WebSphere software

IBM

WebSphere MQ Workflow Transition to WebSphere Process Server

- Transition concepts and planning
- Transition support and guidance
- Transition examples

Saida Davies
Aditya P Dutta
Marc Fasbinder
Kurt Fleckenstein
Michael Fox
David Kadlecek
Augusto Kiramoto
Dr Hans-Joachim Novak
Elke Painke
Bo Wang

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere MQ Workflow Transition to WebSphere Process Server

January 2009

SG24-7282-00

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

First Edition (January 2009)

This edition applies to:

Version	Release	Modification	Fix pack	Product name
6	1	0		WebSphere Process Server
6	1	0	0	WebSphere Integration Developer
6	1	0	0	WebSphere Application Server
3	6	0	4	WebSphere MQ Workflow
6	0	2	2	WebSphere MQ
8	2	7		DB2 Universal Database
6	0			Tivoli Directory Server
XP			Service pack 2	Microsoft Windows

Contents

Figures

Tables

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 about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	HACMP™	RACF®
CICS®	i5/OS®	Rational®
DB2 Universal Database™	IBM®	Redbooks®
DB2®	Informix®	Redbooks (logo) ®
developerWorks®	Lotus®	Tivoli®
FFST™	MQSeries®	WebSphere®
Focal Point™	Parallel Sysplex®	z/OS®
GPFS™	pSeries®	

The following terms are trademarks of other companies:

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

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

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

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

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

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

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

Preface

This IBM® Redbooks® publication guides the transition from your WebSphere® MQ Workflow 3.6 environment to WebSphere Process Server V6.1. It provides a conceptual overview of WebSphere MQ Workflow and WebSphere Process Server, describes new features provided by the WebSphere Process Server, and discusses benefits of a transition to the new environment.

This book discusses the various areas that must be considered in a transition project:

- Human interaction in business processes
- Integration of backend applications
- Clients and application programming interfaces
- Operational aspects
- Information about how to plan for all aspects of a transition project

This book discusses the various areas that must be considered in a transition project:

- The transition concepts available:
  - Options for converting a business process from WebSphere MQ Workflow to WebSphere Process Server
  - Considerations for long-running processes and for process instances
  - Information about how to map features of a WebSphere MQ Workflow process to features in WebSphere Process Server
- Human interaction in business processes:
  - A description of the staff architecture in WebSphere MQ Workflow, and of people assignment in WebSphere Process Server
  - Information about how to map the staff model and repository data, the authorization information, and the staff-related definitions of process models to concepts available in WebSphere Process Server
- Integration of backend applications: A description of user-defined program execution server implementations for backend integration in WebSphere MQ Workflow, and of backend integration concepts in WebSphere Process Server, as well as information about how to transition from one to the other.
- Clients and application programming interfaces: A description of how clients were implemented in WebSphere MQ Workflow and how the corresponding implementation is done in WebSphere Process Server.

- Operational aspects: Assessing the topology in place in WebSphere MQ Workflow and defining a corresponding topology in WebSphere Process Server, as well as best practices for target topology high availability, scalability, deployment of applications, and administration.
- Information about how to plan for all aspects of a transition project.

This book has two sections:

- Part 1, “Transition planning” on page 1, provides planning information required to assess the current environment, define the target environment, and plan for a transition from WebSphere MQ Workflow 3.6 to WebSphere Process Server V6.1.
- Part 2, “Transition techniques” on page 227, provides detailed information about transition techniques, areas for transition, tools available, artifacts involved, and best practices.

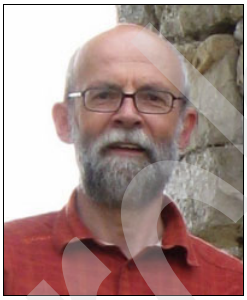
The team that wrote this redbook

This IBM Redbooks publication was produced by a team of specialists from around the world working for the International Technical Support Organization, at IBM Deutschland Entwicklung GmbH, Boeblingen, Germany.

