figure 9-125 Insert logic in the Details window of the snippet: Log material does not exist

21. Add a Invoke node after the snippet and name it Create Material in ERP.

22. Insert the content shown in Figure 9-126 in the Details window of the invoke node.

	Name	Type	Variable
Input(s)	material	PartBG	ActualPart
Output(s)	result	ResultBG	MaterialCreationResult

Figure 9-126 Invoke node Create Material in ERP: Details

23. Add a fault handler for the ApplicationFault to the invoke node Create Material in ERP.

24. Add the following nodes to the ApplicationFault case:

- a. Add a snippet node and name it Log Application Exception during Material creation.
- b. Insert the following logic (Figure 9-127) in the Details window of the snippet.

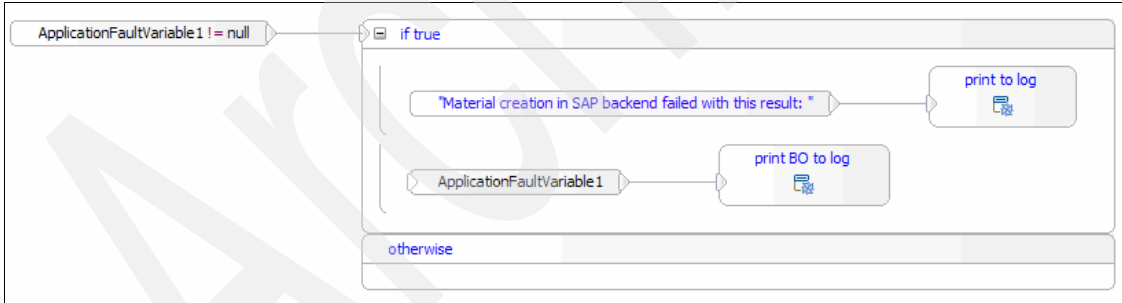


Figure 9-127 Log Application Exception during Material creation snippet: Details

- c. Add a Assign node with the name Prepare Failure response ApplicationFault.

- d. Insert the following assign rules (Figure 9-128) in the Details window of the Assign node.

Assign - prepare Failure response ApplicationFault			
Assign From		Assign To	
E		Oresult Result/Type	
ApplicationFaultVariable2 ApplicationFault FaultName	x/y	Oresult Result/ErrorMessage/ERPPrimary	
ApplicationFaultVariable2 ApplicationFault MessageText	x/y	Oresult Result/ErrorMessage/ERPSecondary	

Figure 9-128 Assign rules in the Details window of the Assign node: Prepare Failure response ApplicationFault

The fault handler section is done.

- e. Add a snippet node and name it dump Result Object.  
f. Insert the following logic (Figure 9-129) in the Details window of the snippet.

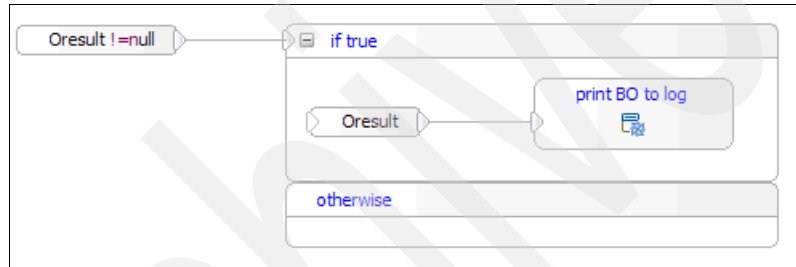


Figure 9-129 Insert logic in the Details window of the snippet: Dump Result Object

- g. Add a Reply node after the snippet and set the properties as shown in Figure 9-130.

Reply - Reply3			
Partner:*	PLMERPInterface		Browse...
Interface:*	PLMERPInterface		
Operation:*	submitECR2ERP		
<input checked="" type="checkbox"/> Use data type variables mapping			
	Name	Type	Variable
Output(s)	result	ResultBG	Oresult

Figure 9-130 Add a Reply node after the snippet: Properties

- h. Place a Terminate node after the Reply node.

The fault handler section is done.

1. Add a snippet node after the invoke node name it Log Material creation result.
2. Insert the following logic (Figure 9-131) in the Details window of the snippet.

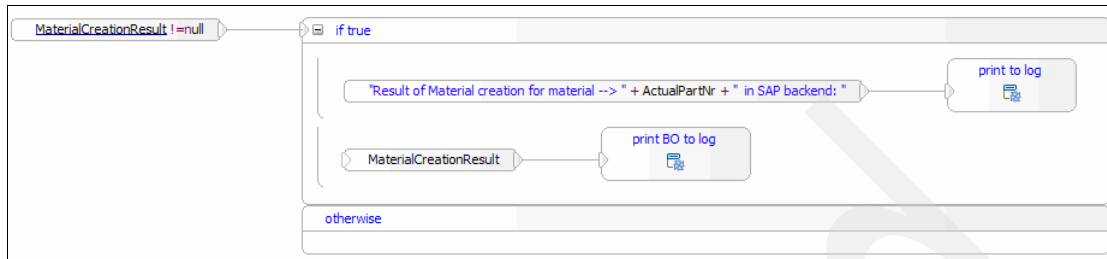


Figure 9-131 Insert logic in the Details window of the snippet: Log Material creation result

3. Add a Choice node and name it Material creation result.
4. Add a new case to the choice with the name Success.
5. Insert the following expression (Figure 9-132) for this new case in the Details window.

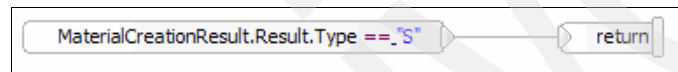


Figure 9-132 Insert expression in the Details window: Material creation result choice node: Success case

6. Add an Otherwise case to the choice node,
7. Add a snippet node to the Success case and name it Attach material creation success to Result.
8. Insert the following logic (Figure 9-133) in the Details window of the snippet.

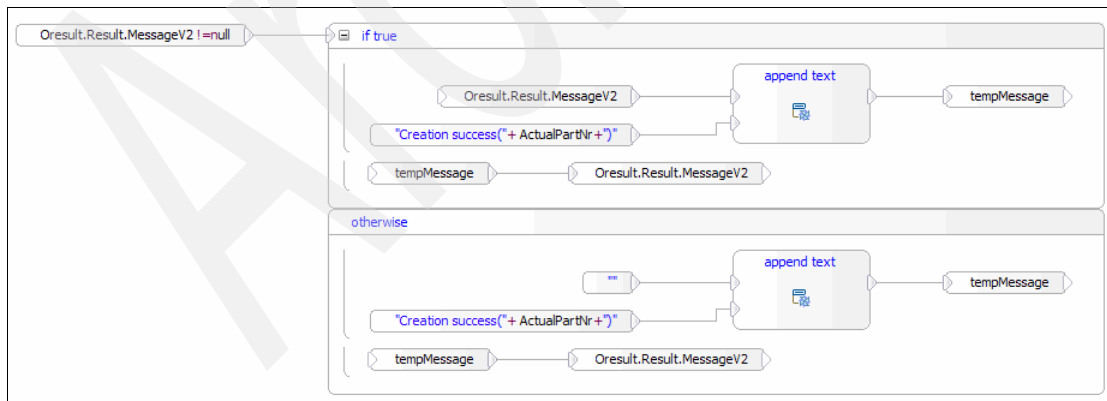


Figure 9-133 Insert logic in the Details window of the snippet: Attach material creation success to Result

This is the end of the lucky path of this forEach section.

9. Add a snippet node to the Otherwise case and name it Attach material creation failure to Result.
10. Insert the following logic (Figure 9-134) in the Details window of the snippet.

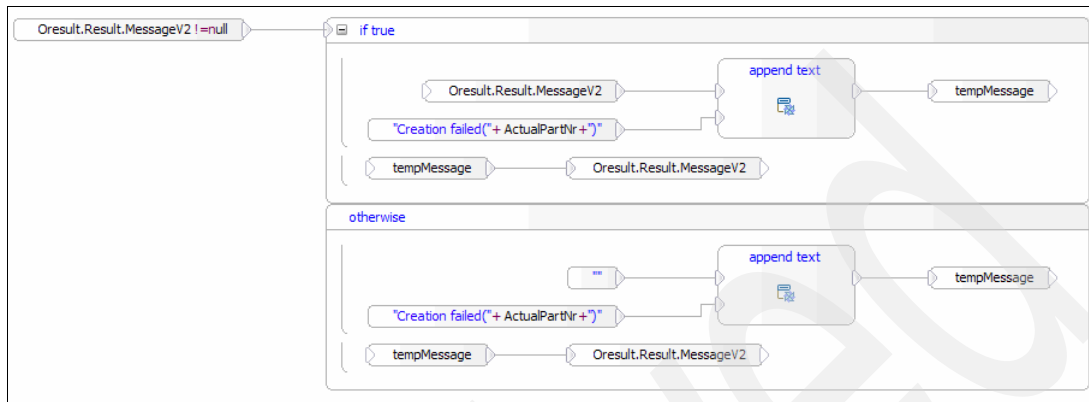


Figure 9-134 Insert logic in the Details window of the snippet: Attach material creation failure to Result

11. Add an Assign node with the name prepare Result Object.
12. Insert the following assign rules (Figure 9-135) in the Details window of the Assign node.

Assign - Assign	
Assign From	Assign To
E	Oresult.Result.Type
MaterialCreationResult.Result/Message	Oresult.Result.ErrorMessage/ERPPrimary
MaterialCreationResult.Result/MessageV1	Oresult.Result.ErrorMessage/ERPSecondary

Figure 9-135 Insert assign rules in the Details window of the Assign node: prepare Result Object

13. Add a snippet node and name it Dump Result Object.
14. Insert the following logic (Figure 9-136) in the Details window of the snippet.

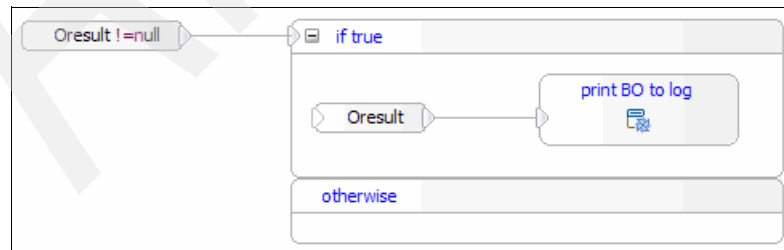


Figure 9-136 Insert logic in the Details window of the snippet: Dump Result Object



15. Add a Reply node after the snippet and set the properties as shown in Figure 9-137.

Name	Type	Variable
result	ResultBG	Oresult

Figure 9-137 Add a Reply node after the snippet

16. Place a Terminate node after the Reply node.

This is the end of the faulty path of this forEach section.

The Process BOMs forEach node and the Process Documents forEach node are built the same way. First a function is called to check if the object already exists. If this function returns a positive response, the change case is approached. Otherwise, it is assumed that the object can be created in the SAP system. When the creation fails, the process terminates and returns the reason for the exception. To see the complete BPEL process, import the TransferECRtoERPSytem.bpel file in the WebSphere Integration Developer and browse through the various elements.

Figure 9-138 shows how the `forEach` section for the Process BOM logic must appear.

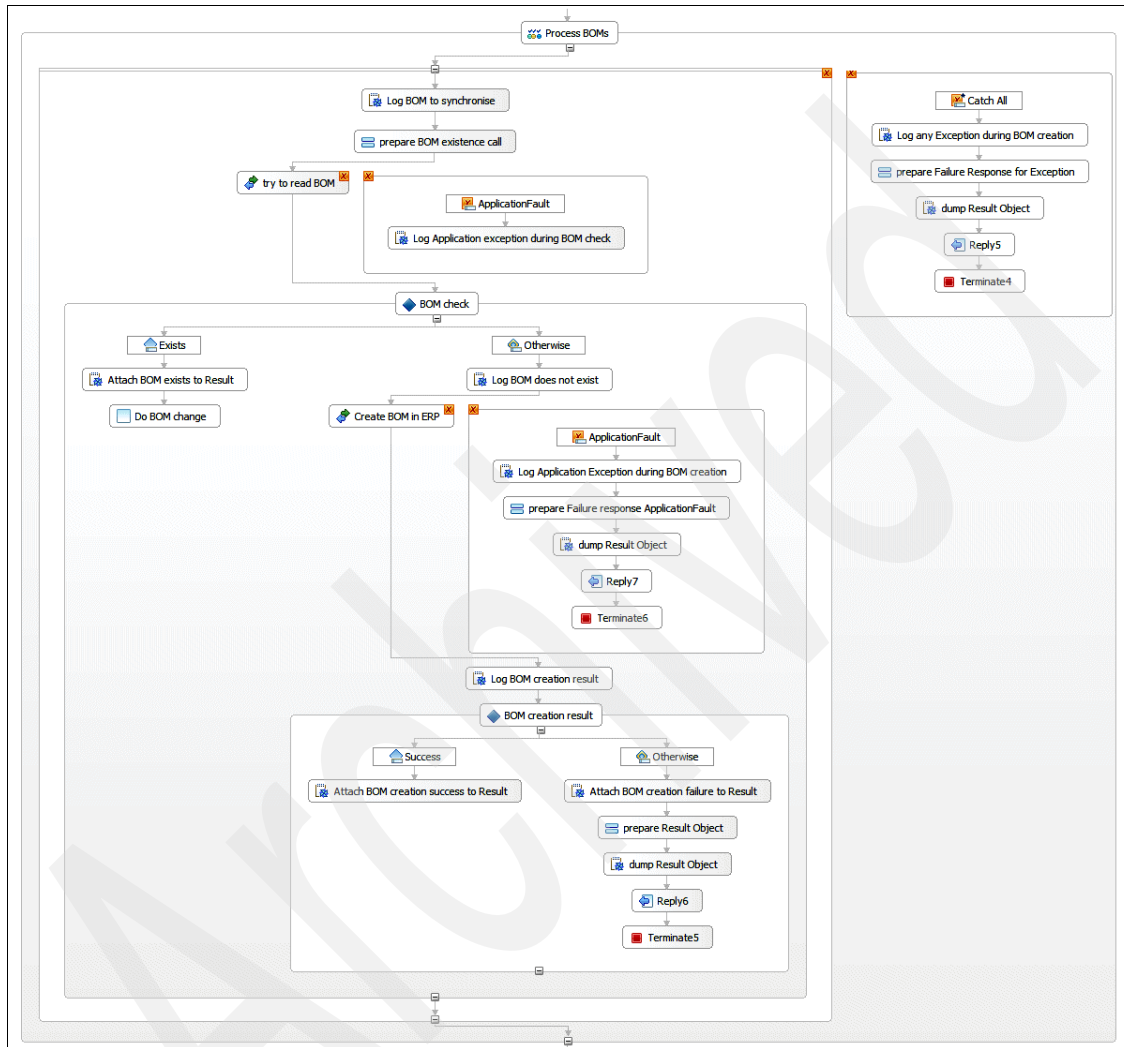


Figure 9-138 `forEach` node Process BOMs

Figure 9-139 shows how the `forEach` section for the Process Documents logic must appear.

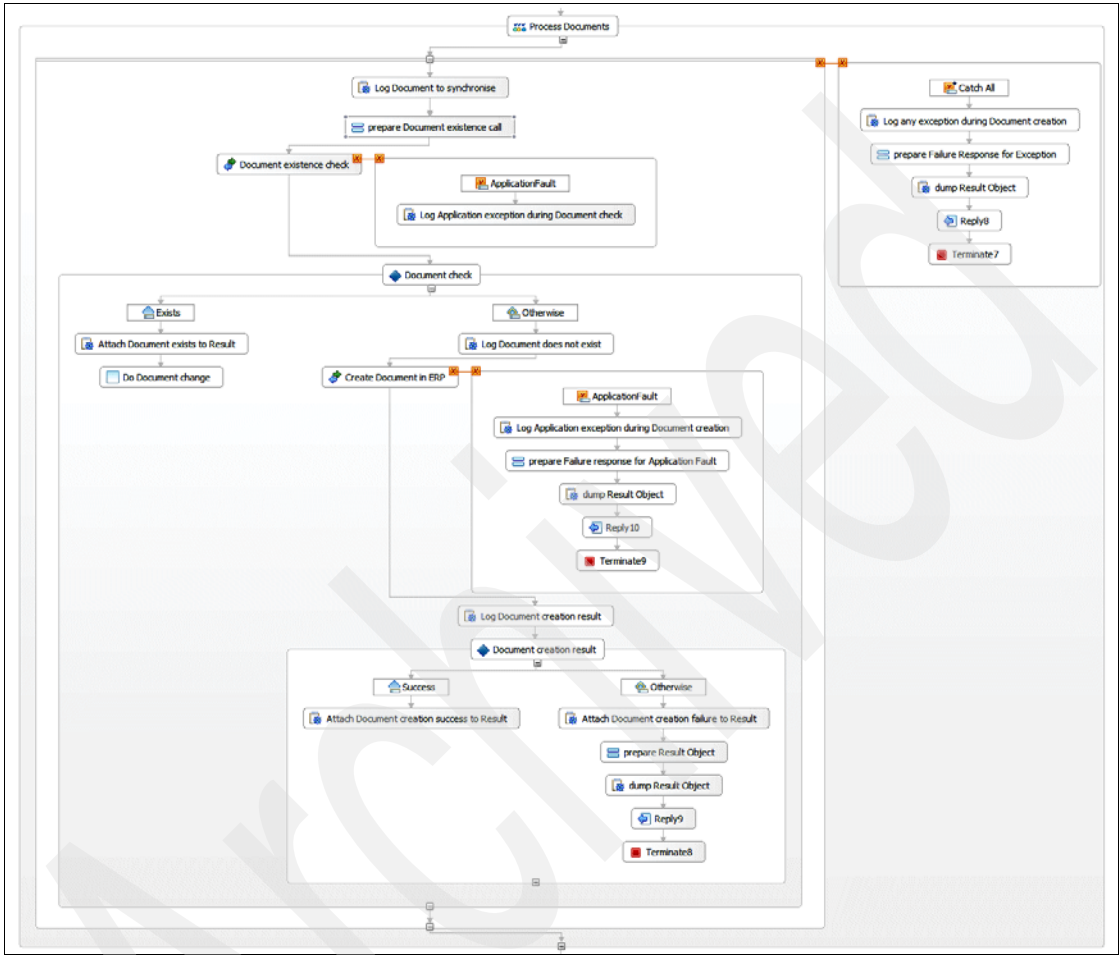


Figure 9-139 `forEach` node Process Documents

## 9.4 Running the scenario

In this section, we run the engineering change scenario.

### 9.4.1 Configuring user and group

This section describes the steps to create the CCB group, and users who belong to the group.

1. Launch the WebSphere Process Server administrative console.
  - a. Start the WebSphere Process Server if it is not already started.
  - b. Right-click **WebSphere Process Server V6.1**, and select **Run administrative console** from its context menu.
  - c. When the Security Alert dialog box opens, click **Yes** to proceed. At the welcome window, enter the following values:
    - User ID: admin
    - Password: admin
2. Create a new managed group, CCB.
  - a. In the administrative console, expand **Users and Groups**.
  - b. Select **Manage Groups**.
  - c. Click **Create...**, and enter the following values:
    - Group name: CCB
    - Description: Change Control Board
  - d. Click **Create**. The successful group creation message must appear.
  - e. Click **Close**. The newly created group, CCB appears in the group list as shown in Figure 9-140 on page 391.

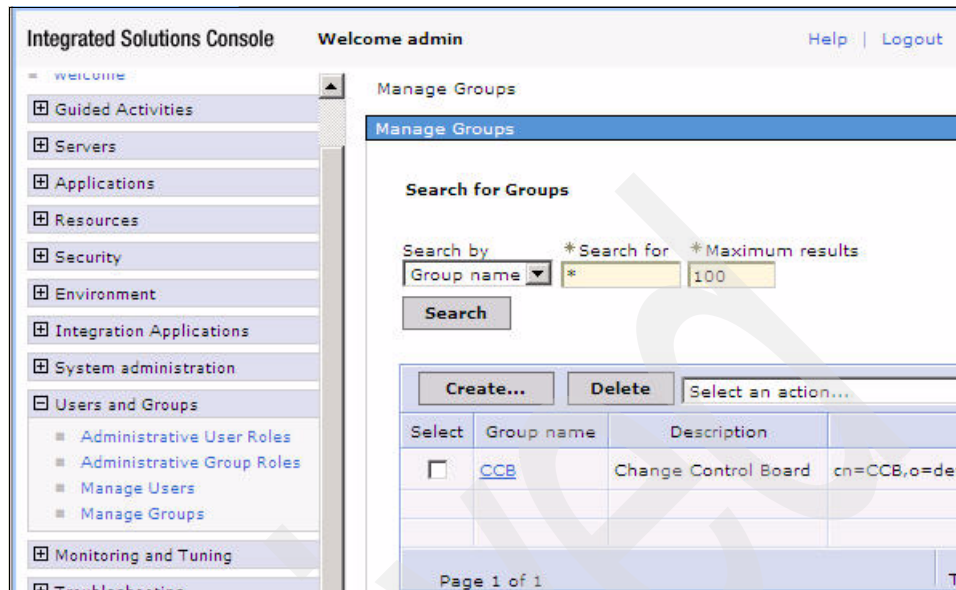
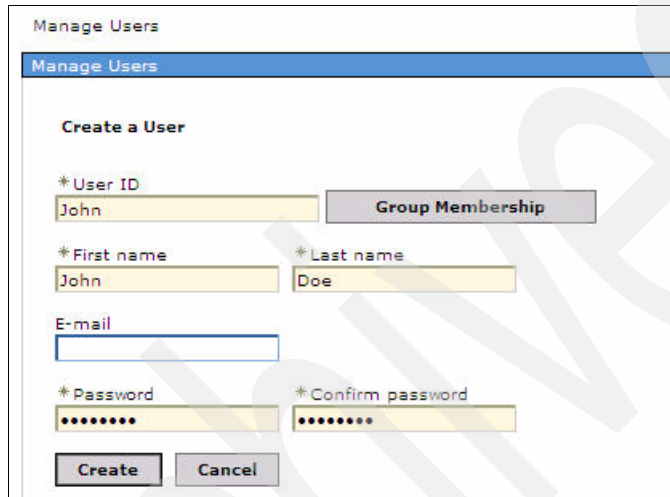


Figure 9-140 Create a new group

3. Create a new user belongs to the CCB group.
  - a. Go to **Users and Groups** → **Manage Users**.
  - b. Click **Create...**, and enter user information. Figure 9-141 shows an example.

**Note:** In this example, a new user of User ID John has been created with password set to pa\$\$w0rd.



The screenshot shows a 'Manage Users' dialog box with a 'Create a User' section. The form contains the following fields and buttons:

- \* User ID:** A text box containing 'John'.
- Group Membership:** A button located to the right of the User ID field.
- \* First name:** A text box containing 'John'.
- \* Last name:** A text box containing 'Doe'.
- E-mail:** An empty text box.
- \* Password:** A text box filled with dots.
- \* Confirm password:** A text box filled with dots.
- Create:** A button at the bottom left.
- Cancel:** A button at the bottom right.

Figure 9-141 Create a new user

- c. Click **Group Membership**.
    - d. Click **Search** to get the list of existing groups.

e. Select **CCB** from the list, then click **< Add** as shown in Figure 9-142.

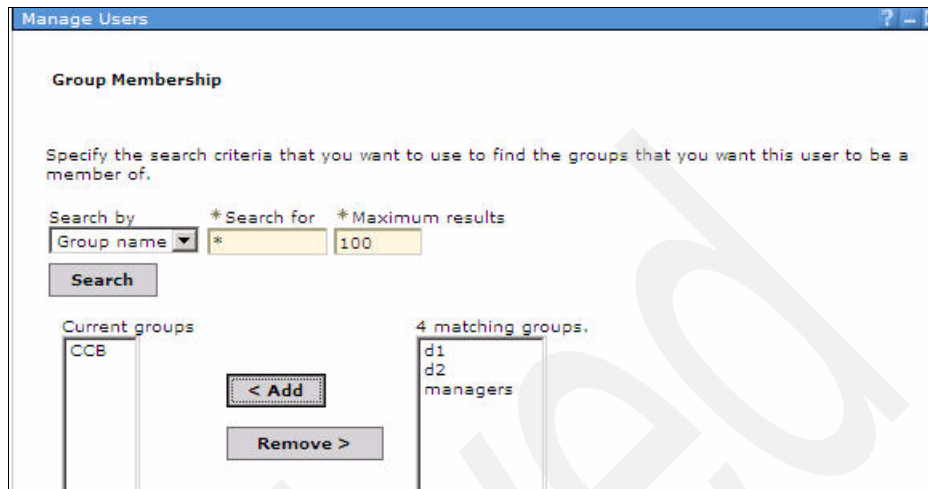


Figure 9-142 Set the group membership

- f. Click **Close**.
  - g. Click **Create** to complete the user creation.
  - h. Click **Close** when the successful user creation message appears. You may repeat the steps to create multiple users that belong to CCB group.
4. Verify the members of the CCB group.
- a. Go to **Users and Groups** → **Manage Groups**.
  - b. Select **CCB** from the list.