Saida Davies is a Project Leader for the International Technical Support Organization (ITSO) and has extensive experience in information technology. She has published several Redbooks publications and Redpapers publication on WebSphere Business Integration, Web services, and WebSphere Service-Oriented Middleware using multiple platforms. Saida has experience in the architecture and design of WebSphere MQ solutions, extensive knowledge of the z/OS® operating system, and a detailed working knowledge of both IBM and independent software vendor operating system software. As a Senior IT Specialist, her responsibilities included the development of services for WebSphere MQ within the z/OS and Windows® platform. This covered the architecture, scope, design, project management, and implementation of the software on stand-alone systems or on systems in a Parallel Sysplex® environment. She has received Bravo Awards for her project contributions. Saida has a degree in computer studies and her background includes z/OS systems programming. Saida supports Women in Technology activities and contributes to and participates in their meetings.

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

Marc Fasbinder is a BPM Integration Solution Architect in the WebSphere brand for IBM in Southfield, Michigan. His responsibilities in technical sales include conducting Proof of Concepts, teaching customer Proof of Technology sessions, as well as working with customers in the business process management (BPM) space. He has worked at IBM for 18 years in the Industrial Sector Division, Global Services, and most recently the Software Group. He has specialized in WebSphere MQ Workflow since 2000, WebSphere Business Integration Server Foundation, WebSphere Process Server, as well as WebSphere Business Modeler and Monitor. Marc has broad experience across a variety of industries including manufacturing, banking, and insurance. He is the author of SupportPac WA0B: *WebSphere MQ Workflow - Best Practices Guide*. He has also written multiple articles for IBM WebSphere developerWorks®. Marc holds a degree in computer engineering and computer science from Oakland University in Rochester, Michigan.



Kurt Fleckenstein has 30 years of experience in the IT industry where he has held a wide variety of roles. During the last 13 years his role has been consulting, development, and testing in the business process management segment, specifically for WebSphere Process Choreographer, WebSphere MQ Workflow, and Flowmark. For the last seven years he has been working in the Business Process Management Competency Center in Boeblingen, Germany. He is an IBM Certified Solution Designer for WebSphere MQ Workflow and has contributed to the IBM WebSphere MQ Workflow SupportPacs WA04: WebSphere MQ Workflow - Java™ Generic API test and prototyping tool and WA09: WebSphere MQ Workflow - Generic C API test and prototyping tool. He also participated in WD01: Business Process Modeling with WebSphere MQ Workflow.

Michael Fox is a WebSphere Process Server consultant and member of the Business Process Management Competency Center within the WebSphere Business Process Solutions Development in Boeblingen, Germany. He has more than 20 years of experience in software development and worldwide technical customer consulting. Michael has been working on design and development of workflow management systems for the last 10 years, starting with FlowMark, moving to WebSphere MQSeries® Workflow, and now for the business process engine in WebSphere.

David Kadlecek is an IT architect in Global Technology Services in Prague. He has six years of design and development experience with J2EE and four years of experience with WebSphere application server and Workflow products. He has broad experience in application and enterprise architecture in a number of industries including banking, logistics, and telecommunication. His areas of expertise also include Web services, security, and performance tuning. He is currently finishing a Ph.D. in artificial intelligence at Czech Technical University in Prague.

Augusto Kiramoto is a Software Engineer and Project Leader at IntVision Solutions LTDA, IBM Business Partner in São Paulo, Brazil. He has been working on design and implementation of Enterprise Application Integration and Business Process Management solutions with WebSphere Business Integration products for five years. He holds a bachelor's degree in computer engineering and is finishing his master's degree in software engineering at Escola Politecnica of São Paulo University.

Dr Hans-Joachim Novak joined IBM right after he finished his Ph.D. work in artificial intelligence about combining image processing and natural language processing. He has held numerous positions at IBM ranging from developer, researcher, development manager, to consultant. He currently leads a team of WebSphere consultants specializing in doing Proof of Concepts for the WebSphere brand products. He has developed courses on WebSphere MQ Workflow as well as on WebSphere Process Server and has published numerous articles on a wide variety of topics.

Elke Painke is a WebSphere Process Server Consultant and member of the Business Process Management Competency Center within the WebSphere Business Process Solutions Development in Boeblingen, Germany. Her main focus is on supporting customers who are setting up complex WebSphere Process Server environments. Before joining the Competency Center, she worked on development projects for machine translation, z/OS user applications, and has six years of experience as a Technical Writer and Editor for various software projects at IBM. She holds a degree in applied language sciences from the Johannes-Gutenberg University in Mainz, Germany.

Bo Wang is a Software Support Professional for the WebSphere product family in IBM China. He has worked on WebSphere Business Integration products since he joined IBM in 2004. His main experience is system architecture design and system maintenance on the IBM WebSphere product family. Prior to this, he worked for an IBM business partner as a Software Engineer for three and a half years. He worked on application design and coding based on some IBM products during this period. He holds a bachelor's degree in computer science from Peking University.

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

Rolf Baeurle
WebSphere MQ Workflow Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Bernd Breier
WebSphere MQ Workflow Development and WSS Business Process Solutions Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany, for his contributions in the area of WebSphere Process Server clustering and runtime engine expertise

Michele Chilanti
Senior Consultant - WebSphere Services; Integration Architect: AIM. Service-oriented architecture (SOA) implementation, IBM Software Group, Application and Integration Middleware Software, IBM USA, for his contributions in the area of WebSphere Process Server high availability

Klaus Deinhart
Product Manager, WebSphere Business Process Integration (WebSphere Process Serverz, WebSphere Studio Application Developer Integration Edition, WebSphere Business Integration Server Foundation, and WebSphere MQ Workflow), IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Gabriel Dermler
WebSphere Process Choreographer Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Michael Friess
Business Process Choreographer Client Architect, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Erich Fussi
Business Process Choreographer Architect, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany, for his contributions in the area of WebSphere Process Server configuration, clustering, and topology details

Boris Feist
Lab-based Services, WebSphere Solution Center, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

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

Jens Grotrian
WebSphere MQ Workflow Development and WSS Business Process Solutions Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Manfred Haas
Worldwide WebSphere Process Integration Technical Sales: Team lead Business Process Management Competency Center, IBM Boeblingen, Germany

Volker Hoss
WebSphere MQ Workflow Development and WSS Business Process Solutions Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Ulrike Knapp
WebSphere MQ Workflow Development and WSS Business Process Solutions

Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Dieter Koenig
Senior Technical Staff Member, Business Process Choreographer, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Ines Lehmann
Consultant, WebSphere Solution Center, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Kurt Lind
Former WebSphere Process Server - Business Process Choreographer security and Human Task Manager architect, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Hubertus Meiwes
WebSphere Process Choreographer Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Richard Metzger
WebSphere MQ Workflow and WebSphere Process Server Performance Focal Point™, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Andreas Niemeyer
Release Manager WebSphere MQ Workflow, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Roland Oberle
WSS Business Process Solutions Test - Test Team Lead, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Gerhard Pfau
Senior Technical Staff Member, Business Process Choreographer, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Charles J Redlin
WebSphere Software Architect: AIM.Application Services/Infrastructure, IBM Software Group, Application and Integration Middleware Software for in-depth information about WebSphere Process Server topologies

Markus Reichart
BPC Observer Architect & Workflow Clients Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Rolf Schaefer
WebSphere MQ Workflow developer, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Stefan Ruettinger
WebSphere Integration Developer Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Andre Weiser
Software Developer, WebSphere MQ Workflow Development and WSS Business Process Solutions Development, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Andreas Wickenhaeuser
Certified WebSphere MQ Workflow System Planning and Modeling specialist, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Eva Wiese
Software planning for IBM WebSphere MQ Workflow, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Claudia Zentner
Senior Technical Staff Member, Business Process Choreographer, IBM Software Group, Application and Integration Middleware Software, IBM Boeblingen, Germany

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Part 1

Transition planning

This book is divided into two parts. Part 1, “Transition planning” on page 1, provides a product overview of WebSphere MQ Workflow Version 3.6 and WebSphere Process Server Version 6 and helps you assess the existing environment as well as the planned environment. The information provided in Part 2, “Transition techniques” on page 227, enables you to plan for a WebSphere MQ Workflow V3.6 transition to WebSphere Process Server V6.1.