**Note:** If no group appears, click **Search**.

- c. Select Members tab. All the users you have created in step 3 on page 392 are listed as shown in Figure 9-143.

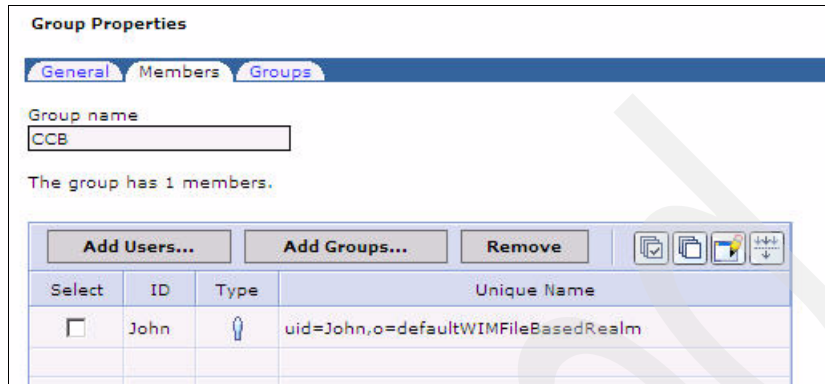


Figure 9-143 Group members for group CCB

5. Click **Logout** to exit the administrative console.

## 9.4.2 Running the scenario

In this section, we run the scenario without involving an SAP system.

1. Deploy all applications to the server. Verify that the TCP/IP Monitor is started.
2. Start the ECR Web client tool to create a new ECR.
  - a. Start a Web browser and enter the following URL:  
`https://localhost:9443/ECRWebClient/Setting.jsp`  
When the browser display the security warning, select **Continue to this website (not recommended)**.
  - b. Enter some test ECR data, then click **Create initial ECR**. Figure 9-144 on page 395 shows an example of a new ECR.



**IBM ECR Client**

**Actions**

- Create initial ECR
- Query ECR status

DUNS number: 1001

Interaction Id: ABC101

Requestor: Sammy Griffin

Affected part: Rad

Problem description: Defect was found during the test drive.

Create initial ECR

Figure 9-144 Create a new ECR request

This takes you to Query ECR status page. If you click **Query ECR status**, the **Messages** must be set to Respond\_initial\_ECR.

**Note:** The DUNS number must be an unique string value. If you wish to create multiple ECRs, be sure to enter a unique DUNS number each time. Remember that the requestor is not a CCB member.

In this example, a problem with wheel (Rad) is reported.

3. An initial ECR must be reviewed by a CCB member to verify its relevance. For this testing, launch Business Process Choreographer Explorer, and log in as a CCB member to review the submitted ECR.
  - a. In the Servers view, right-click **WebSphere Process Server V6.1**, and select **Launch** → **Business Process Choreographer Explorer**.
  - b. At the welcome window, enter a user ID and password that you created earlier.

**Note:** In this example, we used the following values:

- user ID: John
- password, pa\$\$w0rd

- c. There must be a to-do item ready to be worked on as shown in Figure 9-145.

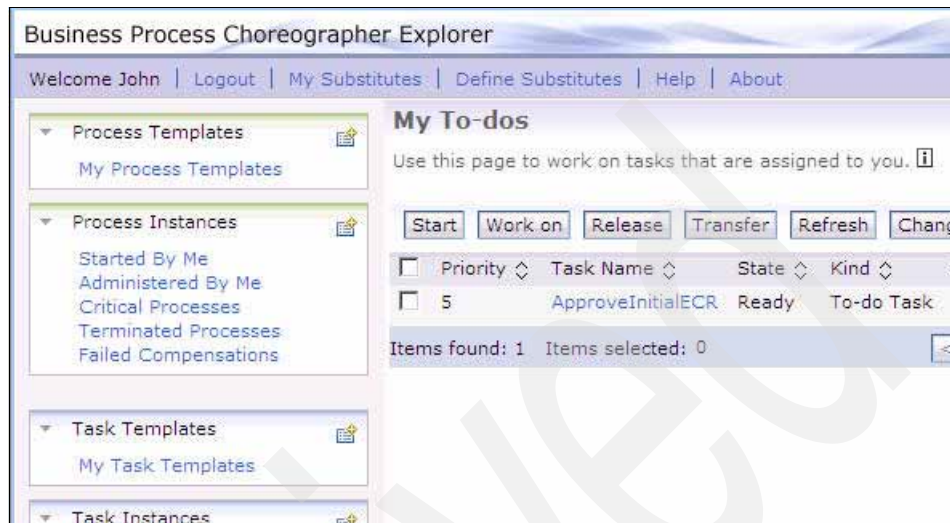


Figure 9-145 Business Process Choreographer Explorer

- d. Select the check box for the to-do item and click **Work on**. This displays the detail of the ECR.
- e. Scroll to the Task Output Message section. You may reject this initial ECR by changing the approval value to false. In this example, however, we are going to approve this initial ECR to see the entire event flow. Enter some value in the comment field if you wish, then click **Complete** to finish the ECR submission.

**Note:** After the work item has been completed, you may return to the ECR Client Web tool to query the current ECR status that the ECR requester, Sammy Griffin has submitted. The Messages type must be set to Notify\_initial\_ECR\_accepted. Remember that the message contents were set by the addNotifyInitialECRAcceptedMsg invoke activity.

4. In the BPC Explorer, go to **Task Instances** → **My To-dos**. A new to-do is ready to be worked on.

At this point, CCB performs risk assessment before deciding to raise the ECR. In our example, an ECR was submitted on wheel (Rad), which affects rim (Felge) and tire (Reifen). CCB must evaluate the affect on dependant parts in order to make the final decision.

Repeat steps d and e of step 3 on page 395 to approve this ECR. This completes the ECR and a BOM update is triggered. By default, it is assumed that no SAP system is present and the data is dumped out to the console.

**Note:** In this simplified scenario, the engineering change (modification of the affected parts) and specification of change effectivities, and so forth, are not implemented.

Review the output displayed in your console view. You see simulated data as shown in Figure 9-146.

```

WebSphere Process Server v6.1 [WebSphere ESB Server] WebSphere Process Server v6.1 (WebSphere v6.1)
[3/19/08 20:33:12:156 EDT] 00000005b SystemOut O handleSyncRequest: query_messages:ECRClient:test
[3/19/08 20:41:30:656 EDT] 00000120 WSChannelFram A CHF00019I: The Transport Channel Service has star
[3/19/08 20:41:35:484 EDT] 000000f8 SystemOut O -----
[3/19/08 20:41:35:484 EDT] 000000f8 SystemOut O Simulating SAP: The following data was received:
[3/19/08 20:41:35:484 EDT] 000000f8 SystemOut O multiParts:
[3/19/08 20:41:35:484 EDT] 000000f8 SystemOut O <?xml version="1.0" encoding="UTF-8"?>
<p:MultiParts xsi:type="p:MultiParts"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://ITSOPMLLibrary/bo">
  <Parts>
    <UID>id_10019_793</UID>
    <PartNumber>A4000102101</PartNumber>
    <Description>Rad</Description>
    <ChangeNumber>1001</ChangeNumber>
  </Parts>
  <Parts>
    <UID>id_10019_785</UID>
    <PartNumber>A5000100011</PartNumber>
    <Description>Felge</Description>
    <ChangeNumber>1001</ChangeNumber>
  </Parts>
  <Parts>
    <UID>id_10019_786</UID>
    <PartNumber>A5000100010</PartNumber>
  </Parts>

```

Figure 9-146 Simulated SAP test result

### 9.4.3 Running the scenario with attached SAP system

If you have access to an appropriate SAP system, the following steps can be performed to propagate the BOM update to the real SAP system.

1. Start the PLM Web Admin client tool to deactivate the simulation mode.
  - a. Start a Web browser and enter the following URL:  
<https://localhost:9443/PLMS2AdminWeb>  
 When the browser display the security warning, select **Continue to this website (not recommended)**.
  - b. Login with a UserID that has administrative right, for example, admin.

- c. Select the write\_message ECRClient radio box (Figure 9-147) and click **Edit subscription**.

The screenshot shows the IBM Subscription Manager interface. It features a table with subscription details. The first row is selected, indicated by a radio button and a red box. The details for this subscription are as follows:

type	source	user	description
write_messages	ECRClient	*	

Below the table, there are several buttons: 'Edit subscription', 'Remove subscription', 'Add subscription', 'Reload configuration', and 'Update configuration'. The 'Edit subscription' button is circled in red.

Figure 9-147 Edit subscription for ECRClient: write\_message

- d. Select the parameter **Simulate SAP** (Figure 9-148), set the value to false, and click **Update**.

IBM Subscription Manager

Key: write\_messages Source: ECRClient User: \*

Description:

Enabled: ☒

Action: PLMWriteMessagesIF Url: sca://TruckInc/InitializeECRExport

Action description:

Action user: useDefaultUser ☐ User: Password:

	Id	Url	Description	useDefaultUser	User
Action endpoints	pdmsystem1	http://localhost:9090/PLM		<input type="checkbox"/>	
	pdmsystem2	http://localhost:9090/PLM		<input type="checkbox"/>	
	query	sca://TruckInc/Federate		<input type="checkbox"/>	

Add endpoint

	Key	Value
Action properties	documentUrlPrefix	http://localhost:9090/Do
	simulateSAP	false

Add property

Update Cancel

Figure 9-148 Deactivate SAP simulation

- e. Save the configuration by clicking **Update configuration**. It becomes effective immediately.
2. Start the ECR Web client tool to create a new ECR.
    - a. Start a Web browser and enter the following URL:  
https://localhost:9443/ECRWebClient/Setting.jsp
    - b. When the browser display the security warning, select **Continue to this website (not recommended)**.

3. Enter some test ECR data, then click **Create initial ECR**. See Figure 9-149.

IBM ECR Client

Actions

- Create Initial ECR
- Query ECR status

DUNS number: PLMECN1

Interaction Id: ABC1001

Requestor: Sammy Griffin

Affected part: Rad

Problem description: Defect found during test run.

Create initial ECR

Figure 9-149 Create initial ECR request using ECR client

**Note:** The subsequent steps of approving the ECR request are exactly the same as described in the previous section.

After the second approval, the request is not simulated but sent to the BPEL process TransferECRtoERPSysm. The process checks some values and calls the SAP system business functions according the process design.

The resulting Log file looks similar to Figure 9-150.

```
[5/7/08 4:31:39:531 CEST] 0000003e SystemOut O <?xml version="1.0" encoding="UTF-8"?>
<p:ResultBG xsi:type="p:ResultBG"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://ITSO-SAI707/PLMLibrary">
  <Result>
    <Type>S</Type>
    <Message>ECR process successfully executed.</Message>
    <MessageV1>SAP ECN created.</MessageV1>
    <MessageV2>Creation success(A4000102101)Creation success(A5000100011)Creation success(A5000100010)</MessageV2>
    <MessageV3>BOM creation success(A4000102101)BOM creation success(A4000102101)BOM creation success(A4000102101)</MessageV3>
    <MessageV4>Document creation success(id_10009_507)Document creation success(id_10009_517)Document creation success(id_10009_537)</MessageV4>
    <ChangeNumber>PLMECN1</ChangeNumber>
  </Result>
</p:ResultBG>
```

Figure 9-150 Log entries of successful SAP access

The automatic process created a Engineering Change Number, three materials, one BOM, and three documents in the SAP systems.

The result in the SAP system can be checked by using the following SAP transaction codes:

- ▶ CC03: Display Engineering Change Number
- ▶ MM03: Display Materials
- ▶ CS03: Display Bill of Materials
- ▶ CV03N: Display Documents

For example, the Engineering Change Number can be reviewed using transaction code CC03. See Figure 9-150 on page 400.

The screenshot displays the SAP 'Display Change Master: Change Header' interface. The top menu bar includes 'Change Master', 'Edit', 'Goto', 'Extras', 'Settings', 'Environment', and 'System'. The main title is 'Display Change Master: Change Header'. Below the title, there are tabs for 'Object Types', 'Objects', and 'Alternative Dates'. The 'Change number' field contains 'PLMECN1'. The 'Defect found during test drive.' is entered in the 'Reason for Change' field. The 'Valid From' date is '07.05.2008'. The 'Status Information' section shows 'Change no. status' as '1 Active'. The 'Administrative Data' section shows 'Created On' as '06.05.2008' and 'Created by' as 'KTW'. The bottom status bar indicates 'BSC (2) (800)' and 'wccpc11 INS'.

Description	
Valid From	07.05.2008
Authorization group	
Reason for Change	Defect found during test drive.

Status Information	
Change no. status	1 Active
<input type="checkbox"/> Usage	
<input type="checkbox"/> Deletion Indicator	

Administrative Data	
Created On	06.05.2008
Created by	KTW
Changed On	
Changed by	

Figure 9-151 SAP ECN created by process

Archived





## **SOA realization of integrated PLM solutions: Working with examples**

This chapter discusses the use of WebSphere Service Registry and Repository in a service-oriented architecture (SOA).

## 10.1 Dynamic service invocation (integration with WSRR)

This section discusses the role and implementation of WSRR and WESB. In addition, this section provides instructions on integrating the WSRR in the dynamic routing process in an SOA-based OMG PLM Services 2.0-compliant solution.

### 10.1.1 WSRR role

This section explains the role of WSRR in an ESB environment, using WESB as the ESB implementation.

A graphic presentation of this integration is shown in Figure 10-1.

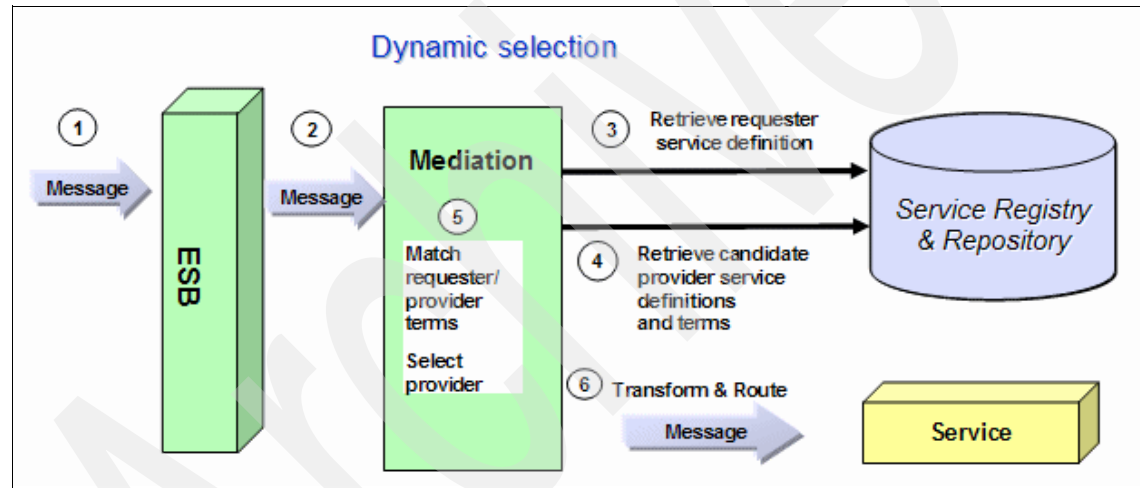


Figure 10-1 WESB—WSRR integration

The advantage of using WSRR in a SOA-based solution is described in the WSRR Handbook *WebSphere Service Registry and Repository Handbook*, SG24-7386:

“The use of WSRR allows ESB mediations that interact with some set of endpoint services to be more tolerant of changes within the environment. This is enabled by the ESB using WSRR as a dynamic lookup mechanism, providing information about service endpoints (the service metadata). So, when changes occur in the service endpoint environment, the associated ESB mediations do not have to change.”

## 10.1.2 WSRR and PLM Services 2.0

This section details WESB v6.1 features that enable dynamic service invocation by demonstrating how to create a mediation module to locate services from WSRR and dynamically invoke appropriate endpoints. More importantly, it demonstrates how to integrate the WSRR in the dynamic routing process in an SOA-based OMG PLM Services 2.0-compliant solution.

The following topics are discussed:

- ▶ A typical business scenario requirement
- ▶ Overview of a sample technical implementation
- ▶ Leveraging WSRR
- ▶ Laboratory: Demonstrating the technical solution

## 10.1.3 A typical business scenario requirement

Trucks Inc, our fictitious company (see Chapter 6, “Achieving business flexibility with a service-oriented architecture” on page 121 about Trucks Inc), has one enterprise PDM system. Because of cost restraints, it has decided it will not pay for the maintenance and support of two PDMs (a typical PDM implementation usually costs USD\$10–15m and on-going support costs per PDM usually runs up to USD\$2–3M per year).

We take the business scenario further and assume that Trucks Inc decides to keep the separate mechanical design (M-CAD) engineering divisions that it inherited out of the merge. That is, Trucks Inc now has an M-CAD that is based in Boeblingen and handles Cabin designing, and another M-CAD division that is located in La Guede that handles Chassis designing. In the course of the ECR progression, the ECR needs to be routed to the appropriate truck divisions so that the respective engineers can perform the required redesign tasks.

Now we define two sample use cases based on the PLM Services 2.0 ECM use cases derived from the ECM VDA standard.

## Use Case 1: Dynamic business process selection

This demonstrates the selection of BPEL triggered by a request that is initiated by the ECM\_coordinator (Original Equipment Manufacturer) and is going to the ECM\_participant (Supplier).

The routing criteria is content-based, using the value of the Approval\_status/Status\_name element of the PLM Services 2.0 container, as illustrated in the sample XML shown in Figure 10-2. The XMLs have been collapsed and expanded on the elements we need for our use cases.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <PLM_container uid="Redbook_Use_Case_1" version_id="2.0"
  xmlns="urn:omg.org/plm20/informational/model"
  xmlns:core="urn:omg.org/plm20/informational/core"
  xmlns:info="urn:omg.org/plm20/schemaInfo"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:omg.org/plm20/informational/model
  http://schema.omg.org/specs/PLM/2.0/schema/InformationalModel.xsd">
+ <Activity uid="id_a1">
+ <Activity uid="id_a2">
+ <Application_context uid="id_ac1">
- <Approval_status uid="id_as1">
  <Status_name>Request_ECR_comments</Status_name>
- <Approval uid="id_app1">
  <Is_applied_to>id_wr1</Is_applied_to>
  </Approval>
</Approval_status>
+ <Date_time uid="id_dt1">
+ <Date_time uid="id_dt2">
+ <Item uid="id_i1">
+ <Organization uid="id_o1">
+ <Person uid="id_p1">
+ <Specific_item_classification uid="id_sic1">
+ <Work_request uid="id_wr1">
</PLM_container>
```

Figure 10-2 Sample PLM Container XML for BPEL selection

### Routing rules

If the Status\_name = Request\_ECR\_comments, then we invoke the ECMParticipantComments BPEL.

If the Status\_name = Notify\_ECR\_creation, then we invoke the ECMParticipantDetailingAndComments BPEL

## Use Case 2: Dynamic service selection

This use case demonstrates the selection of services provided by different departments. The routing is content-based, using the value of two elements of the PLM Services 2.0 container:

- ▶ Application\_context/Application\_domain
- ▶ Organization/Organization\_name.

A sample PLM Container XML is shown in Figure 10-3.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <PLM_container uid="Redbook_Use_Case_2" version_id="2.0"
  xmlns="urn:omg.org/plm20/informational/model"
  xmlns:core="urn:omg.org/plm20/informational/core"
  xmlns:info="urn:omg.org/plm20/schemaInfo"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:omg.org/plm20/informational/model
  http://schema.omg.org/specs/PLM/2.0/schema/InformationalModel.xsd">
- <Application_context uid="id_ac1">
  <Application_domain>Mechanical design</Application_domain>
  <Life_cycle_stage>design</Life_cycle_stage>
</Application_context>
+ <Item uid="id_i1">
- <Organization uid="id_o1">
  <Organization_name>La Gaude</Organization_name>
  <Organization_type>department</Organization_type>
  <Id>o1</Id>
</Organization>
+ <Specific_item_classification uid="id_sic1">
</PLM_container>
```

Figure 10-3 Sample PLM Container XML for WS selection

### Routing rules

If Application\_domain = Mechanical design and Organization\_name = Boeblingen, then we request a service provided by the Boeblingen M-CAD division.

If Application\_domain = Mechanical design and Organization\_name = La Gaude, then we request a service from the La Gaude M-CAD division.

## 10.1.4 Overview of the technical implementation

WebSphere Enterprise Service Bus has capabilities that enable solutions to have dynamic routing mechanisms and dynamic service invocations. Integrating the use of a services registry system like the WSRR system makes the solution more manageable.

A complete discussion of the features and capabilities of WSRR is not in scope for this Redbooks publication. You can view detailed coverage of WSRR in the IBM Information Center. Visit the following Web page:

<http://publib.boulder.ibm.com/infocenter/sr/v6r1/index.jsp>

We discuss the WSRR features that are relevant to our laboratory exercise.

## 10.1.5 Leveraging WSRR

Following the SOA approach, we assume the business tasks of re-designing the cabin and chassis mechanical parts and the underlying information technology are now exposed as reusable components or services.

In our solution example, we use the following WSRR capabilities:

- ▶ WSRR support for the deployment phase of services

WSRR can store the record for metadata describing service interaction endpoints. The population of metadata into WSRR can be done manually by importing your SOA solution documents in WSRR, or automatically, when your SOA solution is deployed and the WSRR service discovery mechanism is configured.

- ▶ WSRR support for managing services

Our proposed solution performs a lookup of WSRR to retrieve the appropriate services to invoke. WSRR allows attaching of metadata like classifications, properties and relationships, to service artifacts. These metadata enhances the capability for determining the correct service to use.

## 10.1.6 Laboratory: Building the sample mediation module

This section shows how to create some of the modules required to show dynamic routing and how to integrate the use of WSRR. This section also describes what prerequisite development environment setup needs to be done.

### Requirement overview

The following diagrams show our target solution artifacts in the context of the use cases we are fulfilling.

Figure 10-4 shows the assembly diagram of our sample mediation module to handle dynamic BPEL selection.

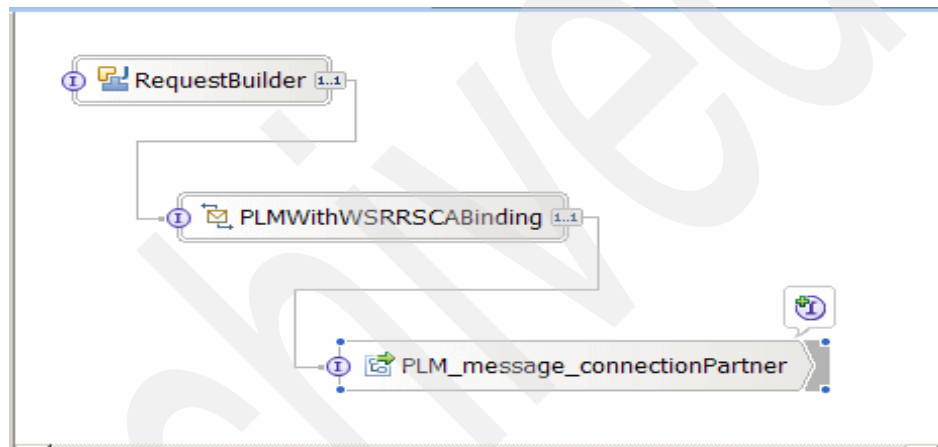


Figure 10-4 Dynamic BPEL selection assembly diagram

Figure 10-5 shows the assembly diagram of our sample mediation module to handle dynamic service selection.