The following chapters are discussed:

- Chapter 1, “Introduction to this book” on page 3, discusses the scope and the intended audience of the book.
- Chapter 2, “Products overview” on page 7, briefly describes the main concepts of WebSphere MQ Workflow V3.6 and WebSphere Process Server V6.1. This chapter also focuses on the new capabilities of WebSphere Process Server.
- Chapter 3, “Transition concepts” on page 29, describes the different approaches for converting an existing WebSphere MQ Workflow business process to the WebSphere Process Server environment. Examples are used to illustrate the value and limitations for each approach.
- Chapter 4, “Planning for human interaction in business processes” on page 57, describes the architecture of human interaction in WebSphere MQ

Workflow and WebSphere Process Server. It then discusses how a conversion of people-related information can be performed.

- Chapter 5, “Planning for back-end application integration” on page 89, describes how to integrate backend applications.
- Chapter 6, “Planning for clients based on application programming interfaces” on page 115, describes the clients available in WebSphere MQ Workflow as well as WebSphere Process Server. In addition, it discusses the application programming interfaces available for writing your own clients.
- Chapter 7, “Planning for operational aspects” on page 135, provides conceptual information about capacity planning for a WebSphere Process Server environment.
- Chapter 8, “Transition planning” on page 175, helps you define and plan for all aspects and phases of an overall transition project. Such a project is complex and will most likely differ from customer to customer. This section therefore does not provide an exact project plan or detailed information about sizing and topology, but rather a set of hints, considerations, and functional areas to assess when planning the transition of your environment.
- Chapter 9, “Business Process conversion” on page 187, describes three different approaches for converting the business process to the WebSphere Process Server environment. Examples are used to illustrate the value and limitations for each of the three approaches.

1

Introduction to this book

This chapter discusses the following:

- The scope of this book
- The intended audience
- Assumptions
- What is not covered in the book

1.1 The scope of this book

The aim of this IBM Redbooks publication is to provide guidance for a transition from WebSphere MQ Workflow V3.6 to WebSphere Process Server V6.1 to help assess and plan the transition options and identify transition issues.

The difference in programming model and programming constructs of WebSphere MQ Workflow and WebSphere Process Server in many cases does not cater to a fully automated transition. The first step in a transition project is therefore to assess the business processes currently in place in WebSphere MQ Workflow, as well as the infrastructure supporting them.

As a next step, some initial considerations may then need to be made when defining the target environment:

- Will the business processes be unchanged in the new environment, or are you planning on modifying them to cater to new or changed business needs?
- Will additional processes be implemented?
- Do you want to exploit WebSphere Process Server functionality previously not available in WebSphere MQ Workflow?

This book does not provide transition tasks on a step-by-step basis. Instead, it aims at providing you with the information required to evaluate the environment currently in place in WebSphere MQ Workflow, plan the target environment, and use the tools and information available to decide what can be transitioned in an automated fashion.

This book is divided into two parts:

- Part 1, “Transition planning” on page 1, provides a product overview of WebSphere MQ Workflow Version 3.6 and WebSphere Process Server Version 6, helps you assess the existing environment as well as the planned environment, and provides information about planning a WebSphere MQ Workflow V3.6 transition to WebSphere Process Server V6.1.
- Part 2, “Transition techniques” on page 227, provides detailed information required for a transition, the tools available, artifacts involved, and some best practices.

1.2 Intended audience

The intended audience of this book ranges from transition planners to those who actually perform the transition.

We recommend that planning experts read the information provided in Part 1, “Transition planning” on page 1, and technical experts responsible for performing the actual transition read Part 2, “Transition techniques” on page 227.

1.3 Assumptions

This book makes a number of assumptions. These are:

- You are familiar with WebSphere MQ Workflow V3.6 and the related products with the latest fixpacks:
  - WebSphere MQ V6.0.
  - DB2 Universal Database or Oracle®
- You have an understanding of the Eclipse and Java/J2EE platform.
- You have working knowledge of WebSphere Application Server V6.
- You have a ready environment with WebSphere Integration Developer and WebSphere Process Server installed and configured.

1.4 What is not covered in the book

This book focuses on information and tasks relating to the *transition* to WebSphere Process Server from WebSphere MQ Workflow V3.6.