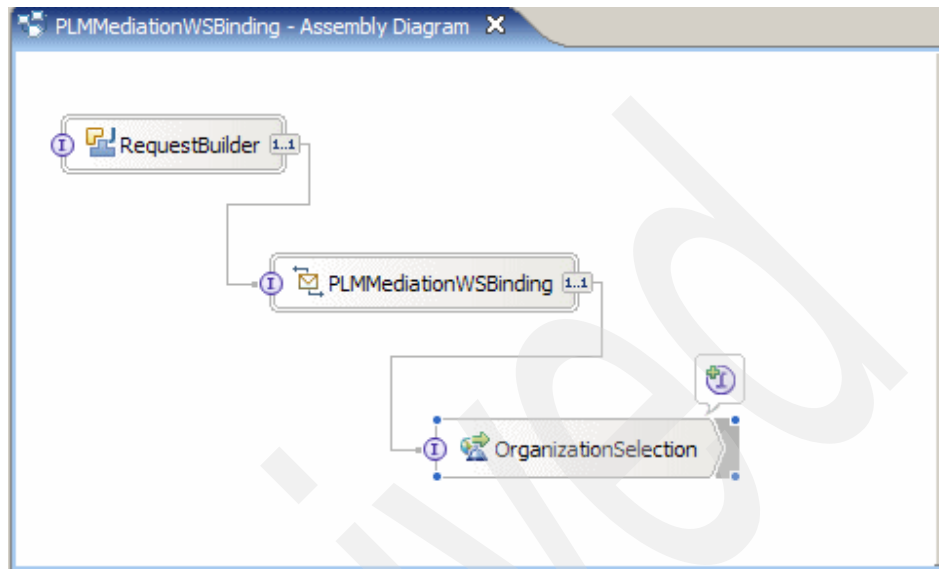


Figure 10-5 Dynamic service selection assembly diagram



Figure 10-6 shows the assembly diagram of the process module with the business processes.

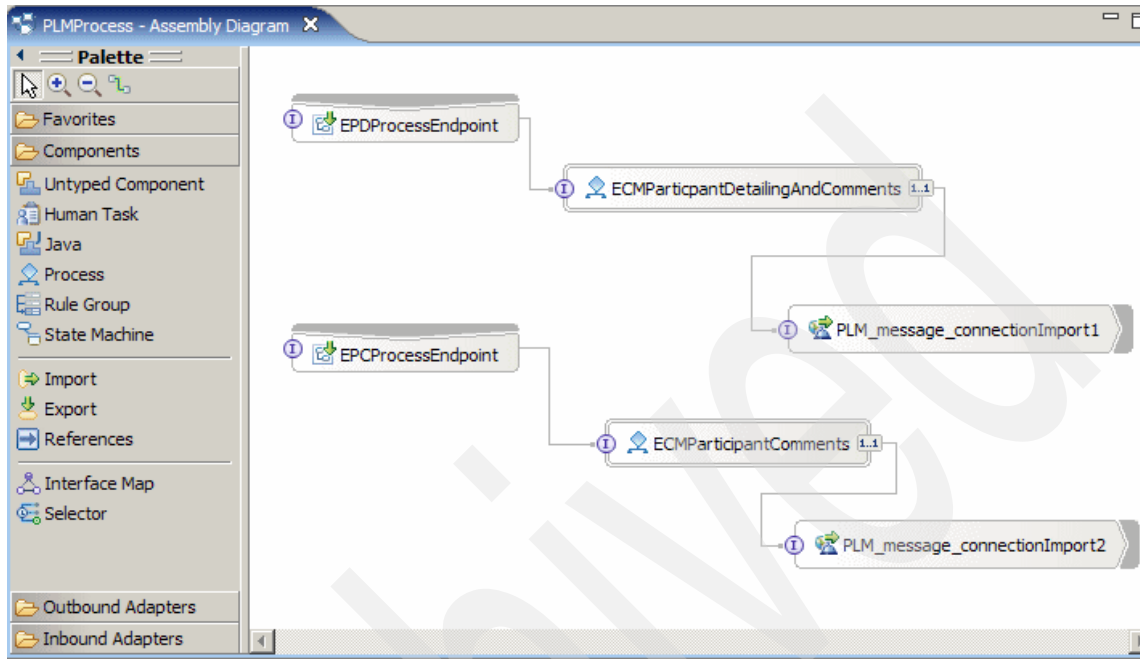


Figure 10-6 Process module assembly diagram

### Development environment setup

You need the following software components on your PC:

- ▶ WebSphere Integration Developer V6.1 (WID)
- ▶ WebSphere Process Server V6.1 (WPS)
- ▶ WebSphere Service Registry and Repository V6.1 (WSRR)

### WID and WPS

Refer to Section 7.1.1, “Installing WebSphere Integration Developer V6.1” on page 180.

By default, security is enabled on the WPS. For this exercise, we do not use a secure environment. Perform the following steps to disable security:

1. Go to the Servers tab and ensure your WPS is started.
2. Right-click **WPS server** → **Run administrative console**.
3. Log in using the installation default administrative user/password, admin/admin.



**Important:** For the installation directory, choose a directory path where the directories do not have spaces in their names. Otherwise, some WSRR jacbs may not run properly when using the WSRR deployment wizard.

After the installation, a set of files and folders are created in your installation directory as shown in Figure 10-8.

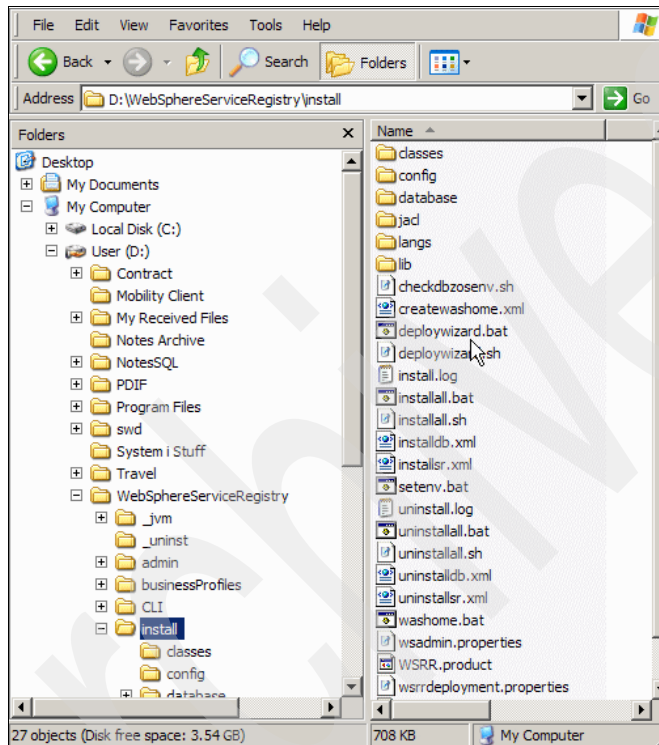


Figure 10-8 WSRR installation directory

### WSRR deployment

You now need to deploy WSRR into the WPS V6.1 test environment inside your WID V6.1.

1. Deploy WSRR using the deployment wizard from the WSRR installation root:  
`<WSRR install root>/install/deploywizard.bat`
2. Fill in the prompts in the deployment wizard panels following the detailed instructions in the guide found at the following Web page:

[http://publib.boulder.ibm.com/infocenter/sr/v6r1/index.jsp?topic=/com.ibm.sr.doc/twsr\\_installn02\\_standalone.html](http://publib.boulder.ibm.com/infocenter/sr/v6r1/index.jsp?topic=/com.ibm.sr.doc/twsr_installn02_standalone.html)

**Tips:**

- ▶ Use local database.
- ▶ For WebSphere Application Server installation directory:
  - use the <install root> of your WPS test environment. For instance C:\IBM\WID61\runtimes\bi\_v61
  - set the Profile Name to wps
- ▶ For the SOAP Port, ensure it matches the SOAP\_Connector\_Address of your WPS. You can check this from the WPS Administration Console

**Servers → Application Servers → <your WPS server name> → Ports → SOAP\_Connector\_Address**

Set the Server Name to your WPS server name

3. Import the Governance profile if it has not been imported. Use the GovernanceProfile.zip file from the WSRR installation root  
`<install_root>\businessProfiles\GovernanceProfile\configurationProfile`

## WSRR configuration

The setup required for WSRR involves some WESB/WPS configuration as well as introducing a customized WSRR classification system.

### WESB/WPS configuration

We need a mechanism to connect to WSRR and look up a service. This mechanism is called a *WSRR definition*. This access definition can be created and managed using the WESB/WPS administration console.

Create your WSRR definition using the detailed instructions in the IBM Redbooks publication, *WebSphere Service Registry and Repository Handbook*, SG24-7386.

**Attention:** For this chapter exercise, we are not using security. You do not need to configure Authentication alias and SSL Configuration in the WSRR Connection properties panel. To create a WSRR definition with security, refer to the following Web page:

[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb602.doc/sibxwsrrresources/RegistryWSConnection\\_DetailForm.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb602.doc/sibxwsrrresources/RegistryWSConnection_DetailForm.html)

After creating your WSRR definition, ensure the connection properties panel looks similar to what is shown in Figure 10-9. The Registry URL may have a different port number depending on the value of the WC\_defaulthost of your WPS.

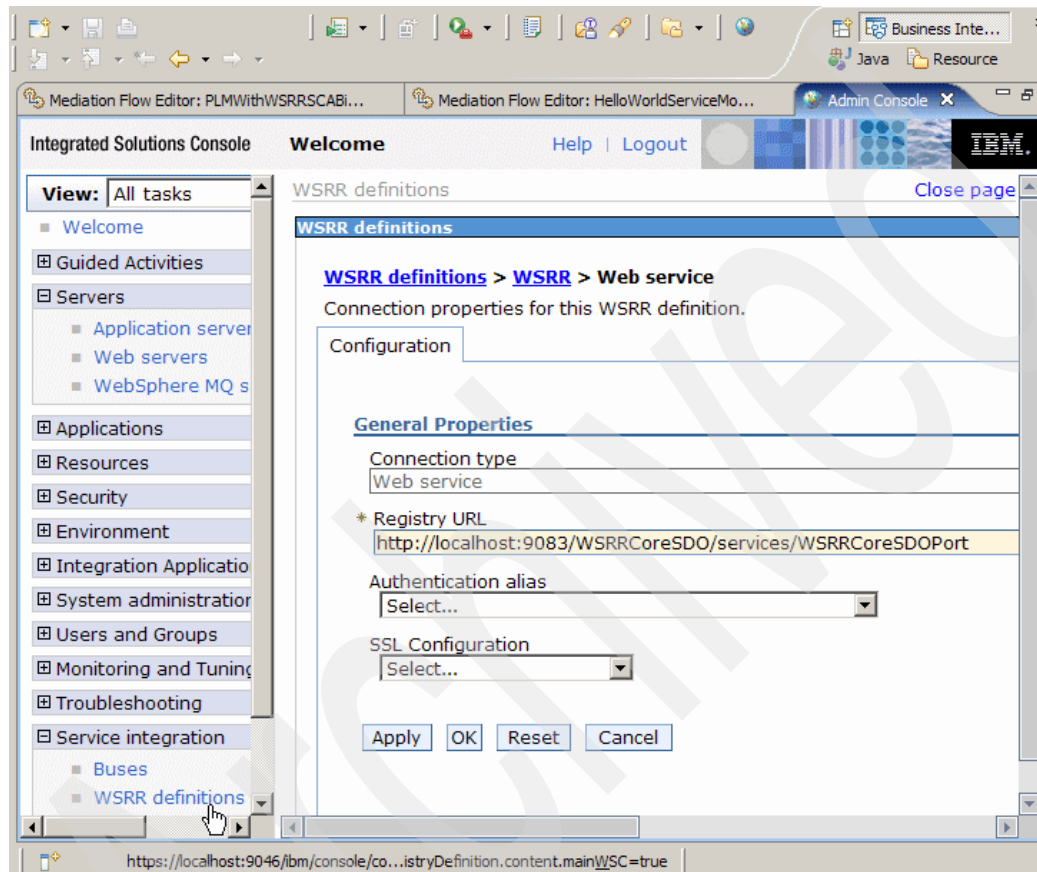


Figure 10-9 WSRR definition connection properties

### **WSRR classifications**

The WSRR support for managing systems provides the use of Classification Systems to organize and categorize services. In WSRR, the classification systems are defined using the Web Ontology Language (OWL).

A good classification system makes it easy to find services. For example, applying classifications to entities in the WSRR enables the user to locate an available service provided by the M-CAD department in City X.

For this lab, we use a sample customized classification system that could represent a typical Automotive Industry Component Business Model (CBM). The taxonomy is based on the ProSTEP iViP documentation Usage Guide for OMG PLM Services 2.0.

**Tip:** A guide for creating a customized WSRR classification system is available from developerWorks® at the following Web page:

[http://www-128.ibm.com/developerworks/websphere/library/techarticles/0703\\_shore/0703\\_shore.html](http://www-128.ibm.com/developerworks/websphere/library/techarticles/0703_shore/0703_shore.html)

### ***Customizing WSRR classifications***

This section details how to introduce a sample company-specific classification system that can be used for looking up registered services in WSRR. Refer to "Configuring the WSRRLookup primitive" List number (f) on page 432 of the section, "Build the business process selection mediation logic flow" List number (9) on page 429 for more information about options to do WSRR look-up.

Follow these steps to load a sample customized classification into your WSRR:

1. Download the PLMWithWSRR.zip file from Appendix A, "Additional material" on page 465 and extract it into a local directory.

Inside the resulting PLMWithWSRR folder, there is a sample customized classification system OWL file called TrucksProfileTaxonomy.owl. Copy this file to your WSRR installation root directory in the folder path:

```
<install_root>\businessProfiles\GovernanceProfile\configurationProfile\OWL_CONTENT\Classifications
```

2. Open a browser to your WSRR admin console.

```
http://localhost:<port>/ServiceRegistry
```

In the above URL, <port> is your WebSphere Process Server WC\_defaulthost.

3. In the Perspective drop down menu, select **Configuration** → **Go**.

4. From the Configuration tasks panel shown Figure 10-10, expand the **Active Configuration Profile**.

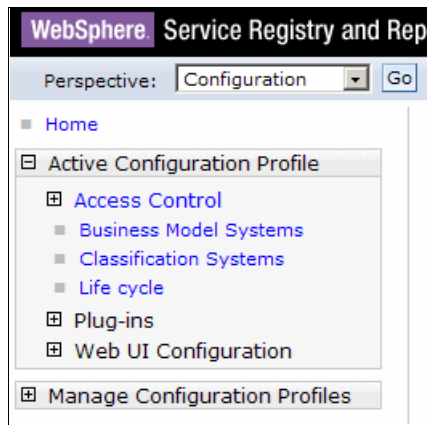


Figure 10-10 Configuration tasks

5. Click **Classification Systems** to open the Load Classification System window shown in Figure 10-11.

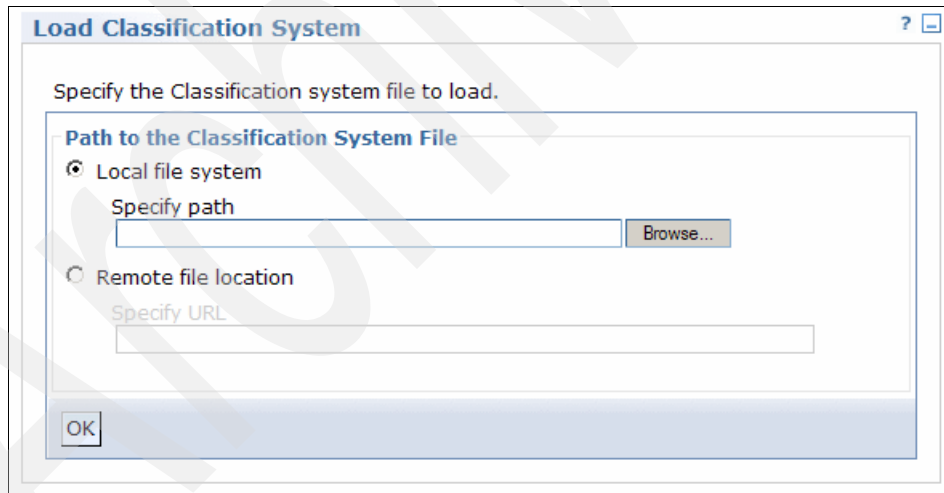


Figure 10-11 Load classification system

6. With the Local file system radio button selected, click **Browse** and navigate to the directory where you copied the sample classification system OWL file
7. Select the TrucksProfileTaxonomy.owl file. Click **OK**.

After loading the sample classification file, the Classification Systems panel now includes the new taxonomy shown in Figure 10-12.

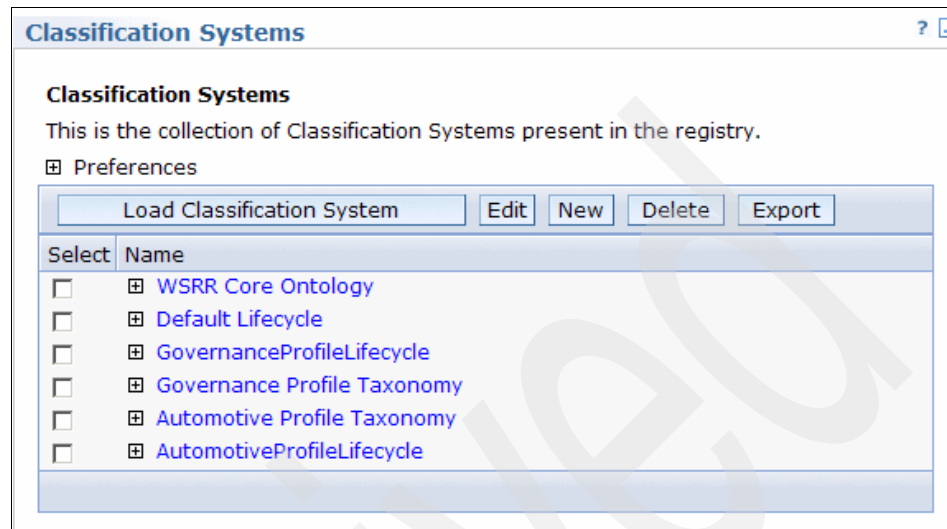


Figure 10-12 Customized Classification System

8. Expand **Automotive Profile Taxonomy** → **Business Domain**.

**Business Domain** contains the sample Application Domains.

a. Expand **Change Management**. This application system contains the sample business processes used for the dynamic BPEL selection use case:

- ECMParticipantComments
- ECMParticipantDetailingAndComments

b. Expand **Mechanical Design**. This application system contains the sample truck design departments used for our dynamic service selection use case:

- Boeblingen—Cabin MCAD
- La Gaude—Chassis MCAD

The expanded Classification Systems panel is shown in Figure 10-13 on page 419.



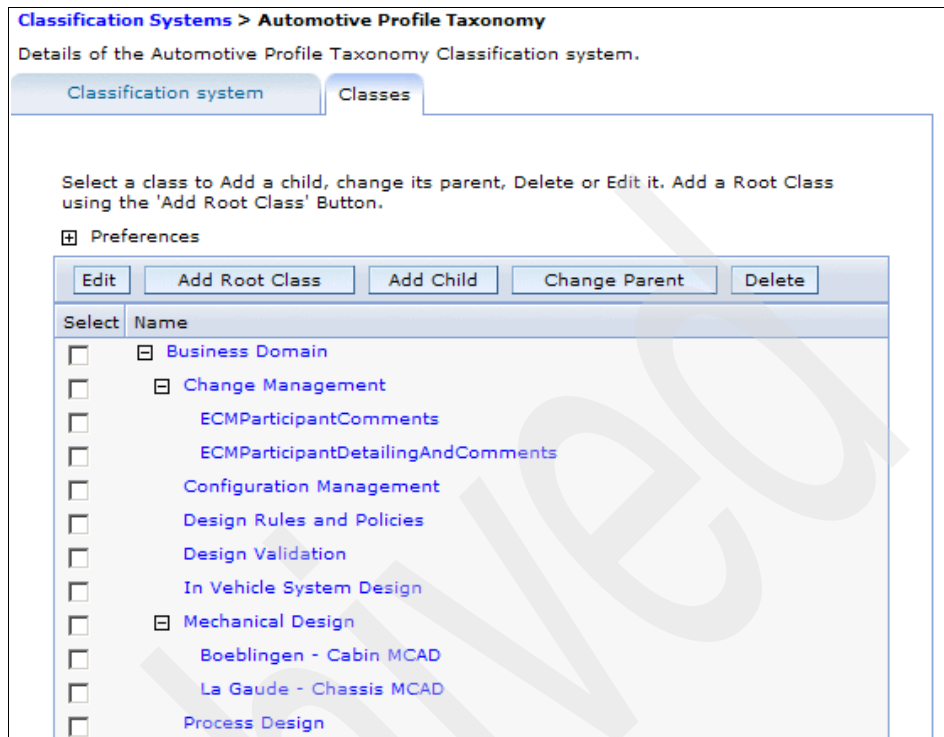


Figure 10-13 Sample customized classification

## Hands-on laboratory

This section assumes you have a basic knowledge of working inside the Business Integration perspective of WID and developing integration artifacts like SCA exports and imports and mediation modules. You can also refer to the *Business Process Management Samples & Tutorials - Version 6.1* Web site:

<http://publib.boulder.ibm.com/bpcsamp/index.html>

The article *Manage service availability dynamically using WebSphere Enterprise Service Bus and WebSphere Service Registry and Repository* provides more hands-on exercises demonstrating dynamic routing of services. It is available through the developerWorks Web site:

[http://www.ibm.com/developerworks/websphere/library/techarticles/0804\\_perepa/0804\\_perepa.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0804_perepa/0804_perepa.html)

In this lab, we demonstrate how to create the modules needed for the two use cases. You then register the necessary documents into WSRR prior to testing the use cases.

## **Architecture**

For our dynamic business process selection use case, we create two process components:

- ▶ **ECMParticipantComments**  
Triggered by a Request\_ECR\_comments write\_message request operation
- ▶ **ECMParticipantDetailingAndComments**  
Triggered by a Notify\_ECR\_creation write\_message request operation

The input request is in the form of a PLM Services 2.0 container. The PLM interface we use is PLM\_message\_connection as this request is typically an asynchronous one. The operation we use is write\_messages.

The routing selection is based on what contents we get in the Approval\_status/status\_name element of our input PLM Container 2.0.

For our dynamic service selection use case, we create two Web services:

- ▶ **Endpoints\_PLM\_general\_connectionExport1 service**  
Provided by the La Gaudé mechanical design department handling truck cabin designing
- ▶ **Endpoints\_PLM\_general\_connectionExport2 service**  
Provided by the Boeblingen mechanical design department handling truck chassis designing

The input request is also in the form of a PLM Services 2.0 container. The PLM interface we use is PLM\_general\_connection as this request is typically a synchronous one. The operation we use is import\_data.

The service selection is based on what contents we get in the Application\_context/Application\_domain and Organization/Organization\_name elements of our input PLM Container 2.0. Typically, you can get several of these content data in element arrays. We are taking a simple scenario where we receive a PLM Container that carries a single application domain and organization.

Our architecture uses a mediation module with the following components:

- ▶ Endpoint Lookup primitives (ELP) to hold the criteria for selecting the appropriate BPEL or service endpoints as well as performing the look up of WSRR to get the matching endpoint.

This endpoint is a SCA export for our dynamic BPEL selection use case. It is a Web service port for our dynamic service selection use case.

- ▶ SetMessageType primitives to map weakly-typed input elements
- ▶ Enhanced Callout Nodes to dynamically bind/invoke the endpoints

The dynamic BPEL selection architecture is illustrated in the mediation flow shown in Figure 10-14. The Operation we use is write\_messages.

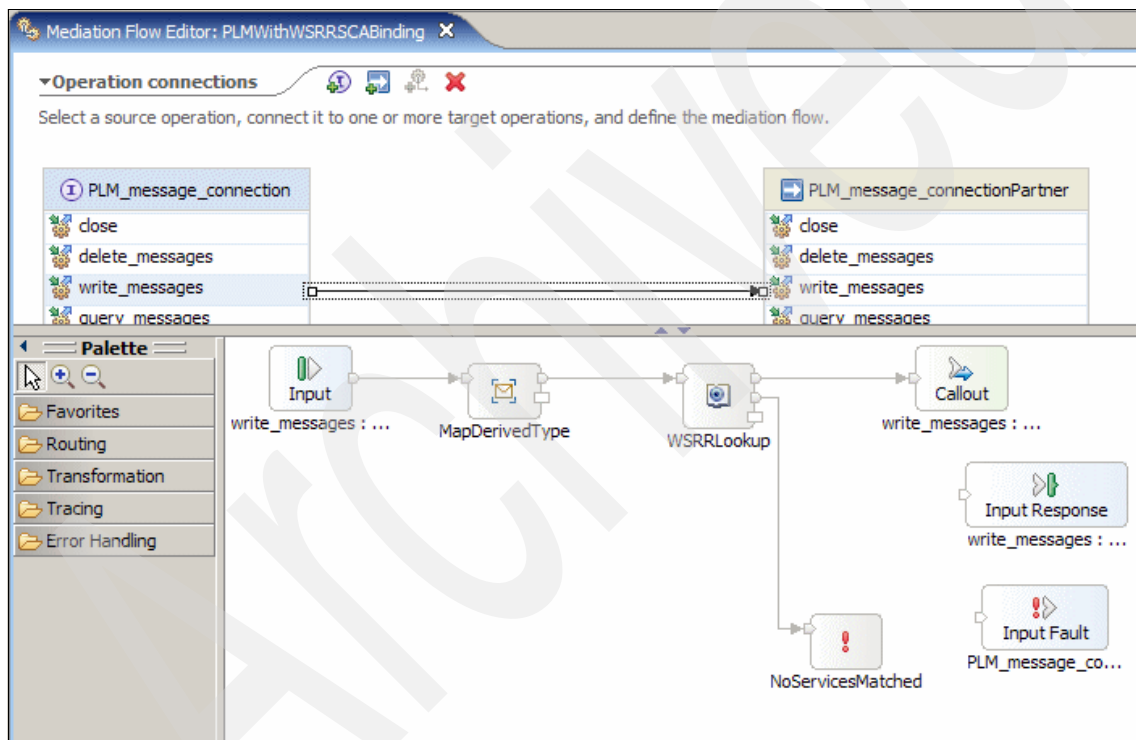


Figure 10-14 Dynamic BPEL selection mediation flow

The dynamic service selection architecture is illustrated in the mediation flow shown in Figure 10-15. The Operation we use is `import_data`.

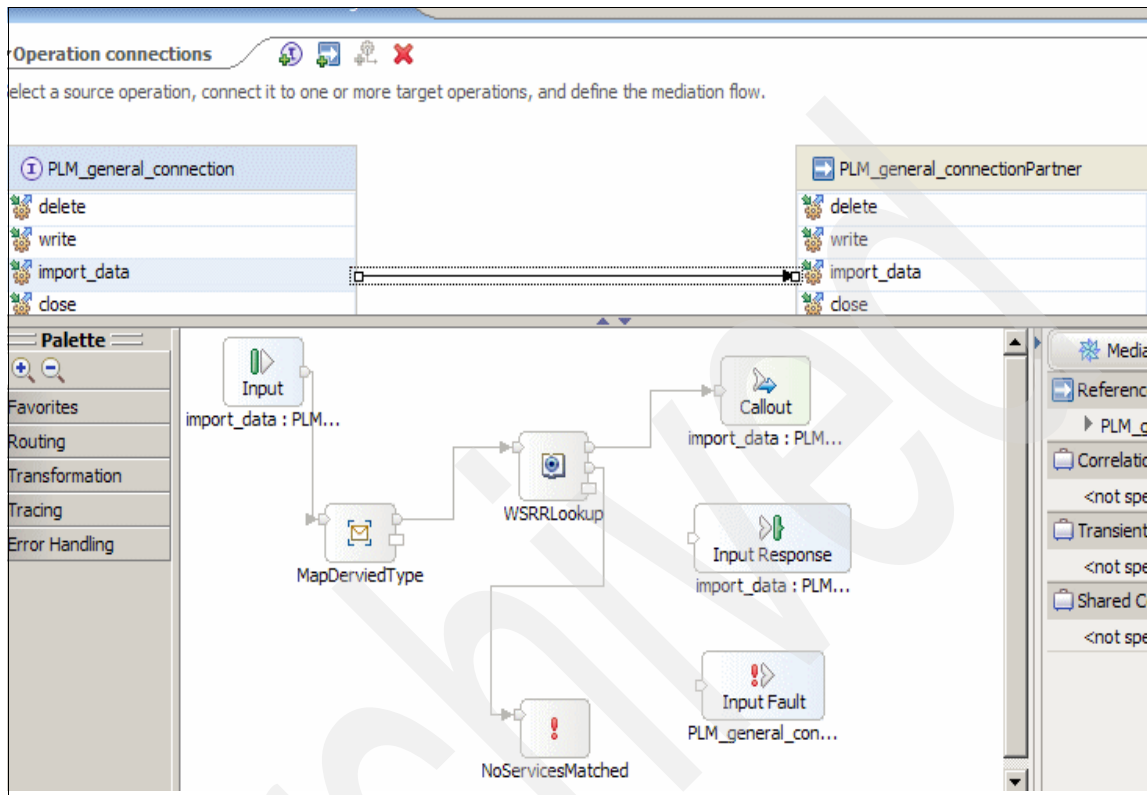


Figure 10-15 Dynamic service selection mediation flow

### Introduction to development steps

This section lists the tasks we need to perform to build our sample use cases. It also provides an overview of the tasks involved in each step.

1. Create library `PLMServices20` to hold the common definition of PLM Services 2.0 interfaces, data types, and Web service ports.
2. Create/import mediation modules.
  - `PLMMediationSCABinding`: For dynamic BPEL selection
  - `PLMMediationWSBinding`: For dynamic service selection
3. Implement the mediation flows logic.

4. Create/import process application PLMProcess with two business processes (BPELs).
  - ECMParticipantComments
  - ECMParticipantDetailingAndComments
5. Create two Java bean implementation-based Web services as dummy Web service bindings for the dynamic BPEL selection.
  - Endpoints\_PLM\_message\_connectionExport1.wsdl
  - Endpoints\_PLM\_message\_connectionExport2.wsdl
6. Create two Java bean implementation-based Web services for the dynamic service selection.
  - Endpoints\_PLM\_general\_connectionExport1.wsdl
  - Endpoints\_PLM\_general\_connectionExport2.wsdl
7. Specify ELP User Properties in the mediation flows.
  - PLMWithWSRRSCABinding
  - PLMMediationWSBinding
8. Populate WSRR metadata using the process module EAR file PLMProcessApp.ear.
9. Per-use case testing using the Integrated Test Component (ITC) in WID V6.1.

**Note:** In the following steps, we will create the common library. Start your WebSphere Integration Developer and use a new workspace name. For instance, use sa7593wsrr. Go to the Business Integration perspective.

10. Follow the steps described in Section 7.2, “PLM Service 2.0 library and Java utilities” on page 181. Use PLMServices20 as the library name.

Alternatively, you can perform the following procedure:

- a. Download the PLMWithWSRR.zip file from the Appendix A, “Additional material” on page 465. Extract the file into a local directory. There is a resulting folder PLMWithWSRR with two sub folders.
- b. In the Business Integration view, create a new library in your workspace and name it PLMServices20.
- c. Import all the PLM Services artifacts
  - i. Highlight your library. Go to **File** → **Import** → **General** → **File System** → **Next**.
  - ii. Browse and navigate to the local directory where you put PLMWithWSRR folder and import all the artifacts from the PLMServicesDownload sub folder.

**Note:** The latest version of the OMG PLM Services 2.0 is available from this Web site:

<http://schema.omg.org/specs/PLM/2.0/schema/>

11. Ensure that these three PLM Services 2.0 interfaces are imported:

- PLM\_connection\_factory
- PLM\_general\_connection
- PLM\_message\_connection

Your expanded library folder has three imported interfaces, as shown in Figure 10-16.

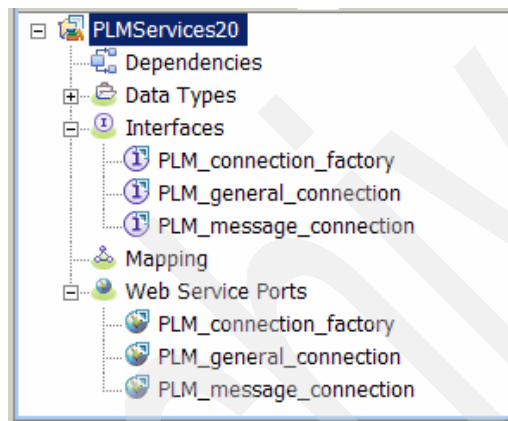


Figure 10-16 PLM Services library import

### ***Building the dynamic business process selection project***

We now create the mediation module for our dynamic business process (BPEL) selection.

1. Right-click anywhere on the Business Integration view. Select **New** → **Mediation Module**.
2. In the Mediation Module panel shown in Figure 10-17 on page 425, enter PLMMediationSCABinding as the Module name and change the mediation component name to PLMWithWSRRSCABinding. Take all other defaults and click **Next**.

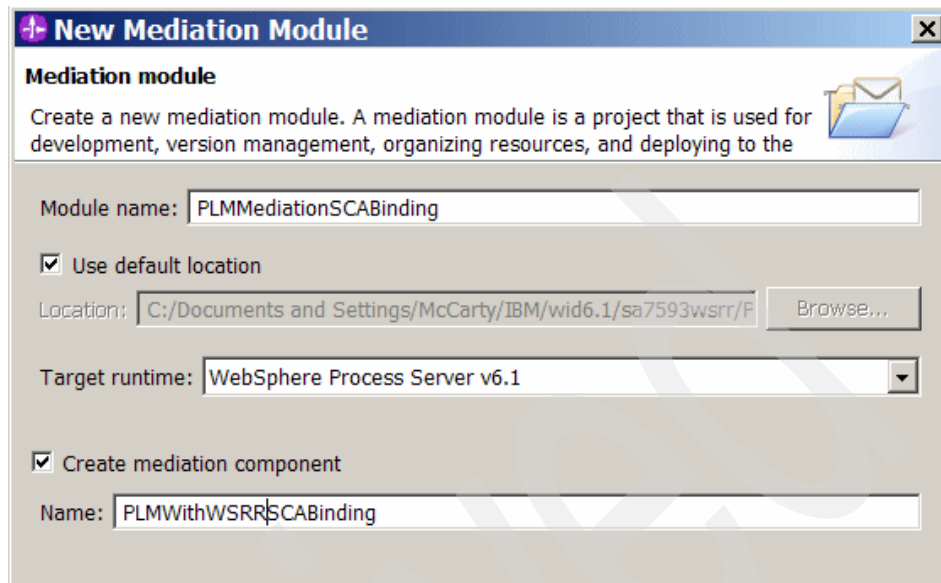


Figure 10-17 Mediation module creation

3. In the Select required libraries window, select **PLMServices20** → **Finish**.
4. In the Business Integration view, you must see your new mediation module **PLMMediationSCABinding** and the mediation component **PLMWithWSRRSCABinding** on the Assembly Diagram panel.
5. In the Assembly Diagram panel, click **PLMWithWSRRSCABinding** component to select it.
  - a. Click **Add Interface** → **Show WSDL** → **PLM\_message\_connection** → **OK**.
  - b. Click **Add a Reference** → **PLM\_message\_connection** → **OK**. The reference name is defaulted to **PLM\_message\_connectionPartner**.
6. Right-click the mediation component. Select **Wire References to New** → **Imports**.  
 An import component **PLM\_message\_connectionPartner** is generated on the assembly diagram.
  - a. Right-click **PLM\_message\_connectionPartner** import. Select **Generate Binding** → **SCA Binding**.
  - b. Go to **Properties** → **Bindings**. The Binding Properties panel at the bottom opens. Enter dummy values in the Module name and Export name fields. Enter Ctrl-S to save your entries.

**Attention:** These dummy values are to force the creation of the two elements, module and export, in the XML, PLM\_message\_connectionPartner.import, that is generated for the import component and is visible in the workspace using the **Resource** perspective. You can remove the dummy values afterwards and save the component properties again.

- c. If the PLMWithWSRRSCABinding shows an error, right-click the component. Choose **Synchronize Interfaces and References → to Implementation**.
  - d. Press Ctrl-S to save your changes.
7. Create the RequestBuilder java component

Our test requires one parameter, either Request\_ECR\_comments or Notify\_ECR\_creation, to trigger the BPEL selection. This parameter is in an element inside a PLM Services 2.0 container, which is a derived type of PLM\_core\_container.

**Important:** At the time of writing of this Redbooks publication, we encountered an issue when using the Integrated Test Client (ITC) inside WebSphere Integration Developer (WID). Upon invoking a test, ITC could not recognize the array elements of our PLM\_container (our input request) that is a derived type of the PLM\_core\_container. The WSADIE/WID Service of IBM has acknowledged this as a product defect (JR29055). The fix is packaged into ifix001 for WID 6.1.0.1.

Due to a possible Integrated Test Client (ITC) limitation (see Important box above), we enter our parameter in the UID parameter of the PLM\_core\_container. This necessitates the creation of a java component that picks up this UID parameter and populate the appropriate array element of the derived type, PLM\_container.

For this use case, the request parameter, either Request\_ECR\_comments or Notify\_ECR\_creation, is transferred to the Status\_name of the Approval\_status array element.

- a. From the Palette, click **Java component** and drop it on the assembly diagram panel before the PLMWithWSRRSCABinding component and change the name to RequestBuilder.
- b. Click **RequestBuilder** to highlight it and click the **Add Interface** icon to bring up the **Add Interface**. Select the Show WSDL check box to filter the list. Select the PLM\_message\_connection WSDL interface. Click **OK**.



- c. Now click the **Add Reference** icon to bring up the Add Reference window and again select the PLM\_message\_connection WSDL interface. Click **OK**.

When prompted to convert the WSDL interfaces to use Java interfaces, click **Yes**.

- d. Right-click **RequestBuilder** → **Generate Implementation** → **(default package)** or **sca.component.java.impl** (if listed). Click **OK**.
- e. The java editor opens the implementation code. Add the following import statements after the package declaration:
- import java.util.ArrayList
  - import java.util.List
  - import com.ibm.websphere.sca.sdo.DataFactory
- f. Replace the following public DataObject write\_messages {} method implementation with the code shown in Example 10-1.

```
//TODO Needs to be implemented.  
return null;
```

*Example 10-1 RequestBuilder write\_message method*

---

```
/*  
 * Initialization of elements of PLM request  
 */  
DataFactory dataFactory = DataFactory.INSTANCE;  
DataObject plmcontainer = dataFactory.create(  
    "urn:omg.org/plm20/informational/model", "PLM_container");  
DataObject plmmessage = dataFactory.create(  
    "urn:omg.org/plm20/informational/core", "PLM_message");  
DataObject approvalstatus = dataFactory.create(  
    "urn:omg.org/plm20/informational/model", "Approval_status");  
String tmp = aMessagesAndBooleanParameter.getString(  
    "messages[1]/PLM_core_container/@uid");  
plmcontainer.setString("uid", tmp);  
List approvalstatuslist = new ArrayList();  
approvalstatuslist.add(approvalstatus);  
/*  
/* Filling in of the new PLM container of the request. Here we are  
using the UID of the PLM_core_container instead of the actual element  
Status_name of the array element Approval_status because of the ITC  
problem of handling derived type array elements.  
*/  
approvalstatus.setString("Status_name", tmp);  
plmcontainer.setList("Approval_status", approvalstatuslist);
```

```

/*
 * Launch of the new PLM request
 */
aMessagesAndBooleanParameter.setDataObject(
    "messages[1]/PLM_core_container", plmcontainer);
List messages = new ArrayList();
messages = aMessagesAndBooleanParameter.getList("messages");
messages.add(plmmessage);
return locateService_PLM_message_connectionPartner().write_messages(
    aMessagesAndBooleanParameter);

```

Take note of the section marked “/\* Filling in of the new PLM container of the request.” This loads the UID of the PLM\_core\_container (stored in the variable tmp into the Approval\_Status/Status\_name array element of the PLM\_container.

- g. Enter Ctrl-S to save the RequestBuilder component.
8. In the PLMMediationSCABinding Assembly Diagram editor, connect or wire the RequestBuilder component to the PLMWithWSRRSCABinding component and press Ctrl-S to save.

If the PLMWithWSRRSCABinding still shows an error, right-click the component. Select **Synchronize Interfaces and References** → **to Implementation**. Enter Ctrl-S to save your changes.

Your PLMMediationSCABinding module now displays in the Business Integration view, as in Figure 10-18.

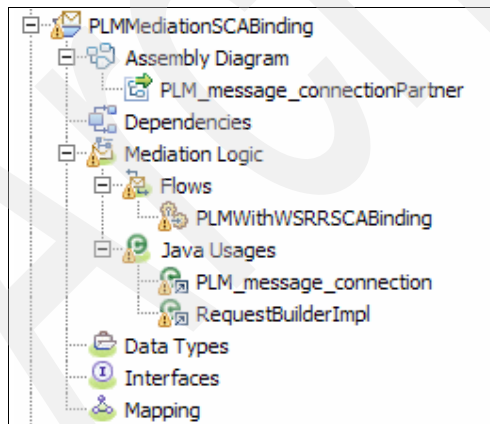


Figure 10-18 PLMMediationSCABinding module

Your Assembly Diagram will look like the diagram shown in Figure 10-19.

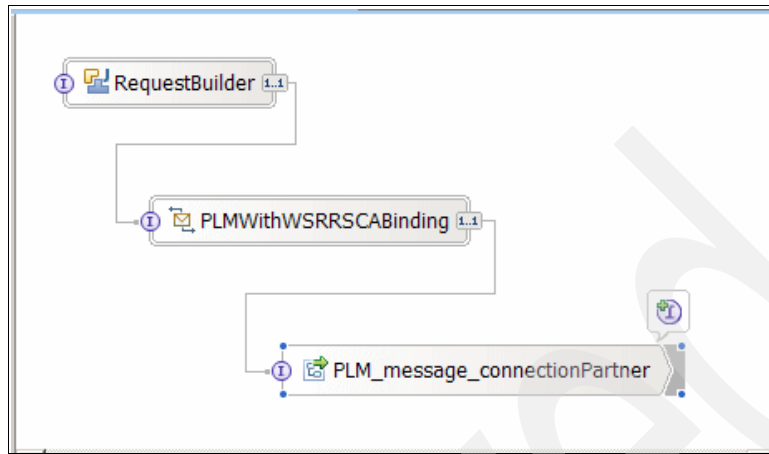


Figure 10-19 PLMMediationSCABinding module

9. Build the business process selection mediation logic flow
  - a. Double-click **PLMWithWSRRSCABinding** to open the mediation flow diagram for editing.
  - b. Create an Operation Connection (draw a line) from write\_messages operation of the PLM\_message\_connection interface to the write\_messages operation of the PLM\_message\_connectionPartner reference.
  - c. Click the operation connection to highlight it. This opens the mediation flow diagram in the Request:write\_messages view.
  - d. Complete the mediation logic.
    - i. Expand the Routing tab in the Palette on the left-hand side and drop an Endpoint Lookup primitive (ELP) on the mediation flow and change the Display name to WSRRLookup.
    - ii. Expand the Transformation tab and drop a SetMessageType primitive on the mediation flow before the WSRRLookup primitive and change the Display name to MapDerivedType.
    - iii. Expand the Error Handling tab and drop a Fail primitive on the mediation flow below the WSRRLookup primitive and change the Display name to NoServicesMatched.
    - iv. Connect the out: terminal of the Input Node to the in: terminal of the MapDerivedType primitive.

- v. Connect the out: terminal of the MapDerivedType primitive to the in: terminal of the WSRRLookup primitive.
- vi. Connect the out: terminal of WSRRLookup primitive to the in: terminal of Callout component.
- vii. Connect the noMatch: terminal of WSRRLookup component to the in: terminal of NoServicesMatched component.
- viii. Press Ctrl-S to save the changes so far.

The completed mediation flow looks similar to that shown in Figure 10-20.

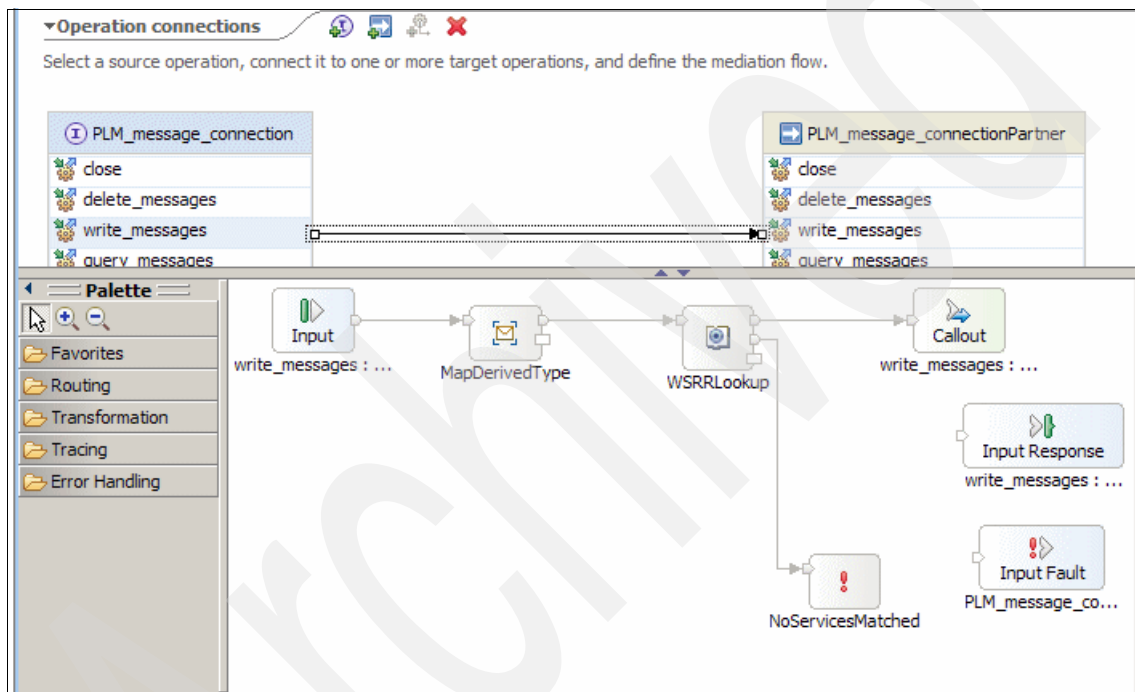


Figure 10-20 BPEL selection mediation flow diagram

e. Configuring the MapDerivedType primitive

Highlight the MapDerivedType primitive. Click **Properties** → **Description**. Read the details to understand the role of this SetMessageType primitive in the mediation flow.

i. Click **Details** → **Add** to introduce some Message field refinements

ii. In the Add/Edit properties window, enter the following values:

- Weakly typed field:

Click **Edit** to open the Data Types Viewer window. Expand **ServiceMessageObject** → **body** → **write\_messages** → **messages**. Double-click **PLM\_core\_container** and enter 1 in the Element declaration occurrence window to get the following XPath expression as shown in the Data Types Viewer of the XPath Expression® Builder shown in Figure 10-21:

/body/write\_messages/messages[1]/PLM\_core\_container

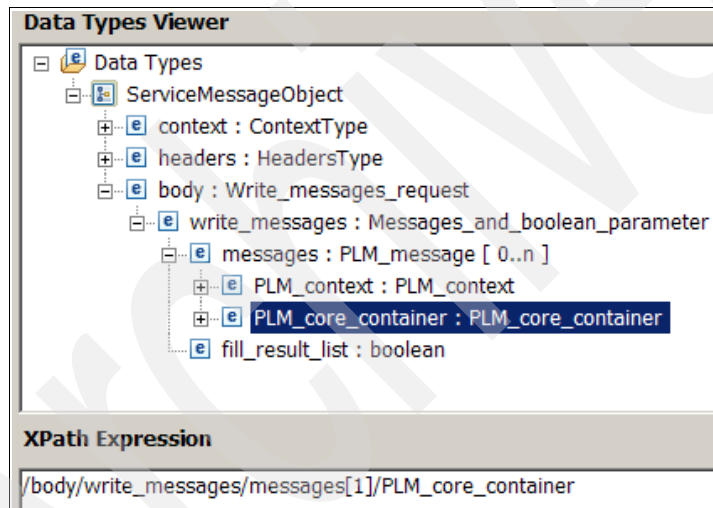


Figure 10-21 MapDerivedType XPath

- Actual field type:

Click **Browse** and enter PLM in the Filter prompt. Select **PLM\_container** → **OK**. The returned value is as follows:

{urn:omg.org/plm20/informational/model}PLM\_container

This mapping gives us access to the individual elements in the input PLM\_container.

iii. Press Ctrl-S to save your changes.

f. Configuring the WSRRLookup primitive

Highlight the WSRRLookup primitive. Click **Properties** → **Description** tab and read the details to understand the role of this EndPoint Lookup primitive in the mediation flow.