This book does not cover:

- Details on the installation of software products, including WebSphere MQ Workflow, DB2 Universal Database, WebSphere Integration Developer, and WebSphere Process Server. Therefore, no step-by-step instructions are included.
- Installation of prerequisite and co-requisite software, such as WebSphere MQ and operating system updates.
- Instructions about the installation, configuration, or use of any previous version of WebSphere Process Server.
- Step-by-step transition instructions about how to transition a given business process. Instead, this book focusses on describing how business processes could be implemented in WebSphere MQ Workflow and what corresponding functionality could be used in WebSphere Process Server.

Install and use of software products

Refer to the installation and system requirements documentation for further details about WebSphere Process Server and WebSphere Process Server products at:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp

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

Refer to the following documentation for WebSphere MQ Workflow:

- *IBM WebSphere MQ Workflow: Concepts and Architecture,* GH12-6285
- *IBM WebSphere MQ Workflow: Administration Guide*, SH12-6289
- *IBM WebSphere MQ Workflow: Installation Guide*, SH12-6288
- *IBM WebSphere MQ Workflow: Getting Started with Runtime*, SH12-6287
- *IBM WebSphere MQ Workflow: Getting Started with Buildtime*, SH12-6286
- *ÌBM WebSphere MQ Workflow: Programming Guide,* SH12-6291
- *IBM WebSphere MQ Workflow for z/OS: Programming Guide*, SC33-7031

These guides are available at:

http://www.ibm.com/software/integration/wmqwf/library/index.html

2

Products overview

This chapter discusses the following:

- WebSphere MQ Workflow architecture
- WebSphere Process Server architecture

2.1 WebSphere MQ Workflow architecture

WebSphere MQ Workflow is an application built on top of WebSphere MQ. Therefore, it uses reliable messaging as provided through guaranteed once-and-only-once delivery of messages as the underlying infrastructure. It provides an easy way to manage complex and changing enterprise business processes. Using WebSphere MQ Workflow enterprises can design, develop, deploy, and modify business processes easily and quickly.

2.1.1 Architectural overview

This section describes the architecture of WebSphere MQ Workflow in detail. There are two main parts of the system:

- The modeling component called Buildtime
- The Runtime environment

In the Buildtime the user creates the business models that are later executed in the Runtime environment.

Figure 2-1 shows all the components of the WebSphere MQ Workflow system. Depicted are three main parts:

- In the lower part, all application programming interfaces (APIs) and the respective clients
- At the far right, the modeling component called WebSphere MQ Workflow Buildtime
- In the main part, the Runtime environment consisting of the different servers and the messaging and database layers