- i. From the Details tab, click **Browse** to select an Interface to pick up PLM\_message\_connection and click **OK**.
- ii. Keep the following value defaulted in Namespace:  
urn:omg.org/plm20/services/message
- iii. For Registry Name, enter the name of the WSRR definition you set up in “WESB/WPS configuration” on page 414. For instance, WSRR, or you can keep the <Use default registry> entry.
- iv. For our exercise, keep the value Return first matching endpoint and set routing target for the Match Policy.

**Note:** Other choices for Match Policy (return all matching endpoints, return all matching endpoints, and set alternate routing routes) are available. These require wiring your ELP to another custom mediation primitive (instead of to the CallOut node) to handle the routing logic. Refer to the WebSphere Integration Developer V6.1 Information Center at the following Web page for details on this topic:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.sib.mediation.ui.doc/topics/twsrrmany.html>

- v. Go to the Advanced tab to enter User Properties for WSRRLookup. Scroll down to the User Properties panel. Click **Add** to get the Add/Edit properties for the component. Here we retrieve the PLM\_container element that we use to look up WSRR. Enter the following information for the property arguments:

- Name: status\_name.

We use this name as a property against our SCA exports that we register later on in WSRR as detailed in “Registering the PLMProcess module in WSRR” on page 444.

- Type: XPath (default)
- Value: the XPath expression.

Click **Edit** to open the Data Types Viewer window. Expand **ServiceMessageObject** → **body** → **write\_messages** → **messages** → **PLM\_core\_container** → **Approval\_status**.

Double-click **Status\_name: string** to get the XPath expression as shown in Figure 10-22.

/body/write\_messages/messages/PLM\_core\_container/Approval\_status/Status\_name

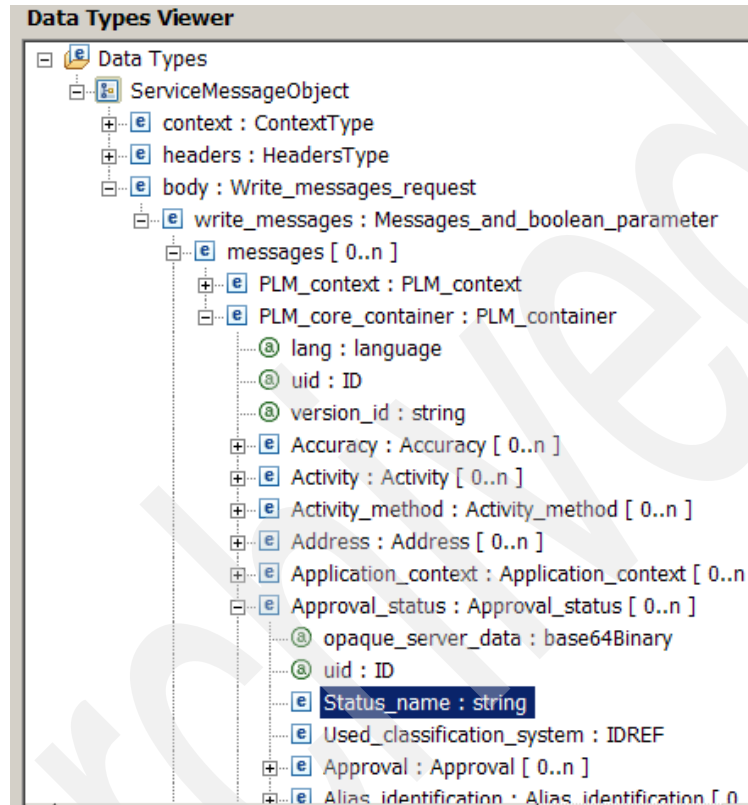


Figure 10-22 Status\_name XPath

The completed User Properties panel now has entries as shown in Figure 10-23.

Name:	status_name
Type:	XPath
Value:	/body/write_messages/messages/PLM_core_container/Approval_status/Status_name

Figure 10-23 Completed Status\_name User Property

- vi. Click Promoted Properties. In this panel you can specify which property of the EndPoint Lookup primitive you want to change dynamically at runtime. This is done by selecting the Promoted check box as shown in Figure 10-24.

Property	Promoted	Alias	Alias value
Registry Name	<input type="checkbox"/>		
Name	<input type="checkbox"/>		
Match Policy	<input type="checkbox"/>		
Version	<input type="checkbox"/>		
Value{/body/write_messages/messages/PL...	<input checked="" type="checkbox"/>	BPEL_Lookup	/body/write_mess...
Namespace	<input type="checkbox"/>		

Figure 10-24 Promoted ELP properties

The promoted property is accessible through the WebSphere Process Server administration console and you can change the value of the User Property here.

- g. Configure the NoServicesMatched primitive.  
Highlight the NoServicesMatched primitive. Select **Properties** → **Details**. Enter **No services matched!** in the Error message entry field.
- h. Configure the Callout Node.  
Highlight the Callout endpoint node. Go to **Properties** → **Details**. Ensure that the Use dynamic endpoint if set in the message header check box is selected as shown in Figure 10-25. This is set because we are using the dynamic endpoint selection capability of WESB.

**Callout : write\_messages : PLM\_message\_connectionPartner**

Reference name:

Operation name:

☒ Use dynamic endpoint if set in the message header

Async timeout (seconds):

Figure 10-25 PLMMediationSCABinding Callout endpoint



- i. Configure the Response.
  - i. Go to the Response:write\_messages view of the flow editor. Connect the out: terminal of the Callout Response node to the in: terminal of the Input Response node.
  - ii. Expand the Error Handling tab and drop a Fail primitive on the mediation flow below the Callout Response node.
  - iii. Connect the fail: terminal of the Callout Response node to the in: terminal of the Fail1 primitive.
  - iv. Highlight the Fail1 primitive. Go to **Properties** → **Details** and enter **Error in response!** in the Error message field.
  - v. Press Ctrl-S to save your changes.
10. Go back to the open PLMMediationSCABinding Assembly Diagram panel and press Ctrl-S to save the changes.
11. Close all open editors and clean the projects in the workspace. Click the Project option in the WID menu tool bar. Select **Clean** → **Clean all projects**.

### ***Creating the Web Services bindings module***

We now create the module to contain our dummy Web services artifacts for dynamic BPEL selection.

1. Create the PLM\_message\_connectionExport1 SCA export.
  - a. In the Business Integration view, go to **File** → **Create** → **New Module**. Use Endpoints as Module Name and ensure **Open module assembly diagram** is selected. Click **Next**.
  - b. Select the PLMServices20 library to use and click **Finish**. The Assembly Diagram panel opens for edit.
  - c. Expand **PLMServices20** → **Interfaces**. Drag and drop the PLM\_message\_connection WSDL interface on the Assembly Diagram editor. Choose **Export with Web Service Binding** in the Component Creation window. Click **OK**. Select soap/http for Transport Selection. Click **OK**.
  - d. When prompted for Dependencies, add the PLMServices20 library under Libraries in the Dependencies window. Press Ctrl-S to save the entry. Close the Dependencies window.

You now have a PLM\_message\_connectionExport1 SCA export component in your assembly diagram panel.

- e. Expand **PLMServices20** → **Web Service Ports**. You have the following entry:

PLM\_message\_connectionExport1\_PLM\_message\_connectionHttpPort

You need to configure the correct soap address later as detailed in "Configure the Web Service Ports" List number (5) on page 438.

2. Create the NotifyECR java component.
- From the Palette, click **Java**, drop it on the editor panel, and name it NotifyECR.
  - Right-click **NotifyECR**. Click **Add** → **Interface** → **Show WSDL** → **PLM\_message\_connection** → **OK**.
  - Right-click on the **Java component**. Click **Generate Implementation** → **default package** (or **sca.component.java.impl** if it is shown). The implementation java source code editor opens.
  - Add the following imports after the package declaration:
    - import java.util.ArrayList
    - import java.util.List
    - import com.ibm.websphere.sca.sdo.DataFactory
  - Search for the write\_messages method and replace the following implementation with the code shown in Example 10-2.  

```
//TODO Needs to be implemented.  
return null;
```

*Example 10-2 Java implementation: Dummy WS binding*

---

```
/*  
 * Instantiation of the response message  
 */  
DataFactory dataFactory = DataFactory.INSTANCE;  
DataObject processinginformationparameter = dataFactory.create(  
    "urn:omg.org/plm20/services/parameter",  
    "Processing_informations_parameter");  
DataObject processinginformation = dataFactory.create(  
    "urn:omg.org/plm20/computational/core",  
    "PLM_processing_information");  
processinginformation.setString("Message",  
    "Response from Notify_ECR_creation!");  
List responselist = new ArrayList();  
responselist.add(processinginformation);  
processinginformationparameter.setList("return", responselist);  
System.out.println("Notify_ECR_creation services");  
return processinginformationparameter;
```

---

**About this procedure:** We simulate the message response we get when our ECOMPicipantDetailingAndComments BPEL is triggered by our Notify\_ECR\_creation request. We are printing out in the WPS test environment console log a message that we have reached this BPEL.

- f. Press Ctrl-S to save your changes.
3. Connecte the components.
  - a. Wire the PLM\_message\_connectionExport1 to the NotifyECR component on the Endpoints Assembly Diagram.
  - b. Press Ctrl-S to save your Endpoints module.
4. Create the other components.
  - a. Create another SCA export PLM\_message\_connectionExport2 and another java component with a name of RequestECR as in "Create the NotifyECR java component." List number (2) on page 436.
  - b. Double-click **RequestECR** component to open the code editor. Add the following imports after the package declaration:
    - import java.util.ArrayList
    - import java.util.List
    - import com.ibm.websphere.sca.sdo.DataFactory
  - c. Replace the write\_messages method implementation with the code shown in Example 10-3.

*Example 10-3 Java implementation: WS Binding 2*

---

```
/*
 * Instantiation of the response message
 */
DataFactory dataFactory = DataFactory.INSTANCE;
DataObject processinginformationparameter = dataFactory.create(
    "urn:omg.org/plm20/services/parameter",
    "Processing_informations_parameter");
DataObject processinginformation = dataFactory.create(
    "urn:omg.org/plm20/computational/core",
    "PLM_processing_information");
processinginformation.setString("Message", "Request_ECR_comments!");
List responselist = new ArrayList();
responselist.add(processinginformation);
processinginformationparameter.setList("return", responselist);
System.out.println("Request_ECR_comments services");
return processinginformationparameter;
```

---

**About this procedure:** Here we simulate the message response we get when our ECMParticipantComments BPEL is triggered by our Request\_ECR\_comments request. We are also printing out in the WPS test environment console log a message indication that we have reached this particular BPEL.

- d. In the Endpoints Assembly Diagram, wire the PLM\_message\_connectionExport2 to RequestECR component and press Ctrl-S to save the module. The assembly diagram appears as in Figure 10-26.

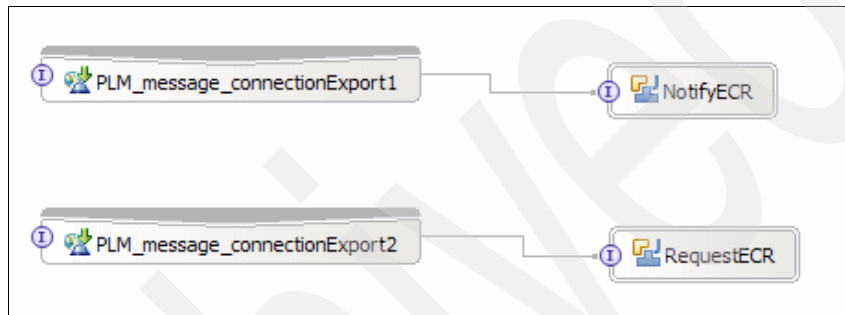


Figure 10-26 Endpoints assembly diagram: BPEL use case

## 5. Configure the Web Service Ports

In the Business Integration View, expand **PLMServices20** → **Web Service Ports**. There are two new ports created:

- PLM\_message\_connectionExport1\_PLM\_message\_connectionHttpPort
- PLM\_message\_connectionExport2\_PLM\_message\_connectionHttpPort

- a. Double-click each of these ports to open the WSDL editor. Go to the Source view. Locate the following soap address lines near the bottom:

```
<soap:address
location="http://localhost:9080/EndpointsWeb/sca/PLM_message_connectionExport1"/>
```

```
<soap:address
location="http://localhost:9080/EndpointsWeb/sca/PLM_message_connectionExport2"/>
```

Replace the default port 9080 with the WC\_defaulthost port that you are using for your WPS.

- b. Press Ctrl-S to save your changes. Close all open editors.

6. Deploy the Web services.

Ensure your WPS is started. Highlight the server. Right-click **Add and Remove Projects**. Add your Endpoints module to deploy it to the server and click **Finish**. If the server State still shows Republish due to SOAP Invoke errors like those shown in Figure 10-27, right-click the server again and select **Publish**.

Publish failed with errors, warnings or both. Please see server logs for more details.  
ADMC0009E: The system failed to make the SOAP RPC call: invoke  
ADMC0009E: The system failed to make the SOAP RPC call: invoke  
ADMC0009E: The system failed to make the SOAP RPC call: invoke

Figure 10-27 Publishing errors

7. Test the Web services.

When the publishing is finished, open a browser and test the Web service soap addresses and make sure you get a response similar to that shown in Figure 10-28.

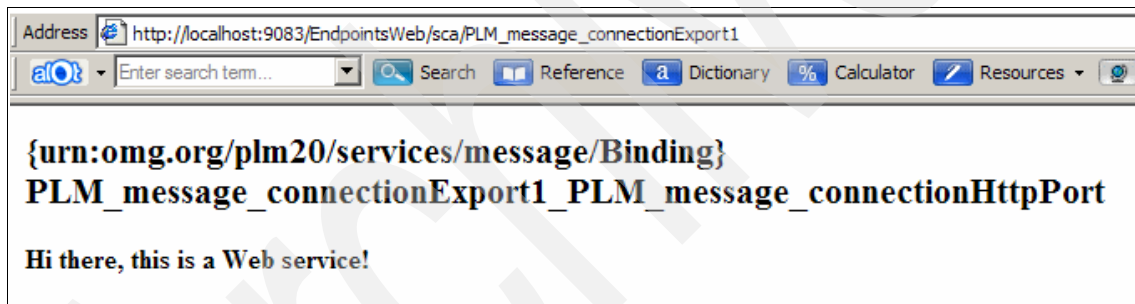


Figure 10-28 PLM\_message\_connection Web service

If your Endpoints module has been deployed on the server and testing does not show the Web page with the message “Hi there, this is a Web service!”, check the following deployment settings:

- a. Right-click **Endpoints** project. Select **Open Deployment Editor**.
- b. In the Module Deployment Editor, click the **Exports** tab.
- c. In the Web Service Exports panel, highlight one of the exports. For instance, PLM\_message\_connectionExport1.
  - i. Expand the **URL Mappings**.
  - ii. Ensure that the Include default mapping check box is selected.

- d. Repeat step c for the second Web service export.
- e. Press Ctrl-S to save the changes. Answer **No** on the Inconsistent Files prompt to update the editor as shown in Figure 10-29.

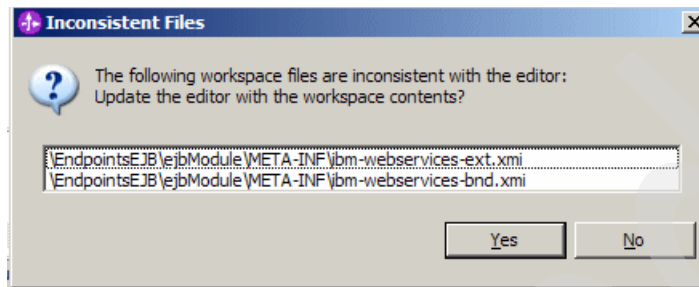


Figure 10-29 Inconsistent Files prompt

### ***Examining the business process module***

We have created a module where our business processes for selection are defined. You can now import it into your workspace for analysis.

1. Import the business process module PLMProcess
  - a. Go to the local directory where you extracted the PLMWithWSRR.zip file and locate the PLMProcess.zip file in the PLMWithWSRR folder. Import it into your workspace as a Project Interchange. Select all projects to import:
    - PLMProcess
    - PLMProcessEJB
    - PLMProcessWeb
    - PLMProcessApp
  - b. After the import, when the Build workspace task finishes, click the **Project** option in the WID menu tool bar. Select **Clean**. Choose **Clean all projects**.
2. Expand **PLMProcess** and double-click **Dependencies**. Ensure that the project library PLMServices20 you created in “Note: In the following steps, we will create the common library. Start your WebSphere Integration Developer and use a new workspace name. For instance, use sa7593wsrr. Go to the Business Integration perspective.” on page 423 is listed under Libraries. Otherwise, click **Add** to put it in. Close the window.

3. Double-click **Assembly Diagram** to view the components as shown in Figure 10-30.

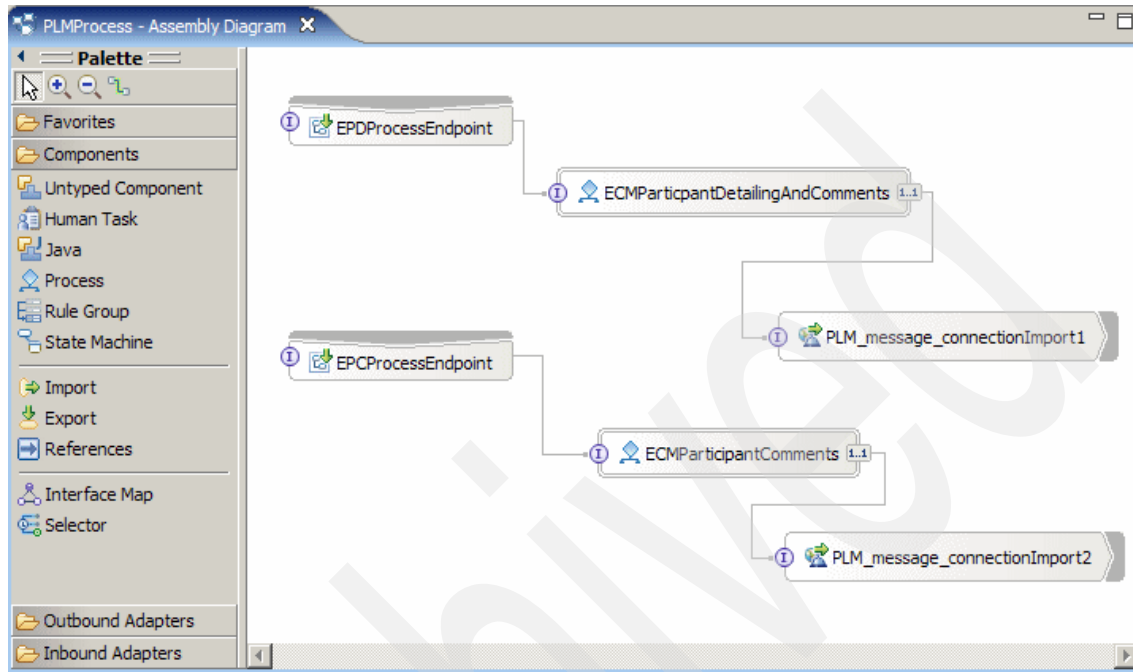
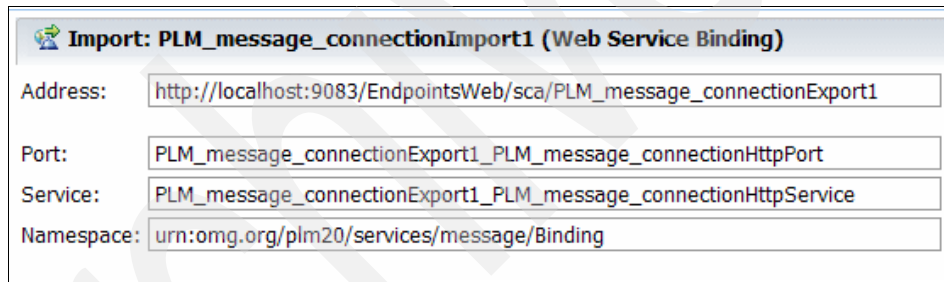


Figure 10-30 PLMProcess assembly diagram

- There are two SCA exports.
  - EPDProcessEndpoint  
This export invokes ECMParticipantDetailingAndComments process component
  - EPCProcessEndpoint  
This export invokes ECMParticipantComments process component
- There are two SCA imports to provide dummy Web service bindings.
  - PLM\_message\_connectionImport1
  - PLM\_message\_connectionImport2

4. Highlight `PLM_message_connectionImport1`. Go to **Properties** → **Binding** to see the Address, Port, Service, and Namespace information.
  - The Address URL carries the port number that was used for the Redbook. This needs to be corrected to use the correct port of your WebSphere Process Server.
  - The Port field has this value:  
`PLM_message_connectionExport1_PLM_message_connectionHttpPort`
5. Click **Browse** for the Port. Select the same port value, `PLM_message_connectionExport1_PLM_message_connectionHttpPort`, from the Matching service ports list. Click **OK**. This selection automatically corrects the value of the Address URL. Press Ctrl-S to save your changes.
6. If your Endpoints module is still deployed on the WPS, you can test the Address URL in a browser to view the Web service. At the time of the writing of this Redbooks publication, using the latest update of the PLM Services 2.0, the value of the URL as shown in Figure 10-31 is:  
`http://localhost:9083/EndpointsWeb/sca/PLM_message_connectionExport1`



Import: PLM_message_connectionImport1 (Web Service Binding)	
Address:	<code>http://localhost:9083/EndpointsWeb/sca/PLM_message_connectionExport1</code>
Port:	<code>PLM_message_connectionExport1_PLM_message_connectionHttpPort</code>
Service:	<code>PLM_message_connectionExport1_PLM_message_connectionHttpService</code>
Namespace:	<code>urn:omg.org/plm20/services/message/Binding</code>

Figure 10-31 Binding information for Web service

7. Repeat steps 4–6 to correct the Address URL of the second import, `PLM_message_connectionImport2`. Save your changes.
8. Expand **Business Logic** → **Processes** to see the following two business processes:
  - `ECMParticipantComments`
  - `ECPParticipantDetailingAndComments`



9. Double-click each one to view their respective process flow diagrams. The EPCParticipantComments BPEL flow diagram is shown in Figure 10-32.

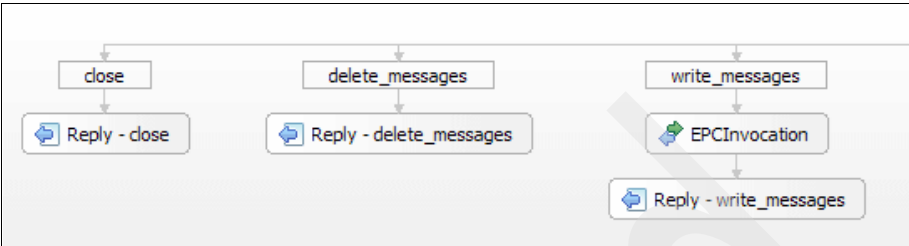


Figure 10-32 Sample BPEL mediation logic flow

10. Highlight the EPCInvocation Invoke component that is between write\_messages operation and the Reply - write\_messages operation. Click **Properties** → **Details**.

This links the input parameter parameters4 of the operation write\_messages to the output parameter parameter5 of the operation Reply - write\_messages as shown in Figure 10-33.

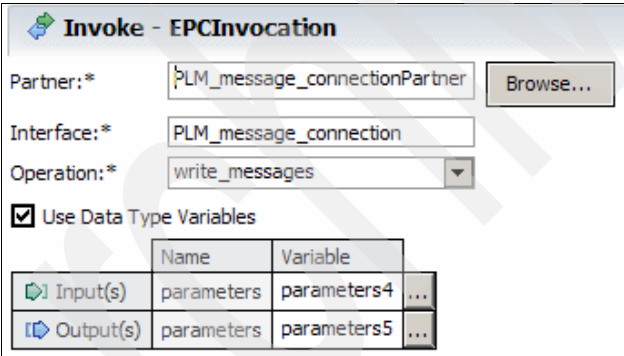


Figure 10-33 EPCInvocation parameters

11. Close all open editors and clean all projects in the workspace again.

### **Registering the PLMProcess module in WSRR**

We now need to load into WSRR the SCA exports that we are using for dynamic BPEL selection, EPDProcessEndpoint, and EPCProcessEndpoint. This is done by using the EAR file of our process module. Because it has a dependency on the PLMServices20 library that holds any Web service ports created, then our two PLM\_message\_connection Web services is also loaded into WSRR.

1. Go to the local directory where you extracted the PLMWithWSRR.zip and locate the PLMProcessApp.ear file in the PLMWithWSRR folder. You need this to import into WSRR.
2. Ensure your WPS test environment server is started. Open a browser and access your WSRR admin console where <port> is your WPS WC\_default host:  
  
`http://localhost:<port>/ServiceRegistry`
3. Select **Administrator** or **GP Administrator** (if you have imported the Governance Profile when you set up WSRR) for the Perspective and click **Go**.
4. Expand **Service Documents** → **Load Documents**.
5. Click **Browse** for Specify Path to the Local file system, navigate to the folder where the PLMProcessApp.ear is located, and select this file.
6. Select **SCA integration module** for Document Type and enter **PLMProcessApp** as Document Description.
7. Click **OK**.
8. After you see the message “The SCA integration module was uploaded successfully.”, expand the **Service Metadata** on the left-hand panel of WSRR and examine what have been imported into WSDL and Concepts.
9. Expand **SCA** → **Exports**. You see the following two Exports:
  - EPCProcessEndpoint
  - EPDProcessEndpoint
10. Enter a new property item for each of the exports:
  - a. Go to **EPCProcessEndpoint** → **Add Property**.
  - b. Enter the following values for the fields listed below:
    - Name: status\_name
    - Value: Request\_ECR\_comments
  - c. Click **OK**.
  - d. Repeat steps a–c for EPDProcessEncpoint. Substitute Notify\_ECR\_creation for the Value field.

You have now annotated the SCA Export with the User Property that is used in your Endpoint Lookup primitive as a criteria for looking up WSRR.

11. Go to **Service Metadata** → **WSDL** → **Ports**. The following two Web services ports created for our first use case are listed:

- PLM\_message\_connectionExport1\_PLM\_message\_connectionHttpPort
- PLM\_message\_connectionExport2\_PLM\_message\_connectionHttpPort

These are dummy Web services. They do not need to be annotated. The endpoints we are getting for dynamic BPEL selection are our SCA exports created in "Build the business process selection mediation logic flow" List number (9) on page 429. These dummy Web services are set up as bindings on the SCA imports in our PLMProcess module. The Web services purpose is to show that the correct BPEL was selected and to get a response.

12. You may now close the WSRR admin console. Inside the WID environment, restart the WPS test server to clear the WSRR cache if you have not set the Time out of cache value to zero when you created your WSRR definition in "WESB/WPS configuration" on page 414.

### ***Testing the dynamic business process selection use case***

We now use the Integrated Test Client (ITC) in your WID to test our first use case.

1. Go to the Servers tab in the lower middle of your workspace and start the WebSphere Process Server. When the server is started, right-click the server and click **Add and Remove Projects**. Ensure that all your modules are added to the server.
2. If the server State shows as Republish, right-click the server. Click **Publish**.
3. When the publishing is finished, open the Assembly Diagram of the PLMMediationSCABinding module.
4. Right-click the **RequestBuilder**. Select **Test Component**. The Integrated Test Client (ITC) opens showing Events, General, and Detailed Properties panels.
  - In the Detailed Properties section, change the Operation to write\_messages.
  - In the Initial request parameters section:
    - i. Right-click **messages**. Click **Add Elements** → **1**.
    - ii. Expand PLM\_core\_container if it is not yet expanded
    - iii. Enter Request\_ECR\_comments as the value of the uid field

Due to an ITC problem explained previously in the section "Create the RequestBuilder java component" List number (7) on page 426, we use the PLM\_core\_container uid element to pass our Status\_name entry parameter.

5. Ensure the Detailed Properties of the Integrated Test Component panel matches the settings shown in Figure 10-34.

**▼ Detailed Properties**

Configuration: Default Module Test

Module: PLMMediationSCABinding

Component: RequestBuilder

Interface: PLM\_message\_connection

Operation: write\_messages

☐ Invoke export using binding

Initial request parameters

Name	Type	Value
parameters	Messages_and_boolean_...	✓
messages	PLM_message[]	60
messages[0]	PLM_message	✓
PLM_context	PLM_context	✓ null
PLM_core_container	PLM_core_container	✓
uid	ID	✓ Request ECR comment
version_id	string	✓ 2.0
lang	language	✓
fill_result_list	boolean	✓ false

Figure 10-34 ITC Detailed properties for dynamic BPEL selection

6. Run the test by pressing the white arrow just above the Invoke entry in the Events panel.
7. In the Select Deployment Location panel, accept the highlighted WebSphere Process Server v6.1. Click **Finish**.

8. Accept the User Login details (admin/admin) if the prompt appears. Click **OK**.  
If the test runs successfully, the ITC Events panel shows entries similar to those shown in Figure 10-35 and the Console messages shows Request\_ECR\_comments services.

**Events**

**Events**

- Invoke (RequestBuilder:write\_messages)
- Invoke started
- Invoke (RequestBuilder:write\_messages)
- Request (RequestBuilder --> PLMMediation)
- Response (RequestBuilder <-- PLMMediation)
- Return (RequestBuilder:write\_messages)
- Invoke returned

**General Properties**

**Detailed Properties**

Module: [PLMMediationSCABinding](#)  
 Component: [RequestBuilder](#)  
 Interface: [PLM\\_message\\_connection](#)  
 Operation: [write\\_messages](#)

Return parameters:

Name	Type	Value
parameters	Processing_informations_p...	✓
return	PLM_processing_information[]	66
return[0]	PLM_processing_information	✓
Message	string	✓ Response Request_ECR_comments!
Object uid	string	✓

**Console**

WebSphere Process Server v6.1 [WebSphere ESB Server] WebSphere Process Server v6.1 (WebSphere v6.1)

```
[3/18/08 10:11:33:203 CET] 00000010 SibMessage I [:] CWSIS1568I: Messaging engine widNode.server
[3/18/08 10:11:33:218 CET] 00000010 SibMessage I [:] CWSIS1568I: Messaging engine widNode.server
[3/18/08 10:11:33:218 CET] 00000010 SibMessage I [:] CWSIS1568I: Messaging engine widNode.server
[3/18/08 10:11:33:218 CET] 00000010 SibMessage I [:] CWSIS1569I: Messaging engine widNode.server
[3/18/08 10:11:33:218 CET] 00000010 SibMessage I [:] CWSIS1569I: Messaging engine widNode.server
[3/18/08 10:12:31:406 CET] 00000055 WSChannelFram A CHFW0019I: The Transport Channel Service has st
[3/18/08 10:12:31:687 CET] 00000057 ServletWrappe I SRVE0242I: [EndpointsApp] [/EndpointsWeb] [PLM_
[3/18/08 10:12:32:015 CET] 00000057 SystemOut O Request_ECR_comments services
```

Figure 10-35 Dynamic BPEL selection successful test

### Examining the PLMMediationWSBinding

Let us now look at the mediation module required to do our dynamic service selection use case. We have created the module for you. The process is similar to building the PLMMediationSCABinding module.

You can now import it into your workspace for analysis.

1. Go to the local directory where you extracted the PLMWithWSRR.zip file and locate the PLMMediationWSBinding.zip file from the PLMWithWSRR folder. Import it into your workspace as a Project Interchange.
2. Click the **Project** option in the menu tool bar. Select **Clean** → **Clean all projects**.
3. Go to PLMMediationWSBinding and expand **Assembly Diagram**, **Mediation Logic** → **Flows**, and **Mediation Logic** → **Java Usages** to view the components as shown in Figure 10-36.

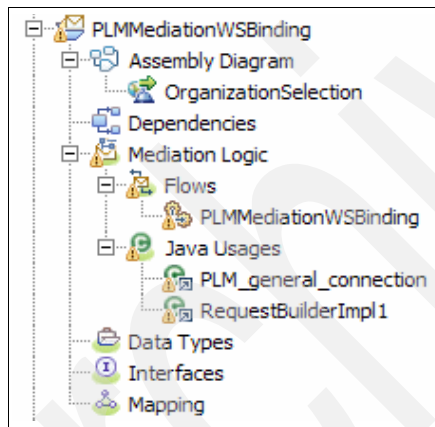


Figure 10-36 PLMMediationWSBinding import

4. Double-click **Dependencies**. Ensure that the project library PLMServices20 you created in “Note: In the following steps, we will create the common library. Start your WebSphere Integration Developer and use a new workspace name. For instance, use sa7593wsrr. Go to the Business Integration perspective.” on page 423 is listed under Libraries. Otherwise, click **Add** to put it in.

5. Double-click **Assembly Diagram** to open the editor shown in Figure 10-37.

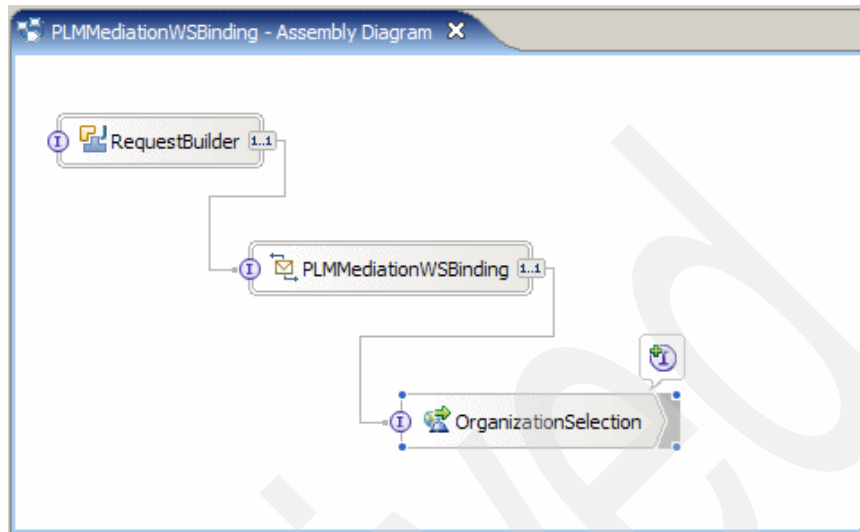


Figure 10-37 Dynamic service selection assembly diagram

6. The diagram shows the following components:

- Java component RequestBuilder
- Mediation flow component PLMMediationWSBinding
- Import component with Web service binding OrganizationSelection

7. Examine the RequestBuilder.

This is our work around for the Integrated Test Client (ITC) problem of recognizing our PLM\_core\_container derived types.

When we test using the ITC, we enter our request parameters, Organization\_name and Application\_domain, as Name/Value pairs of properties of the PLM\_core\_container. In the RequestBuilder, we then pick these values and populate the Organization/Organization\_name and the Application\_context/Application\_domain array elements of our PLM\_container. We use these values in an Endpoint Lookup primitive to do our WSRR look-up.

a. Double-click **RequestBuilder** to open the java implementation.

Look for the import\_data method. The implementation must be like the code shown in Example 10-4 on page 450.

```
/*
 * Initialization of elements of PLM request
 */
DataFactory dataFactory = DataFactory.INSTANCE;
DataObject plmcontainer = dataFactory.create(
    "urn:omg.org/plm20/informational/model", "PLM_container");
DataObject organization = dataFactory.create(
    "urn:omg.org/plm20/informational/model", "Organization");
DataObject applicationcontext = dataFactory.create(
    "urn:omg.org/plm20/informational/model", "Application_context");
String tmp = aContainerAndPropertiesParameter.getString("data/@uid");
plmcontainer.setString("uid", tmp);
List organizations = new ArrayList();
List applicationcontexts = new ArrayList();
organizations.add(organization);
applicationcontexts.add(applicationcontext);
/*
 * Filling in of the new PLM container of the request
 */
List properties = new ArrayList();
properties = aContainerAndPropertiesParameter.getList("properties");
for(int i = 0; i < properties.size(); i++) {
    DataObject property = (DataObject)properties.get(i);
    String name = property.getString("Name");
    String value = property.getString("Value");
    if(name.equals("Organization_name"))
        organization.setString("Organization_name", value);

    else if (name.equals("Application_domain"))
        applicationcontext.setString("Application_domain", value);
}
plmcontainer.setList("Organization", organizations);
plmcontainer.setList("Application_context", applicationcontexts);
/*
 * Launch of the new PLM request
 */
aContainerAndPropertiesParameter.setDataObject("data", plmcontainer);
return
locateService_PLM_general_connectionPartner().import_data(aContainerAnd
PropertiesParameter);
```

---



In the section of the code with comments, “Filling in of the new PLM container of the request”, we are selecting the `Organization_name` and `Application_domain` parameters from the Name/Value properties of the `PLM_core_container` to populate the corresponding array elements of our `PLM_container`. You can now close the RequestBuilder editor.

8. In the Business Integration view, expand **Mediation Logic** → **Flows** if it is not yet expanded. The `PLMMediationWSBinding` is our mediation component that has the logic for the look-up of WSRR for the dynamic selection of the Web service to invoke based on the `Organization_name` and `Application_domain` values in the incoming `PLM_container` input request.

9. Double-click **PLMMediationWSBinding** to open the mediation flow editor.

Highlight the operation connection line between `PLM_general_connection import_data` and the `PLM_general_connectionPartner import_data` to open the mediation flow diagram editor in the `Request:import_data` view.

- a. Configure the `MapDerivedType` primitive.

Click **MapDerivedType** to highlight it. Go to **Properties** → **Details**. Highlight the entry for Message field refinements and click **Edit**. Ensure the following entries have been set:

- Weakly typed field:  
`/body/import_data/data`
- Actual field type:  
`{urn:omg.org/plm20/informational/model}PLM_container`

The `PLM_container` is a derived type of the `PLM_core_container`. This `SetMessageType` primitive configuration ensures our mediation flow has access to the `PLM_container` elements.

- b. Configure the `WSRRLookup` primitive.

Click **WSRRLookup**. This is where we define the rules to look up WSRR.

- i. Go to **Properties** → **Details**. Ensure the Registry Name uses the default registry, or you can put the WSRR definition name you used in the earlier section “WESB/WPS configuration” on page 414.
- ii. Click **Advanced** and scroll down to User Properties. Note that we are assigning the value of two properties:
  - `organization` = `Organization_name` coming from our input `PLM_container`. For our sample scenario, we expect the value to be either Boeblingen or La Gaude, for the two separate departments doing truck design for Cabin and Chassis respectively. This is assigned using the following XPath expression:

`/body/import_data/data/Organization[1]/Organization_name`

This element is obtained by double-clicking the following PLM\_container element:

Organization\_name : string as shown in Figure 10-38.

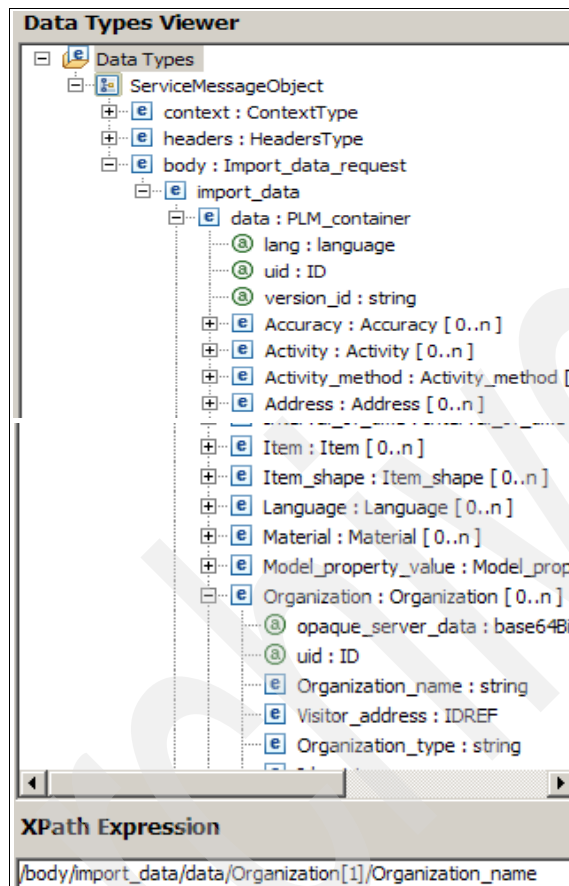


Figure 10-38 Organization\_name XPath expression

- application\_domain = Application\_domain coming from our input PLM\_container. For our sample scenario, we expect the value to be Mechanical design. This is assigned using the following XPath expression:

```
/body/import_data/data/Application_context[1]/Application_domain
```

This expression is obtained by double-clicking the following PLM\_container element:

Application\_domain : string as shown in Figure 10-39 on page 453.

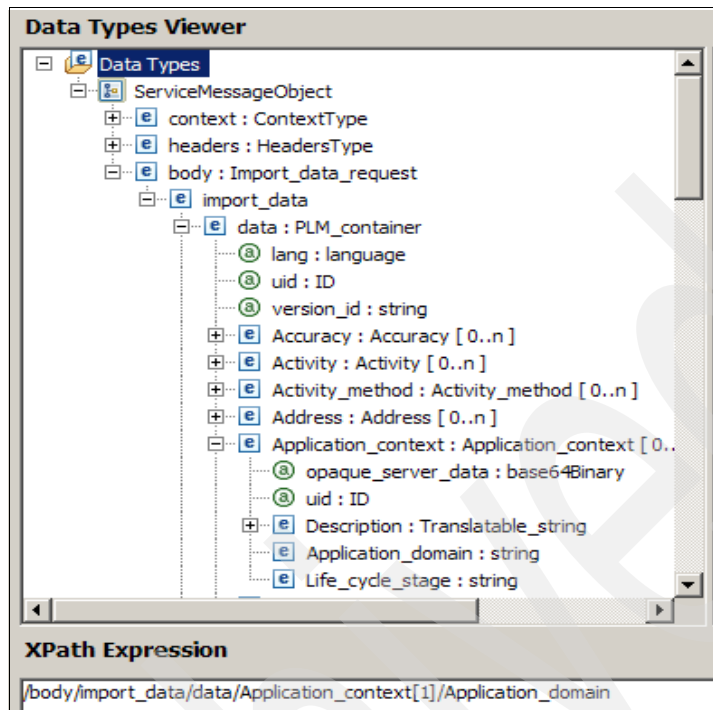


Figure 10-39 Application\_domain XPath expression

We use these organization and application\_domain user properties to annotate the corresponding Web services ports in WSRR.

- c. Configure the Callout Node.
  - i. Highlight the Callout endpoint node.
  - ii. Go to **Properties** → **Details**. Note that the “Use dynamic endpoint if set in the message header” check box is selected as shown in Figure 10-40. This is set because we are using the dynamic endpoint selection capability of WESB.

Callout : import_data : PLM_general_connectionPartner	
Reference name:	PLM_general_connectionPartner
Operation name:	import_data
<input checked="" type="checkbox"/> Use dynamic endpoint if set in the message header	
Async timeout (seconds):	5

Figure 10-40 PLMMediationWSBinding Callout endpoint

- d. Configure the Response.

Go to the Response:import\_data view of the flow editor. Note that the out: terminal of the Callout Response node is wired to the in: terminal of the Input Response node, and the fail: terminal is wired to a Fail1 primitive that was added to the flow.

### Creating the Web Services for dynamic selection

We now create the artifacts required for our dynamic Web service selection use case and put them in the Endpoints module where we created our two dummy Web services for our dynamic business process selection use case.

1. In the Business Integration view, expand the Endpoints module and double-click **Assembly Diagram** to open the panel for editing.
2. Expand **PLMServices20** → **Interfaces** and drag and drop the PLM\_general\_connection WSDL interface onto the Assembly Diagram editor. Choose **Export with Web Service Binding** in the Component Creation window. Click **OK**. Select soap/http for Transport Selection. Click **OK**.
3. If prompted for Dependencies, add the PLMServices20 library under Libraries in the Dependencies window. Press Ctrl-S to save the entry. Close the Dependencies window.

4. Create the LaGaude java component.
  - a. Drag and drop a Java component from the Palette into the editor panel and change the name to LaGaude.
  - b. Click the **Add Interface** icon on the LaGaude component. Check Show WSDL. Select **PLM\_general\_connection** WSDL interface and click **OK**.
  - c. Right-click again on the LaGaude component. Select **Generate Implementation** and choose default package or `sca.component.java.impl` if it is shown.
  - d. The implementation java source code editor opens. Add the following import statements to the code right after the package declaration:
    - `import java.util.ArrayList`
    - `import java.util.List`
    - `import com.ibm.websphere.sca.sdo.DataFactory`
  - e. Search for the `import_data` method and replace the implementation with the code shown in Example 10-5.

*Example 10-5 Java implementation: WS binding*

---

```
/*
 * Instantiation of the response message
 */
DataFactory dataFactory = DataFactory.INSTANCE;
DataObject processinginformationparameter = dataFactory.create(
    "urn:omg.org/plm20/services/parameter",
    "Processing_informations_parameter");
DataObject processinginformation = dataFactory.create(
    "urn:omg.org/plm20/computational/core",
    "PLM_processing_information");
processinginformation.setString("Message",
    "Response from La Gaude organization!");
List responselist = new ArrayList();
responselist.add(processinginformation);
processinginformationparameter.setList("return", responselist);
System.out.println("La Gaude organization services");
return processinginformationparameter;
```

---

**About this procedure:** Here we simulate the message response we get when our the Web services provided by the La Gaude mechanical design site is triggered by our request. We are also printing out in the WPS test environment console log a message indication that we have reached this particular service.

- f. Press Ctrl-S to save your changes.

5. In the Endpoints Assembly Diagram panel, wire the PLM\_general\_connectionExport1 to the LaGaude component.
6. Create another export PLM\_general\_connectionExport2 and another java component Boeblingen as above. Remember to add the three imports:
  - import java.util.ArrayList
  - import java.util.List
  - import com.ibm.websphere.sca.sdo.DataFactory
7. Replace the import\_data implementation with the code shown in Example 10-6.

*Example 10-6 Java implementation: WS Binding 2*

---

```

/*
 * Instantiation of the response message
 */
DataFactory dataFactory = DataFactory.INSTANCE;
DataObject processinginformationparameter = dataFactory.create(
    "urn:omg.org/plm20/services/parameter",
    "Processing_informations_parameter");
DataObject processinginformation = dataFactory.create(
    "urn:omg.org/plm20/computational/core",
    "PLM_processing_information");
processinginformation.setString("Message",
    "Response from Boeblingen organization!");
List responselist = new ArrayList();
responselist.add(processinginformation);
processinginformationparameter.setList("return", responselist);
System.out.println("Boeblingen organization services");
return processinginformationparameter;

```

---

**About this procedure:** Here we simulate the message response we get when our the Web services provided by the La Gaude mechanical design site is triggered by our request. We are also printing out in the WPS test environment console log a message indication that we have reached this particular service.

- g. Press Ctrl-S to save your changes.
8. In the Endpoints Assembly Diagram panel, wire the PLM\_general\_connectionExport2 to the Boeblingen java component.
9. Press Ctrl-S to save the module.