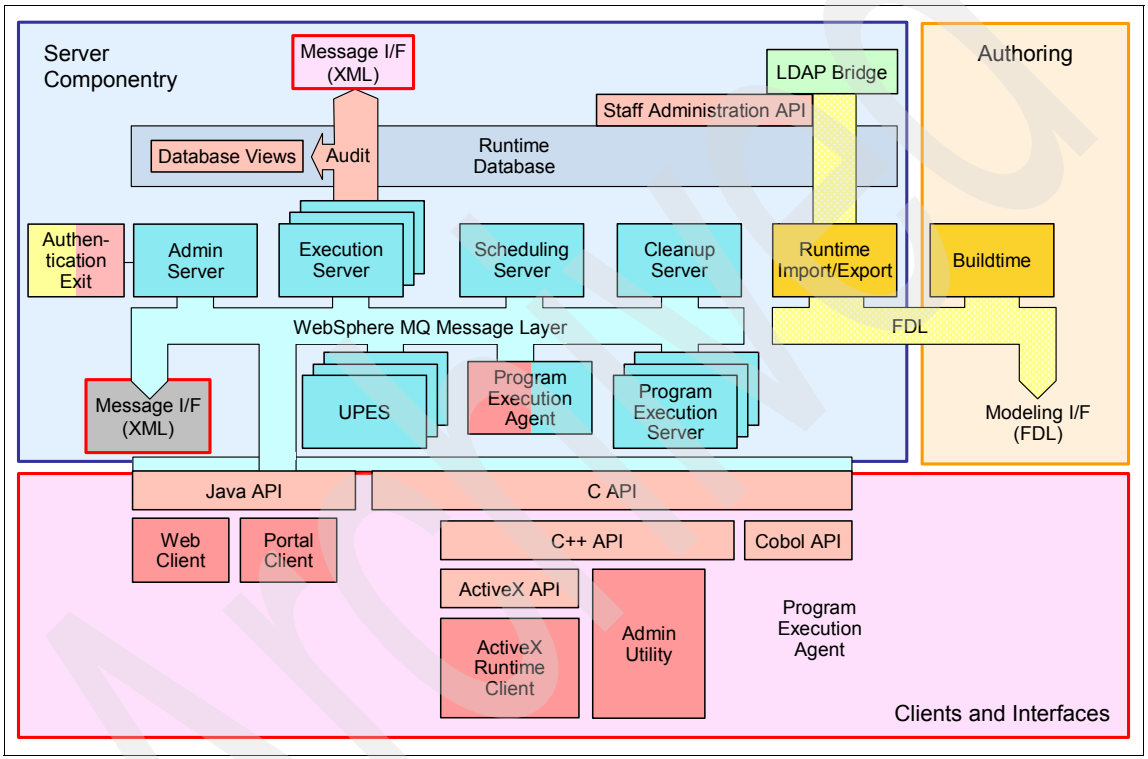


Figure 2-1 Detailed WebSphere MQ Workflow architecture

Moving from the bottom to the top, WebSphere MQ Workflow's functionality is exposed as Java, ActiveX®, C, C++, and Cobol APIs. The Java API is used by a Web client and a portal client, the ActiveX API is built on top of the C++ API and is used by the standard client, and the C++ API is used by the system's own administrative utility. For the COBOL API there is no predefined client.

The modeling component on the right-hand side of Figure 2-1 depicts the Buildtime component and a Buildtime database. Buildtime is a graphical editor

that allows the user to build business flows using different primitive elements that can be put together to form a business process.

Some of the primitive elements are input and output containers for every activity that is modeled. In Buildtime activity flows (processes) as well as data flows are modeled. The result of the modeling is represented as an MQ Workflow Definition Language (FDL) file, which contains more information than the activities and the data structures, but also the setup of the servers and information about the staff that is allowed to work with the business processes. The FDL expressions are stored in a Buildtime database.

In order to run a developed flow in the Runtime environment the FDL needs to be imported into the Runtime database. WebSphere MQ Workflow stores all user IDs in its own Runtime database. The depicted Lightweight Directory Access® Protocol (LDAP) bridge is a means to import and export user data to and from the WebSphere MQ Workflow Runtime database.

The WebSphere MQ messaging layer is the communication bus for all the other depicted components. All communication is done via queues.

Four different servers build the heart of the Runtime environment:

- The *administration server* is the component that allows an operator or system administrator to start and stop the execution servers of the workflow systems.
- The *execution servers* are the components that navigate through a business model. Navigation in that context means computing, which is the next activity in the model that can be started. Automatic activities are started and human activities are sent to the people who should perform them.
- The scheduling server is the component that allows for timed events to happen. For instance, a process may be suspended for a specific amount of time, or a time out occurs after an activity has not been dealt with in a given time window.
- The *cleanup server* is the component that physically deletes process instances in the Runtime database. It uses the scheduling server in order to run at specified time intervals. Typically, it deletes entries of processes that have successfully been finished either immediately or later at a specified time.

On top of the execution servers there is the arrow that denotes the audit trail information that is written to the Runtime database. The audit trail also has an Extensible Markup Language (XML) exit that allows customers to build their own audit system or use the WebSphere Business Monitor .

The WebSphere MQ messaging layer also depicts an XML message interface. This allows customers to build their own XML clients.

All servers on top of the messaging layer are system components that deal with the inner workings of the workflow system. The components at the bottom of the messaging layer deal with the user interaction with the workflow system. The first one to mention is the user-defined program execution server (UPES). This can be viewed as a generalized queue to which the workflow system sends a message. A UPES is the means for a user to execute a program by sending a message to a queue that program listens to. There can be more than one UPES in a given workflow.