10. In the Business Integration View, expand **PLMServices20** → **Web Service Ports**. There are two new ports created:

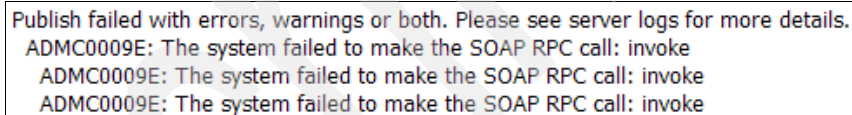
- PLM\_general\_connectionExport1\_PLM\_general\_connectionHttpPort
- PLM\_general\_connectionExport1\_PLM\_general\_connectionHttpPort

Double-click each one of these ports to open the WSDL editor. Go to Source view and locate the following soap address lines:

```
<soap:address location
="http://localhost:9080/Endpoints/sca/PLM_general_connectionExport1"
/>
<soap:address location
="http://localhost:9080/Endpoints/sca/PLM_general_connectionExport2"
/>
```

Replace the defaulted port 9080 with the WC\_defaulthost port that you are using for your WPS.

11. Press Ctrl-s to save the Endpoints module with your changes. Close all open editors and clean all projects in your workspace.
12. Add the PLMMediationWSBinding project to your WPS. and then click **Publish**. If the server State still shows Republish due to SOAP Invoke errors like those shown in Figure 10-41, right-click the server again and select **Publish**.



Publish failed with errors, warnings or both. Please see server logs for more details.  
ADMC0009E: The system failed to make the SOAP RPC call: invoke  
ADMC0009E: The system failed to make the SOAP RPC call: invoke  
ADMC0009E: The system failed to make the SOAP RPC call: invoke

Figure 10-41 Publishing errors

13. When publishing is complete, expand your WPS to see the modules deployed. Right-click **EndpointsApp** and select **Restart EndpointsApp**. This is to ensure the new Web services are deployed to the server.
14. Open a browser and test the Web services and make sure you get a response similar to that shown in Figure 10-42.

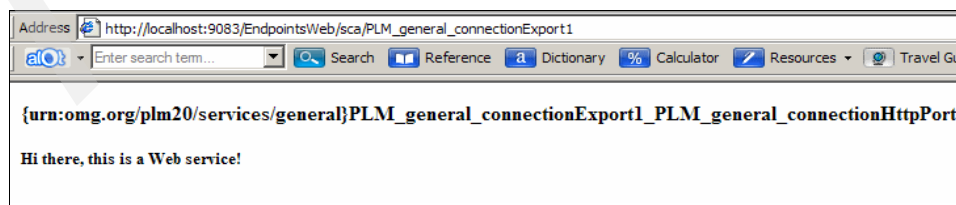


Figure 10-42 Testing the Web service

If you do not get the page displaying “Hi there, this is a Web service!”, perform the following steps to correct the issue:

- a. Right-click **Endpoints**. Click **Open Deployment Editor**.
- b. In the Module Deployment Editor, click the Exports tab
- c. In the Web Service Exports panel, highlight one of the exports. For instance, PLM\_general\_connectionExport1. Expand the **URL Mappings** and check Include default mapping if it is not checked.
- d. Perform steps a–c for the second Web Service export
- e. Press Ctrl-S to save the changes → answer **No** on the Inconsistent Files prompt to update the editor as shown in Figure 10-43.

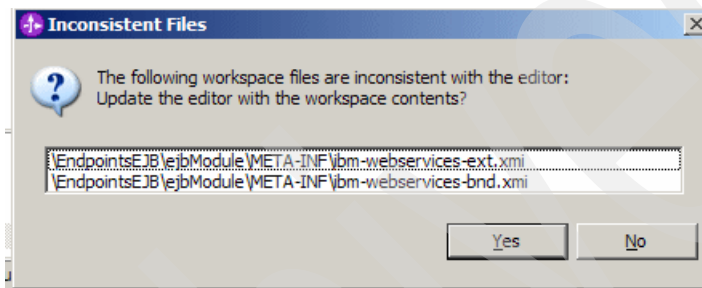


Figure 10-43 Inconsistent Files prompt

### Registering Web services into WSRR

1. Go to a browser and open your WSRR administration console.
2. Go to **Administrator or GP Administrator** (if you imported the Governance Profile into WSRR when you installed it) Perspective, expand **Service Metadata** → **WSDL** → **Ports**. Check if you have the WSDL ports already registered as shown in Figure 10-44 on page 459. Pay special attention to the following two ports:
  - PLM\_general\_connectionExport1\_PLM\_general\_connectionHttpPort
  - PLM\_general\_connectionExport2\_PLM\_general\_connectionHtpPort



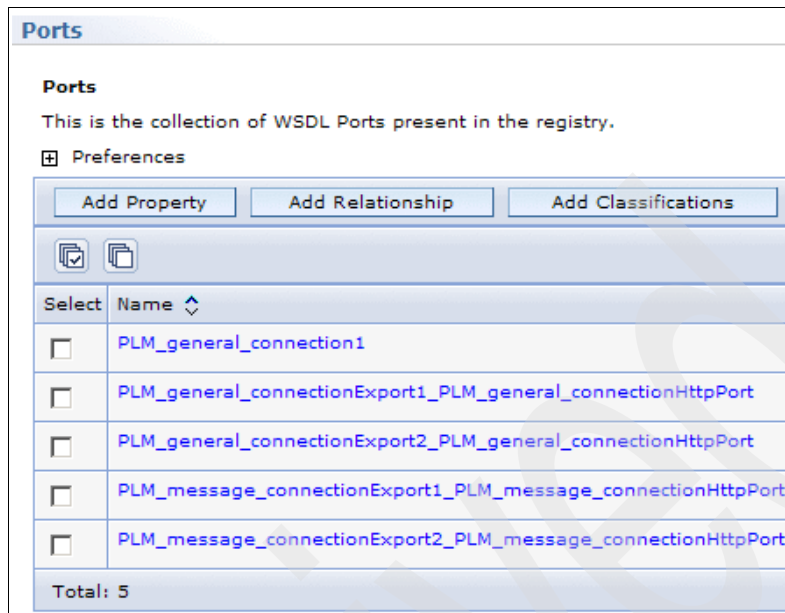


Figure 10-44 Registered WSDL ports

3. If you do not see the PLM\_general\_connection WSDL ports, perform the following steps:
  - a. Expand **Service Documents** and click **Load Documents**
  - b. Click Browse and retrieve the first Web Service WSDL, Endpoints\_PLM\_general\_connectionExport1.wsdl, from your WID project path. For instance, C:\Documents and Settings\Administrator\IBM\wid6.1\sa7593wsrr\PLMServices20
  - c. Retrieve the second Web Service, WSDL Endpoints\_PLM\_general\_connectionExport2.wsdl, and load it as well.
  - d. After loading, the WSDL Documents panel shows the two new WSDLs.
4. Expand **Service Metadata** → **WSDL** and click **Ports**. You must see the two new Ports:
  - PLM\_general\_connectionExport1\_PLM\_general\_connectionHttpPort
  - PLM\_general\_connectionExport2\_PLM\_general\_connectionHttpPort
5. Click **Properties of** **PLM\_general\_connectionExport1\_PLM\_general\_connectionHttpPort**.

6. Click **New** to add a property. In the General Properties panel, use the following Name/Value pairs:
  - Name: organization
  - Value: Boeblingen
- e. Click **OK**.
- f. Add another property using the following Name/Value pair:
  - Name: application\_domain
  - Value: Mechanical design
7. Perform step 5 and 6 for EndpointsPLM\_general\_connectionExport2\_PLM\_general\_connectionHttpPort, but use the following values:
  - organization: La Gaudé
  - application\_domain: Mechanical design

You have annotated your Web services in WSRR with the User Properties defined in your Endpoint Lookup Primitive in "Configure the WSRRLookup primitive." List number (b) on page 451.

**Note:** You may also use the customized classifications that you imported into WSRR (refer to "Customizing WSRR classifications" on page 416). You then use the Classifications entry panel in the Advanced property of your ELP. You can refer to the WSRR Information Center for more details:

[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.medprim.doc/ref/rwesb\\_EndpointLookupmediationprimitive.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.medprim.doc/ref/rwesb_EndpointLookupmediationprimitive.html)

### ***Testing the dynamic service selection use case***

We use the Integrated Test Client to test our second use case. Typically, our request parameters, Organization\_name and Application\_domain, are array elements in the input PLM\_container. Again, due to the ITC problem of recognizing the derived types of our PLM\_core\_container super set (refer to, we use a work around. We input our parameters as properties of the PLM\_core\_container.

1. In the your WID workspace, restart your WPS.
2. In the Business Integration View, expand the PLMMediationWSBinding view. Double-click **Assembly Diagram** to open the editor.
3. Right-click the **RequestBuilder** component. Click **Test Component** to open the Integrated Test Client (ITC) panels.
4. In the Detailed Properties panel, select **Operation: import\_data**.

5. Enter dummy data, like AAA, for the UID parameter.
6. Right-click **properties**. Click **Add Elements**. Type 2 to add two Name/Value property pairs:
  - Two User Properties set in our ELP for dynamic service selection
    - Name = Application\_domain
    - Value = Mechanical design
    - and
    - Name = Organization\_name
    - Value = Boeblingen

**Important:** In these Detailed Properties, we are using the Base Type of our input PLM\_container and setting Name/Value pairs of properties. This is to get around the limitation of the Integrated Test Client, which fails to recognize derived types that are element arrays. The desired way to pass the parameters would be to use Derived Types and enter values in the PLM\_container elements: Approval\_status/Status\_name, Application\_context/Application\_domain, Organization/Organization\_name

The Detailed Properties looks similar to that shown in Figure 10-45.

**▼ Detailed Properties**

Configuration: Default Module Test

Module: PLMMediationWSBinding

Component: RequestBuilder

Interface: PLM\_general\_connection

Operation: import\_data

☐ Invoke export using binding

Initial request parameters

Name	Type	Value
uid	ID	✓ AAA
version_i	string	✓ 2.0
lang	language	✓
properties	PLM_property[]	60
property	PLM_property	✓
Name	string	✓ Application_domain
Value	string	✓ Mechanical design
property	PLM_property	✓
Name	string	✓ Organization_name
Value	string	✓ Boeblingen

Figure 10-45 ITC Detailed Properties

- Click the green arrow to start the test. A successful test displays the Events panel as shown in Figure 10-46. The Console panel shows the message “Boeblingen organization services.”

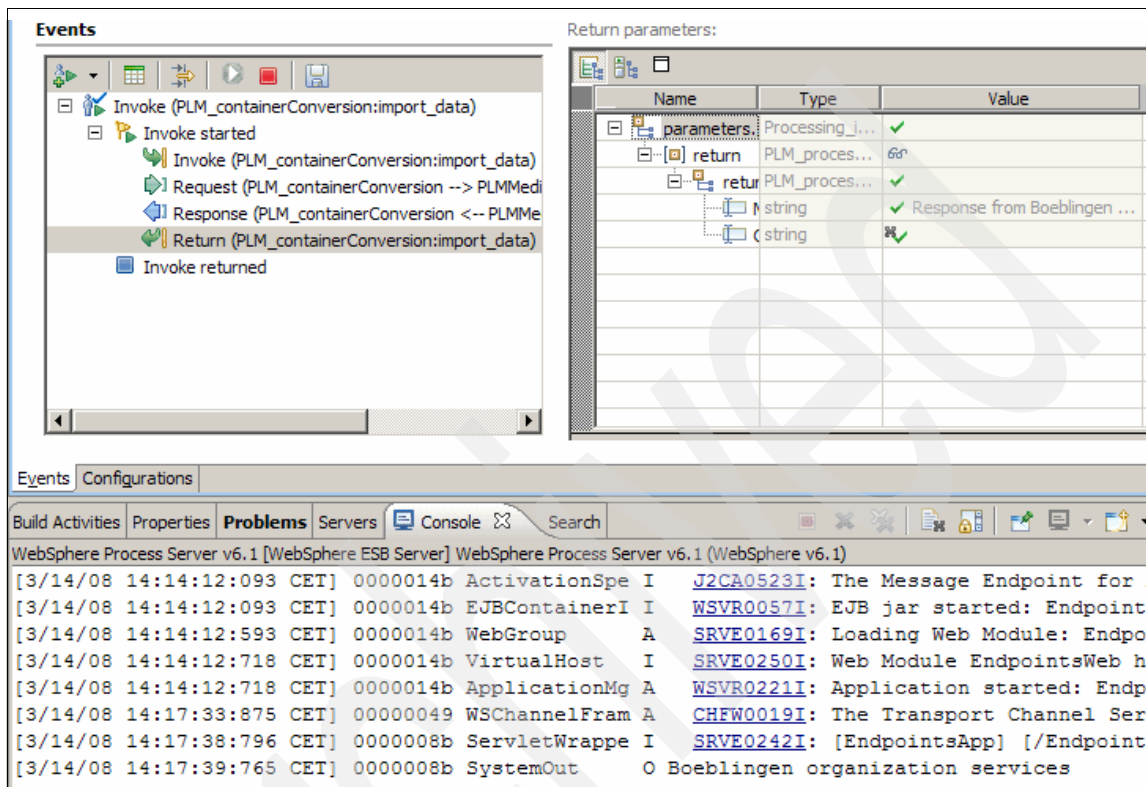


Figure 10-46 End-to-end successful test

At this point, you have successfully taken WSRR and integrated it into a PLM-based solution to manage endpoints and enhance the dynamic routing and selection capability of the WESB.

**Note:** From the Appendix A, “Additional material” on page 465, you can get the complete sample solution code in the sub folder PLMWithWSRR\_Solution of the main folder PLMWithWSRR inside the PLMWithWSRR.zip file.



## Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

### Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247593>

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, SG24-7593-00.

## Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
Redbook247593_lab.zip	Residency lab files
Redbook247593.zip	Residency lab files
PLMWithWSRR.zip	Residency lab files

## How to use the Web material

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material file into this folder.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## Other publications

These publications are also relevant as further information sources:

- ▶ *STEP Application Handbook* ISO 10303, 2006. Version 3, SCRA (South Carolina Research Authority), SC. Free download from <http://isg-scra.org/STEP/STEPHandbook.html>
- ▶ ISO 10303-11: 2004, *Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual*, International Organization for Standardization, Geneva, Switzerland.
- ▶ *Usage Guide for the STEP PDM Schema*, 2002, V1.2, Release 4.3, PDM Implementor Forum. Free download from [http://www.pdm-if.org/pdm\\_schema/pdmug\\_release4\\_3.zip](http://www.pdm-if.org/pdm_schema/pdmug_release4_3.zip)
- ▶ Peak, R., Lubell, J., Srinivasan, V., Waterbury, S., 2004, "STEP, XML, and UML: Complementary Technologies", *ASME Journal of Computing and Information Science in Engineering*, Vol. 4, pp. 379-390.
- ▶ ISO/TS 10303-28: 2003, *Industrial automation systems and integration -- Product data representation and exchange -- Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas*, International Organization for Standardization, Geneva, Switzerland.
- ▶ ISO/TS 10303-25: 2005, *Industrial automation systems and integration -- Product data representation and exchange -- Part 25: Implementation methods: EXPRESS to XMI binding*, International Organization for Standardization, Geneva, Switzerland.
- ▶ *Model Driven Architecture Guide*, Version 1.0.1, 2003, Object Management Group.
- ▶ *OMG PLM Services* 1.0.1, 2006, Object Management Group.
- ▶ *PDM Enablers* Version 1.3, 2000, Object Management Group.
- ▶ *PDTnet Implementation Guide* Version 1.6, 2003, PROSTEP AG.

- ▶ Web Services Business Process Execution Language Version 2.0, 2007, OASIS.
- ▶ Palmisano, S.J., 2006, "The Globally Integrated Enterprise", *Foreign Affairs*, Vol. 85, No. 3, pp. 127-136.
- ▶ Srinivasan, V., 2008, "A Comprehensive Business Approach to Product Realization Using Service Oriented Architecture", *Product Realization: A Comprehensive Approach*, M. Tomovic and S. Wang, eds., Springer.
- ▶ Srinivasan, V., 2008, "An Integration Framework for Product Lifecycle Management", *Computer-Aided Design*, under review.
- ▶ Kemmerer, S.J. (Editor), 1999, *STEP: The Grand Experience*, NIST Special Publication 939, Gaithersburg, MD.
- ▶ Srinivasan, V., 2005, "Open Standards for Product Lifecycle Management", *International Conference on Product Lifecycle Management*, Lyon, France.
- ▶ Fenves, S.J., Sriram, R.D., Subrahmanian, E., and Rachuri, S., 2005, "Product Information Exchange: Practices and Standards", *Journal of Computing and Information Science in Engineering*, Transactions of the ASME, Vol. 5, pp. 238-246.
- ▶ Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S.J., Foufou, S., and Sriram, R.D., 2008, "Information Sharing and Exchange in the Context of Product Lifecycle Management: Role of Standards", to appear in *Computer Aided-Design*.
- ▶ Srinivasan, V. "Standardizing the specification, verification, and exchange of product geometry: Research, status and trends", *Computer-Aided Design* 2007;
- ▶ Srinivasan V. "A geometrical product specification language based on a classification of symmetry groups", *Computer-Aided Design* 1999; 31 (11): 659-68.
- ▶ ASME Y14.5M-1994. *Dimensioning and tolerancing*. New York (USA): The American Society of Mechanical Engineers; 1995.
- ▶ ASME Y14.5.1M-1994. *Mathematical definition of dimensioning and tolerancing principles*. New York (USA): The American Society of Mechanical Engineers; 1995.
- ▶ ISO 1101. *Geometrical Product Specifications (GPS) -- Geometrical tolerancing -- Tolerances of form, orientation, location and run-out*. 2nd ed. Geneva (Switzerland): International Organization for Standardization; 2004.
- ▶ ISO/TS 17450-1. *Geometrical product specifications (GPS) -- General concepts -- Part 1: Model for geometrical specification and verification*. 1st ed. Geneva (Switzerland): International Organization for Standardization; 2005.

- ▶ ASME Y14.41. *Digital product definition data practices*. New York (USA): The American Society of Mechanical Engineers; 2003.
- ▶ ISO 16792. *Technical product documentation -- Digital product definition data practices*. Geneva (Switzerland): International Organization for Standardization; 2006.
- ▶ ISO 19005-1: 2005. *Document management - electronic document file format for long-term preservation - part 1: Use of PDF 1.4 (PDF/A-1)*. Geneva (Switzerland): International Organization for Standardization; 2006.
- ▶ Srinivasan, V., Lammer, L., and Vettermann, S., 2008, "On Architecting and Implementing a Product Information Sharing Service", ASME Journal of Computing and Information Science in Engineering, March, Vol. 8.
- ▶ VDA Recommendation for Engineering Change Management Part 1: *Engineering Change request (ECR)*, VDA recommendation 4965, VDA 2006.

## Online resources

These Web sites are also relevant as further information sources:

- ▶ OMG PLM Services  
<http://www.prostep.org/en/standards/plmservices>
- ▶ American Productivity and Quality Council (APQC) organization  
<http://www.APQC.org>
- ▶ UML 2.0 Profile for Software Services  
[http://www.ibm.com/developerworks/rational/library/05/419\\_soa/](http://www.ibm.com/developerworks/rational/library/05/419_soa/)
- ▶ WebSphere Process Server Infocenter  
[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome\\_top\\_wps.htm](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm)
- ▶ Technical resources for the WebSphere software platform  
<http://www.ibm.com/developerworks/websphere/>
- ▶ Installing WebSphere Integration Developer  
[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.install.doc/topics/c\\_inintro.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.install.doc/topics/c_inintro.html)
- ▶ ProSTEP iViP Project Groups  
<http://www.prostep.org/en/projektgruppen/pdm-if/plmservices.htm>
- ▶ PLM Services 2.0 WSDL and XML schema files  
<http://schema.omg.org/specs/PLM/2.0/>

- ▶ ProSTEP iViP - VDA 4965 ECR/ECM files - downloads  
<http://www.prostep.org/en/standards/doku/standards/>
- ▶ IBM WebSphere Service Registry and Repository Version 6.1 information center  
<http://publib.boulder.ibm.com/infocenter/sr/v6r1/index.jsp>
- ▶ Installing a stand-alone WSRR  
[http://publib.boulder.ibm.com/infocenter/sr/v6r1/index.jsp?topic=/com.ibm.sr.doc/twsr\\_installn02\\_standalone.html](http://publib.boulder.ibm.com/infocenter/sr/v6r1/index.jsp?topic=/com.ibm.sr.doc/twsr_installn02_standalone.html)
- ▶ IBM WebSphere Service Registry and Repository (WSRR) Overview  
[http://publib.boulder.ibm.com/infocenter/sr/v6r0/index.jsp?topic=/com.ibm.sr.doc/cwsr\\_overview\\_overview.html](http://publib.boulder.ibm.com/infocenter/sr/v6r0/index.jsp?topic=/com.ibm.sr.doc/cwsr_overview_overview.html)
- ▶ Connection properties  
[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb602.doc/sibxwsrrresources/RegistryWSSConnection\\_DetailForm.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb602.doc/sibxwsrrresources/RegistryWSSConnection_DetailForm.html)
- ▶ Creating customized classification systems with WebSphere Service Registry and Repository V6.0.0.1 and higher  
[http://www-128.ibm.com/developerworks/websphere/library/techarticles/0703\\_shore/0703\\_shore.html](http://www-128.ibm.com/developerworks/websphere/library/techarticles/0703_shore/0703_shore.html)
- ▶ Business Process Management Samples & Tutorials - Version 6.1  
<http://publib.boulder.ibm.com/bpcsamp/index.html>
- ▶ Manage service availability dynamically using WebSphere Enterprise Service Bus and WebSphere Service Registry and Repository V6.1  
[http://www.ibm.com/developerworks/websphere/library/techarticles/0804\\_perepa/0804\\_perepa.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0804_perepa/0804_perepa.html)
- ▶ Endpoint Lookup mediation primitive  
[http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.medprim.doc/ref/rwesb\\_EndpointLookupmediationprimitive.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.medprim.doc/ref/rwesb_EndpointLookupmediationprimitive.html)

## 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](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)



# Index

## Numerics

3D CAD 66  
model 76

## A

ABAP exceptions 336  
access control 164  
action 261  
    data 226–227  
    parameter 224  
addition 76  
administration client 228  
administrative information 37  
Alias\_identification 78  
American Productivity and Quality Council (APQC) 108  
American Society of Mechanical Engineers (ASME) 34  
AP203 38  
AP212 38  
AP214 38, 69  
    and conformance classes 37  
    differences with EXPRESS 67  
    PdmSTEP module 49  
    PDTnet 51  
AP214 AIM 71  
AP214 ARM 69  
AP232 38  
Application Integration Layer 18  
application integration problems 15  
Application Interpreted Model (AIM) 67  
application programming interface (API) 60  
Application Protocol (AP) 37  
application silos 7  
application-level service 132  
applications 4–5  
ApplicationServerHost 291  
approval 79  
ApproveECR 268  
ApproveInitialECR 268  
APQC Process Classification Framework 131  
arbitrary properties 226  
ASCII character-based syntax 43

assembly structure  
    UML model 57  
asset analysis 129  
associated nterface 224  
authoring tools 5  
authorization 78  
Automotive Industry Component Business Model (CBM) 416  
availability analyst 149

## B

BAPI modules 355  
BAPI return structure 356  
BAPI/RFC 271  
Batch\_query 89  
bill of material (BOM) 10, 14, 28, 37, 65, 258, 272, 324  
    interface 295  
    Synchronization service 136  
BillOfMaterialInterface 292  
BM WebSphere Process Server 178  
BOM (bill of material) 397  
BPC Explorer 268, 396  
BPELprocess 387  
business analysis and modeling 128  
business analyst 148  
business component 115  
business component map 115  
business integration perspective of WID 419  
business logic 224–225, 231  
business management 161  
business models 164  
business object documents (BODs) 44, 61  
business object editor 189  
business objects 37  
    attachment.xsd 292  
    BOM.xsd 292  
    BOMBG.xsd 292  
    BOMItem.xsd 292  
    ComponentLink.xsd 292  
    Document.xsd 292  
    DocumentBG.xsd 292  
    ECN.xsd 292