The program execution agent (PEA) is the means with which to start a program on the client's machine in order to help fulfill a specific task. An example of this is the use of an editor in order to fill in a form. This editor would automatically be started by the PEA when the execution server navigates to the activity that uses this editor on behalf of the user.

The program execution server (PES) is available for z/OS only and supports the invocation of IMS and CICS® transactions. It has an open architecture and therefore program invocation interfaces (for example, TSO) can be plugged in.

2.1.2 System hierarchy

WebSphere MQ Workflow is organized hierarchically. The top-level hierarchy is called *domain*, representing all or parts of your organization. There can only be one domain per setup. Any properties defined at this level, such as staff, data structures, programs, and IT resources, are valid for all systems and are inherited by all lower levels.

The next lower level is the system group. Within a system group, all systems share the same database. By installing more than one system in a system group, the workload for process execution can be distributed while still sharing the same data and the same workflow model. Databases cannot be shared by several system groups.

A *system* contains the client/server components that are needed to run the business processes. It hosts a set of special-purpose servers that use the same queue manager and that are commonly managed.

The components of a WebSphere MQ Workflow system are designed for a multi-tier structure. The scope of each tier is clearly defined to exploit the available computing resources. The major components and their respective tiers are shown in Figure 2-2.

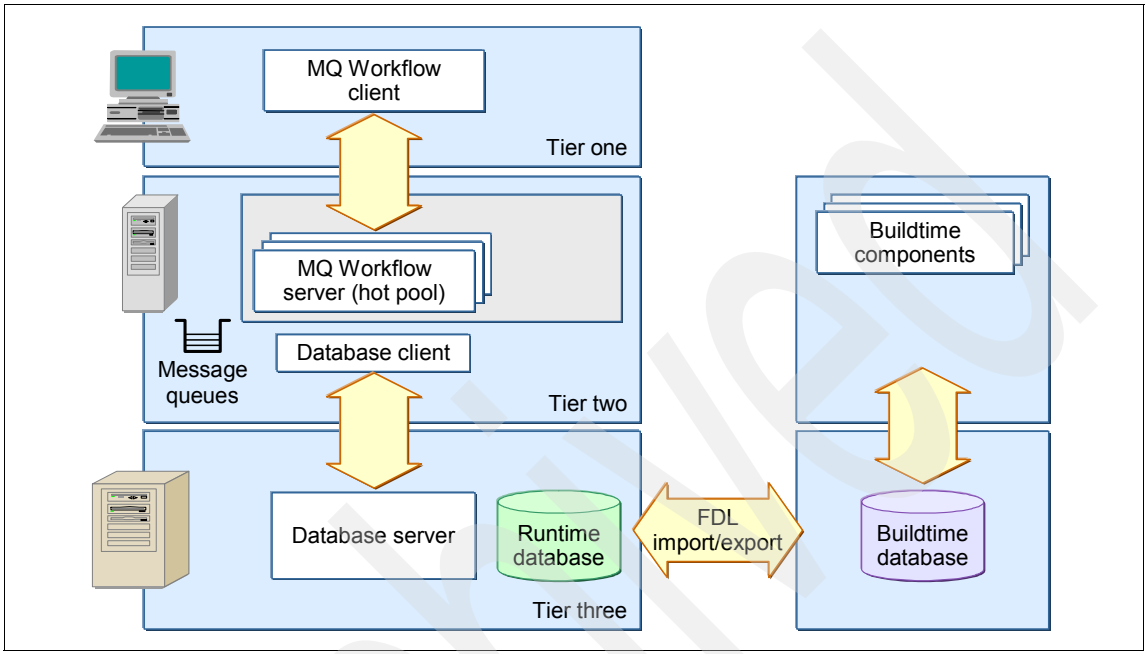


Figure 2-2 Three-tier architecture

The tiers are:

- Tier one: Client components

  Tier one represents the client APIs of WebSphere MQ Workflow and the clients that use these APIs. Clients are responsible for executing the program activities that interact with users. They give users access to the workflow management system (that is, the work items and running processes) and enable them to monitor the processes. Clients communicate with the servers through WebSphere MQ or via a Web-based client.