- ECNBG.xsd 292
  - Effectivity.xsd 292
  - Message.xsd 292
  - MultiBOMs.xsd 292
  - MultiDocuments.xsd 292
  - MultiParts.xsd 292
  - Part.xsd 292
  - PartBG.xsd 292
  - Properties.xsd 292
  - Result.xsd 292
  - ResultBG.xsd 292
  - StandardFaultType.xsd 292
  - business process
    - logic 10
    - modeling 131
    - module 440
    - selection mediation logic flow 429
  - Business Process Choreographer Explorer 395
  - Business Process Execution Language (BPEL) 21–22, 25, 117, 406
  - business process management (BPM) 15, 22, 103, 106, 122, 124, 419
  - business process monitoring 22
  - business profile 160, 164
  - business relationship director 148
  - business rule interactions 123
  - business rules 365, 382
  - business service 132
  - business service champion 148, 153
  - business service governance 23–24
  - business service governance problems 16
  - business transformation 6
- C**
- CAD 12, 74
    - model 69
  - CGM 77
  - change business logic 382
  - change control board (CCB) 257, 264
    - group 390
  - change manager 149
  - classification 77
    - systems 415
  - client 291
  - closePLMSession 193
  - clothes 3
  - collaboration
    - layer 10
    - tools 11
  - common neutral integration language 122
  - Common Object Request Broker Architecture (CORBA) 45
  - company business processes 4
  - competition 4
  - compliance auditor 149
  - component business model (CBM) 115, 126, 129
  - component specification 137
  - component with no implementation type 209
  - computation independent model (CIM) 46, 62
  - computational model 65
  - computational viewpoint 80
  - computer languages 45
  - computer-aided design (CAD) 7, 31, 33
  - computer-aided engineering (CAE) 7, 31, 33
  - computer-aided manufacturing (CAM) 31, 33
  - config.properties 203
  - configuration data 228
  - configuration librarian 148
  - Configuration Management II (CMII) 24
  - configuration\_management 79
  - conformance classes (CC) 37
  - conformance points 68, 85
  - connection factory 282
  - copy 76
  - copyBO 191
  - CORBA interfaces 48
  - create a document record 343
  - create conditional mappings 353
  - create operation 317
  - createInvalidSessionIdException 192
  - createPLMAuthException 191
  - createPLMException 191
  - cross-functionality 104
  - CRUD functions
    - provided by PIM 57
  - custom ECR process 365
  - customer 104
  - customer relationship management (CRM) 33
  - customer resource management (CRM) 7
- D**
- Dassault System Matrix One 132
  - data 4
    - integration 15, 20
    - modeling language 43
  - decomposition 76–77



- defaultUser 226–227
- DefaultUser flag. 225
- definability 104
- define default values 353
- define modification parameters 281
- define revision levels for documents 278
- define revision levels for materials 277
- define statuses of change master records 280
- dependencies 201
- deployment phase of services 408
- derivation 73, 76
- deserializeBO 191
- Design Chain Operations Reference (DCOR) 108
- design review composite service 136
- design review process 135
- Developer Works 178
- development tools specialist 148
- direct exposure 132
- document 273
  - interface 296
  - mediation module 343
- document type DES 275
- DocumentSystemInterface 292
- domain
  - borders 70
  - decomposition 129–130
  - model 46
- DS Enovia LCA 123
- DUNS number 395
- dynamic BPEL selection 435
- dynamic business process (BPEL) selection 424
- dynamic endpoint selection capability of WESB 434
- dynamic service invocation 404

**E**

- EAR file 444
- ECM business process 24
- ECM queries conformance point 68
- ECM\_coordinator (Original Equipment Manufacturer) 406
- ECM\_participant (Supplier) 406
- ECR
  - coordinator (C) 258
  - requestor 266
  - requestor (P) 258
  - web client 270, 394
- ECR\_context 261
- ECR\_header 97
- ECR\_id 97
- ECRProcess business process 260
- ECRWebClient 270
- education and training 116
- electronics design automation (EDA) 7
- embeddedness 104
- empty action 382
- endpoint attributes 227
- endpoint data type 227
- endpoints module 439
- engineering change issue 49
- engineering change management (ECM) 14, 18, 24, 29, 33, 65, 122, 293, 405
  - support by PLM Services 2.0 60
  - use cases support 55
- engineering change notice (ECN) 28, 49
- engineering change number 271, 298, 365
- engineering change order 49
- engineering change process 25, 270
- engineering change request (ECR) 25, 49
- engineering objects 37
- EngineeringChangeManagementInterface 292
- enterprise architecture (EA) 112
- enterprise metadata discovery (EMD) 141
- enterprise PDM
  - adapter 141
  - application 143
  - component 140
  - services 143
  - specific API 140
  - specific service 141
- enterprise resource planning (ERP) 7, 28, 33
- enterprise service bus (ESB) 9, 18, 123
- enterpriseinformation system 19
- EPCProcessEndpoint 444
- EPDProcessEndpoint 444
- ERP (enterprise resource planning) 9
- ESB implementation 404
- Eurostep
  - development of PLCS Web Services 61
- export 426
- export specific settings 210
- EXPRESS
  - differences with AP214 67
- EXPRESS model 68
- EXPRESS schema language 40
- EXPRESS-X mapping 67
- eXtensible Markup Language (XML) 35

## F

federated query business process 222  
FederatedRecursiveStructureQuery 267  
forEach 371  
full enterprise-level adoption 122  
functional area analysis 131  
functional dependencies 134

## G

generalConnectionUrl 197  
generic business objects (GBOs) 62  
generic conformance point 68  
generic queries conformance point 58, 86–87  
Geometric\_model\_relationship\_query 136  
get\_general\_connection and  
get\_message\_connection 193  
getConnectionProperties 191  
getPLMServicesType 192  
getPLMServicesType(DataObject sdo) 190  
globalization 4  
goal-service modeling 129  
governance 159  
governance profile 160  
governance profile business model 165

## H

hierarchy 73  
higher value BPM 124  
human tasks 365, 382

## I

independent software vendor (ISV) 61  
indirect exposure 132  
information exchange 32  
information services 124  
information sharing 32  
Information Technology Infrastructure Library (ITIL) 157  
initECR 260  
Initial Graphics Exchange Specification (IGES) 35  
InitializeECR process 259  
Integrated Test Client (ITC) 426, 445  
Integrated Test Component (ITC) 423  
integration framework for PLM 10  
integration specialist 148  
interaction and collaboration services 124  
interface description language (IDL)

modules 48  
internal service data object 190  
International Organization for Standardization (ISO) 34  
International Standards Organization (ISO) 18  
ISA-95/OAGIS SOA in Manufacturing 61  
ISO 10303-21 43, 67  
ISO 10303-239  
    product life cycle support 61  
ISO 10303-25 44  
ISO 10303-28 43  
ISO AP214 Standard for the Exchange of Product Data (STEP) 18, 26, 51, 108  
ISO STEP  
    community 44  
    OAGIS BODs 61  
IT architect 148  
IT Executive Steering Committee (ITESC) 146  
IT executive steering committee (ITESC) 147  
IT lifecycle processes (ITIL) 157  
IT portfolio manager 148  
ItemIteration 49  
ItemList (and ItemListImpl) class 190  
ItemMaster 49  
ItemRevision 49  
ITSOPLMLibrary 292

## J

J2EE 179  
Java 80, 178, 190  
    Connector API (JCO) 140  
    Connector Architecture (JCA) 18, 132  
    editor 208  
    interfaces 427  
    wrapper classes 177, 190  
JAX-RPC handler 177, 210  
JCA Adapter 139  
JMS export 219  
JSF-based web client 269

## K

key performance indicators (KPIs) 22, 29, 110, 114, 129

## L

language 291  
level 1, enterprise view 104

- level 2, business function view 105
- level 3, process 105
- level 4, sub-process 105
- level 5, activity 105
- level 6, work steps 105
- level 7, step scripts 105
- life cycles 164
- life cycle-stage 153
- line of business (LOB) 148

## M

- managing services 408
- manufacturing data management 14, 28
- manufacturing systems 4
- mapping 190
- mapping file 312
- marketplace demands 3
- markup languages 35
- match policy 432
- material master 272
  - interface 294
  - mediation module 312–313, 320
- MaterialMasterInterface 292
- mechanical design (M-CAD) engineering divisions 405
- mediation
  - flow 308, 355
  - flow diagram 321
  - module 298, 312
- mergers and acquisitions 4
- message queries conformance point 58, 87
- Message\_parameter instance 263
- messageConnectionUrl 197
- metadata 34
  - management 159
- metainformation 222
- migration 161
- model 117
- model driven architecture (MDA) 45
  - development of OMG PLM services 62
- model driven systems development 11
- model transformation 46
- modeling languages 35
- module 426
- monitor 118
- MP3 player 3
- MSC SimManager 123, 132
- Multi\_language\_support 80

- multi-disciplinary business processes 7

## N

- neutral object model 10
- new connection factory 214

## O

- OAGi 61
- OAGIS
  - BODs 61
- OAGIS BODs
  - compared to OM PLM Services specification 61
- OASIS (Organization for the Advancement of Structured Information Standards) 61
  - PLCS Web Services 61
- object management architecture (OMA) 45
- object management group (OMG) 18, 44
- object request broker (ORB) 45
- OMA framework 45
- OMG (object management group) 61
  - omg.org.plm20.util package 191
  - PDM Enablers 68
  - PLM Services 68, 133
  - PLM Services 2.0 20, 25, 66, 132, 136
  - PLM Services 2.0 UML model 142
  - standard specification 47
- OMG PDM Enablers
  - modules 49
- OMG PLM Services
  - case for adopting 61
  - derivation and architecture 54
  - queries that deal with PLM-related atomic tasks 59
  - timeline 53
- open application group integrationspecification (OAGIS) 18, 44
- open standards 9
  - formats 35
- OpenPDM 60, 142
- Operation 224
- order 104
- Organization for the Advancement of Structured Information Standards (OASIS) 44
- organizational structuring 116
- original equipment makers (OEMs) 26

## P

- package explorer 202
- parameterization of business logic 222
- part numbers 37
- part\_identification package 72
- partners 5
- PDES Inc.
  - STEP PDM Schema 38
- PDIFLibrary 260
- PDM (product data management) 9, 12
  - enablers 47
  - schema 38
    - entities 41–42
    - units of functionality 39
    - versus OMG PDM Enablers 48
  - schema definition 67
  - system 218
  - systems 194, 222, 259
- PdmBaseline 49
- PdmChangeManagement 49
- PdmConfigurationManagement 49
- PdmDocumentManagement 49
- PdmEffectivity 49
- PdmFoundation 49
- PdmFramework 49
- PdmManufacturingImplementation 49
- PdmProductStructureDefinition 49
- PdmResponsibility 49
- PdmSTEP 49
- PdmViews 49
- PDTnet 47
  - conformance point 68
  - predefined queries 51
- PDTnet predefined queries
  - AliasIdentificationQuery 51
  - AlternativeSolutionQuery 51
  - ApprovalQuery 51
  - ApprovalRelationshipQuery 51
  - AssemblyStructureQuery 51
  - AssociatedDocumentQuery 51
  - AssociatedOrganisationQuery 51
  - ConnfigurationQuery 52
  - CreateProductStructure 52
  - DateAndTimeQuery 52
  - DocumentClassificationQuery 52
  - DocumentPropertyQuery 52
  - DocumentQuery 52
  - DocumentStructureQuery 52
  - DownloadDigitalFile 52
  - DownloadSetOfDigitalFiles 52
  - EffectivityAssignmentQuery 52
  - ExternalFileQuery 52
  - GetProductStructure 52
  - IntiateOfflineUpload 52
  - ItemClassificationQuery 52
  - ItemPropertyQuery 52
  - ItemQuery 52
  - ItemRelationshipQuery 52
  - ItemWhereUsed 52
  - OrganizationQuery 52
  - ProductStructureQuery 52
  - SearchInDesignSpace 52
  - StartNodeQuery 52
  - SubscriptionListAcknowledgeContent 53
  - SubscriptionListGetContent 53
  - SubscriptionListRemovalContent 53
  - SubscriptionListAddContent 53
  - UploadDigitalFile 53
  - UploadSetOfDigialFiles 53
- PDTnet project 92
- PDTnet queries 214
- PDTnet queries conformance point 58, 86, 92
- PIM (platform independent model) 62
- planes 3
- platform independent model (PIM) 46
- platform independent viewpoint 62
- platform specific model (PSM) 46, 59
- PLM (Product Lifecycle Management)
  - business processes 10
  - Container 2.0 420
  - context 80
  - disciplines
    - engineering change management 14
    - manufacturing data management 14
    - product data management 14
    - service data management 14
    - simulation data management 14
  - ERP interface 297
  - services 10
  - services services assets 360
  - services services client 198
  - services services client. 221
  - services services reference client 214
  - services services standard 45, 68
  - services services to SAP interface 297
  - services services V2.0 s 97
- PLM Services 1.0 47
- PLM Services 2.0 47, 177, 181, 214, 221, 223, 229

- clients 223
- compliant server 177
- connector/wrapper 217
- connectors 177
- container 420
- data model 190
- web service skeleton 217
- web services platform specific model (PSM) 183
- web services PSM 184
- PLM to SAP integration module 360
- PLM to SAP integration process 365
- PLM/PDM systems 223
- PLM\_connection 81
- PLM\_connection service 82
- PLM\_connection\_factory 200, 203
- PLM\_connection\_factory and 81
- PLM\_connection\_factory interface 193
- PLM\_connection\_factory service 82
- PLM\_container 261
- PLM\_context 84
- PLM\_context element 72
- PLM\_core\_container 83–84
- PLM\_core\_container element 72
- PLM\_general\_connection 82, 204
- PLM\_general\_connection interface 200, 231
- PLM\_general\_connection query 224
- PLM\_general\_connection service 83
- PLM\_message\_connection 84, 204, 231
- PLM\_message\_connection interface 200
- PLM\_message\_connection service 259
- PLM\_message\_query 84
- PLM\_object 91
- PLM\_object element 72
- PLM\_object\_factory 81
- PLM\_processing\_information objects 85
- PLM\_property 82
- PLM\_resource\_adapter 81
- PLM\_service service 82
- PLM-application-centric approach 122
- PLMConnectionFactory 198, 203
- PLMConnectionFactory component 206
- PLMERPInterface 292
- PLMGeneralConnection 203
- PLMGeneralConnection component 208
- PLMMesageConnection 203
- PLMMessageConnection 208
- PLMProcess module 444
- PLMQueryIF 224
- PLMS2Server assembly editor 204
- PLMS2Server deployment descriptor 210
- PLMS2Server PDM system 196
- PLMS2Server project. 202
- PLMS2Server SCA applications 212
- PLMS2Util Java project 201
- PLMS2Util package 196
- PLMServer(String configRB) 196
- PLMServerConnectionFactory.java 193–194
- PLMServerConnectionFactory.java class 193
- PLMServerInstance.java 202
- PLMServerQuery 198
- PLMservices data types 177
- PLMServices2WithSessionIDLib 183
- PLMServices2WithSessionIDLib library 201
- PLMServicesToSAP 292
- PLMServicesUtil class 191
- PLMServicesUtil.getPLMServicesType() 190
- PLMWithWSRRSCABinding component 426
- Policy-based logging 161
- Policy-based service selection 161
- polymorphic structure 136
- portable document format (PDF) 34
- process
  - changes 10
  - classification framework (PCF) 108
  - decomposition 131
  - definition 10
  - management 10
  - orchestration 11
- process analyst 22
- Process BOMs forEach node 387
- Process Documents forEach node 387
- Process\_operation\_definition 80
- Process\_planning 80
- processing dependencies 135
- product and process management business unit 122
- product data 34
- product data management (PDM) 7, 14, 26, 31, 33
- product family 153
- product information models
  - PIM support 56
- Product Life Cycle Support (PLCS) 61
- product lifecycle 29
- Product Lifecycle Management (PLM) 3, 31
- product metadata model 44
- productdata structure (PDS) 26
- Project Interchange 440

- propagation mechanism 225
- properties 78
- PROSTEP AG 10, 142
  - OpenPDM 60
- ProSTEP iViP 194
  - OAGIS BODs 61
  - PDTnet 51
  - PLM Services 2.0 181, 212
  - PLM Services reference client 259
  - PLM services reference client 177
  - PLM services standard 47
  - PLM services V2.0 reference client 181
  - STEP PDM Schema 38
- proxy queries conformance point 58, 87, 90
- PSM 62
- PTC Windchill 123, 132

## Q

- quality of service (QoS) 136
- query\_messages call 259
- QueryECRMessages process 259

## R

- Rational Software Architect 138, 142
- Recursive\_query 89
- Redbooks web site 471
  - contact us xv
- reference 77
- registry 159
- Relationship\_query 136
- release specialist 148
- removeExistingMsgs 259
- repeatable business tasks 9
- repository 159
- Request\_initial\_ECR 97
- Request\_initial\_ECR message 259
- RequestBuilder 449
- resource adapter 282
- response diagram 311
- retrieve operation 318
- review the implementation 194
- RFC module 336
- RMI/IIOP 140
- RoHS 156
- role-based access 11
- roles and perspectives 164
- RPC 83

## S

- SAP 123, 140
  - BAPI modules 270
  - ERP
    - customization 273
    - system 270, 365
  - IDES (Internet Demonstration and Evaluation System) 270
  - integrating mediation modules 360
  - integration scenario 291
  - reference IMG 274
  - return structure 323, 355
  - system 365, 397
- SAP JCA Adapter 132
- SCA (service component architecture) 177, 217–218
  - binding 291, 365
  - export 200, 231, 292, 360
  - Import 360
  - imports 270
  - layer 336
  - security and operations specialist 148
  - security of services 125
  - Selection-queries 93
  - sequence 73, 77
  - serializeBO 191
  - serialized PLM\_container 198
  - service
    - component specifications 138
    - connectivity 123
    - creation 123
    - design 123, 126
    - flow specification 136
    - identification 126, 128
    - lifecycle process 151
    - litmus test 133
    - model 136
    - model work product 128
    - portfolio 134
    - provider 141
    - realization 126
    - registrar 148
    - registration 125
    - registry and repository 128
    - specification 126, 133
    - specification task 137
    - tester 148
  - service access 125
  - service architect 148

- service component architecture (SCA) 178
- service data management 14, 28
- service data objects (SDO) 178
- service design 136
- service developer 148
- service exposure decisions 134
- service oriented integration (SOI) 179
- service proliferation syndrome 134
- service registry lookups 24
- ServiceExportHandler 211
- service-oriented architecture (SOA) 3, 32, 45, 122
- service-oriented architectures (SOA) 160
- service-oriented modeling and architecture (SOMA) 126
- set control data 279
- ships 3
- shoes 3
- simulateSAP flag 363
- simulation PDM (SimPDM) 27
- simulationdata management 14, 27
- snippet 366
- SO 10303
  - 25 68
- SOA (service-oriented architecture) 60
  - board 147
  - center of excellence (CoE) 147
  - chief architect 148
  - core team 147
  - data architect 148
  - director 148
  - integration developer 148
  - life-cycle 162
  - operations architect 148
  - preliminary deployment 122
  - project manager (PM) 148
  - realization of integrated PLM solutions 403
  - security and management 125
  - security architect 148
- SOA-based integration applications 179
- SOA-based object management group (OMG) 405
- SOAP header 177, 210
- SOAP/HTTP protocol 200
- SOA-related technology adoption 122
- software architects 128
- SOMA method 126
- specific conformance point 68
- specific queries conformance point 58, 86, 91
- Specific\_query 136
- STandard for the Exchange of Product data model (STEP) 35
- STEP 10303
  - 214 ARM model 67
- STEP AP214 66
  - and OMG PLM Services architecture 47
- STEP AP214 CC21 67
- STEP file 71
- sub-processes
  - design change 50
  - process design and procurement agreements 50
  - product business plan 50
  - product definition 50
  - product design 50
  - product marketing configuration and rules 50
  - product service methods 51
  - production changes 51
  - service distribution plan 51
  - strategic product plan 50
- SubscriptionManager 223–227
- SubscriptionManager framework 259
- SubscriptionManagerConfig.xml) 228
- supplied item 73
- suppliers 5
- Supply Chain Operation Reference (SCOR) 108
- system borders 70
- system data model 123
- SystemNumber 291
- systems engineering approach 6

**T**

- tabular view 115
- taxonomies 164
- technology borders 71
- three-dimensional part geometry 34
- tools 4
- transaction code 274
- transaction handling management 136
- TransferECRtoERPSytem 291, 360, 400
- TransferECRtoERPSytem BPEL process 292
- translation 77
- Trego.xml 199
- Trego.xml data set 194
- truck 3
- tune business performance 114
- two-dimensional graphics 34

## U

- UGS TeamCenter 123, 132
- UID 83
  - used by PLM Services Informational Model 72
- UID parameter 426
- UML (unified modeling language)
  - 2.0 Profile for Software Services 141
  - component 138, 140
  - diagrams 48
  - interface 140
  - model 43
- unified modeling language (UML) 35
- URL-mapping 211
- use cases 405
- useDefaultUser 226–227
- user property 434
- usersResourceBundle 197
- utility queries conformance point 58, 87–88

## V

- Value Chain Operations Reference (VCOR) 108
- value-adding 104
- variation-oriented analysis 131
- VDA 4965 standard 24
- VDA Engineering Change Management 27
- Verband der Automobilindustrie (VDA) 18, 24, 108
- VIVACE project
  - development of PLCS Web Services 61

## W

- W3C XPath specification 58
- Waste from Electrical and Electronic Equipment (WEEE) 156
- Web 178
- Web Ontology Language (OWL) 415
- web service 206
- web service export 211
- web services 66
- web services - Interoperability (WS-I) 61
- web services bindings module 435
- Web Services Business Process Execution Language (WS-BPEL) 178
- Web Services Description Language (WSDL) 59, 69
- Web Services for dynamic selection 454
- web UI views 164
- webservices 203
- WebSphere Adapter 282

- WebSphere Adapter for SAP software 270, 283–284, 291
- WebSphere Application Server V6 Network Deployment 179
- WebSphere Business Modeler 118
- WebSphere Business Monitor 119
- WebSphere Enterprise Service Bus 408
- WebSphere Enterprise Service Bus (WESB) 404
- WebSphere ESB 179
- WebSphere fabric 119
- WebSphere Integration Developer 118, 170, 178–179, 182, 212, 218, 221, 232, 268, 270, 387, 423, 426
  - V6.1 Information Center 432
- WebSphere Process Server 21, 118, 170, 177–179, 189–190, 212, 221, 282, 390, 395
  - administration console 434
  - Infocenter 178
  - V6.1 (WPS) 411
- WebSphere Service Registry and Repository (WSRR) 159, 164, 222, 405, 408, 411
- WebSphere Service Registry and Repository Handbook, SG24-7386. 414
- WebSphere user registry 194
- WEEE 156
- WESB/WPS configuration 414
- World Wide Web Consortium (W3C) 61
- wrap existing function 132
- wrapper classes 190
- write\_messages operation 259
- WS-Addressing compliant endpoint references 230
- WS-Addressing ServiceRefType 227
- WSADIE/WID Service of IBM 426
- WSDL (web services description language) 91
  - interface 363
  - interfaces 427
- WS-Policy based domains 150
- WSRR (WebSphere Service Registry and Repository)
  - classification system 414
  - definition 414
  - Information Center 460
  - installation pointers 412
- WSRR (WebSphereServiceRegistry and Repository) 458

## X

- X\_path 87



XML (eXtensible Markup Language)  
66, 71  
Metadata Interchange (XMI) 44  
model 43  
pathlanguage 86  
schema 59, 91  
support of ECM process 60  
XML Schema PSM  
information model 69  
xmlFile 197–198  
xmlFile parameter value 203  
XPath conformance point 68  
XPath queries conformance point 58, 87, 91  
XR SimpleQueryProcess 227





**Redbooks**

# SOA Approach to Enterprise Integration for Product Lifecycle Management

(1.0" spine)  
0.875" <-> 1.498"  
460 <-> 788 pages







# SOA Approach to Enterprise Integration for Product Lifecycle Management

**Strategic capability to integrate systems, processes, and data**

**Examples based on real-life customer situations**

**PLM Services 2.0 code samples running on WPS**

This Redbooks publication documents the Product Development Integration Framework (PDIF) architectural approach towards Enterprise Integration around the Product Lifecycle Management application space. The primary focus being the integration of Product Data Management (PDM) systems to other PDM systems and PDM systems to Enterprise Resource Planning (ERP) systems, mainly SAP.

The objective of this Redbooks publication is to document the lessons learned (best practices) and articulate the Service Oriented Architecture (SOA) and Integration Framework that the PDIF team has developed over numerous customer engagements.

This Redbooks publication is defined for an audience of Business Analysts, Software Engineers, IT Architects, and IT Specialists.

**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)