- Tier two: Server components and Buildtime

  Tier two represents the server components and Buildtime of WebSphere MQ Workflow. The server components manage the execution of processes at runtime. These components can be distributed across several machines to achieve workload balancing. For communication between the server components, WebSphere MQ message queuing is used.

- Tier three: Databases for Buildtime and runtime

  Tier three represents the database tier. The database holds workflow-relevant data for a system group of WebSphere MQ Workflow. This includes status and setup information. For communication between the database server and its client, the transport protocols supported by the database product are used.

Note: Depending on the size of the organization and the size of the workflow model, the database can also reside on the same machine as all other server components. The system then consists of only two tiers.

In a two-tier structure, the server components, the message queuing components of WebSphere MQ, and the database management system reside on the second tier.

2.1.3 The Buildtime components

With Buildtime workflow models are created and system resources are defined.

Buildtime offers a graphical editor for creating process models. Note that the flow of activities is modeled as well as the flow of data. Other features in Buildtime allow us to define the organization and the programs to be used in the workflow as well as network definitions.

Existing workflow models can be imported into WebSphere MQ Workflow or exported in the MQ Workflow Definition Language (FDL). Workflow models can also be exported as HTML, for instance, in order to print them.

When a workflow model is ready to use, the model is translated into a template that can be started from a WebSphere MQ Workflow client and managed by the server components.

2.1.4 The client components

The WebSphere MQ Workflow client starts processes and monitors their execution. The administration utility administers the system and the program execution agent invokes application programs that are used in the workflow.

WebSphere MQ Workflow client

With a WebSphere MQ Workflow client the execution of processes is started, worklists are used to manage work items, and processes are monitored. WebSphere MQ Workflow offers the following clients as part of the product:

- A *Web client based on the Java API*. This client is a Java servlet that provides a Web interface for WebSphere MQ Workflow. It offers full process and worklist control as well as process-monitoring functions.
- A *portal-based client* that provides access to the WebSphere MQ Workflow system from within IBM WebSphere portal server. The functionality is similar to the IBM WebSphere MQ Workflow Web client. As with the WebSphere MQ Workflow Web client, the look and feel of the portal client can be customized, and its functionality can be extended.
- A standard WebSphere MQ Workflow *client based on the Active X APIs*.

Besides these clients customers may have built their own clients:

- A custom client based on APIs

  Customers may have designed their own interface to perform worklist or monitoring tasks with a custom client using the WebSphere MQ Workflow APIs.
- A Web-based custom client based on the Java APIs

  Customers may have designed their own interface to perform worklist or monitoring tasks with a custom client using the WebSphere MQ Workflow Java APIs.
- A custom application

  A custom application allows you to address other specific business needs, for example, an automated batch application to start business processes.

Administration utility

The administration utility is the administrator's user interface to request services from the administration server. Using this utility, a WebSphere MQ Workflow system is started and stopped and the current state of any server can be listed. Furthermore, error logs can be accessed to check whether any problems exist within the system.

2.2 WebSphere Process Server architecture

IBM WebSphere Process Server is an advanced business process integration platform that offers a Runtime environment for business processes that have

been defined using the industry standard Web Services Business Process Execution Language (WS-BPEL).

IBM WebSphere Process Server builds on the capabilities of WebSphere Application Server Network Deployment 6.1, which is based on the Java 2 Enterprise Edition 1.5 (J2EE). Hence, it inherits and exploits the quality of service this Runtime environment offers, namely clustering, failover, scalability, and security.

WebSphere Process Server is complemented by WebSphere Integration Developer (WID), a tool to develop BPEL processes by means of using a graphical editor, Java code, and GUI components using Java Server Faces (JSF) and Java Server Pages (JSP™). It integrates all the different components by means of the Service Component Architecture (SCA) and builds all other artefacts that are supported by WebSphere Process Server.

2.2.1 Architectural overview

Figure 2-3 depicts the main components of the WebSphere Process Server. Business processes are managed by the Business Flow Manager (BFM) and the human tasks are managed by the Human Task Manager (HTM). Together they represent the Business Process Choreographer (BPC).

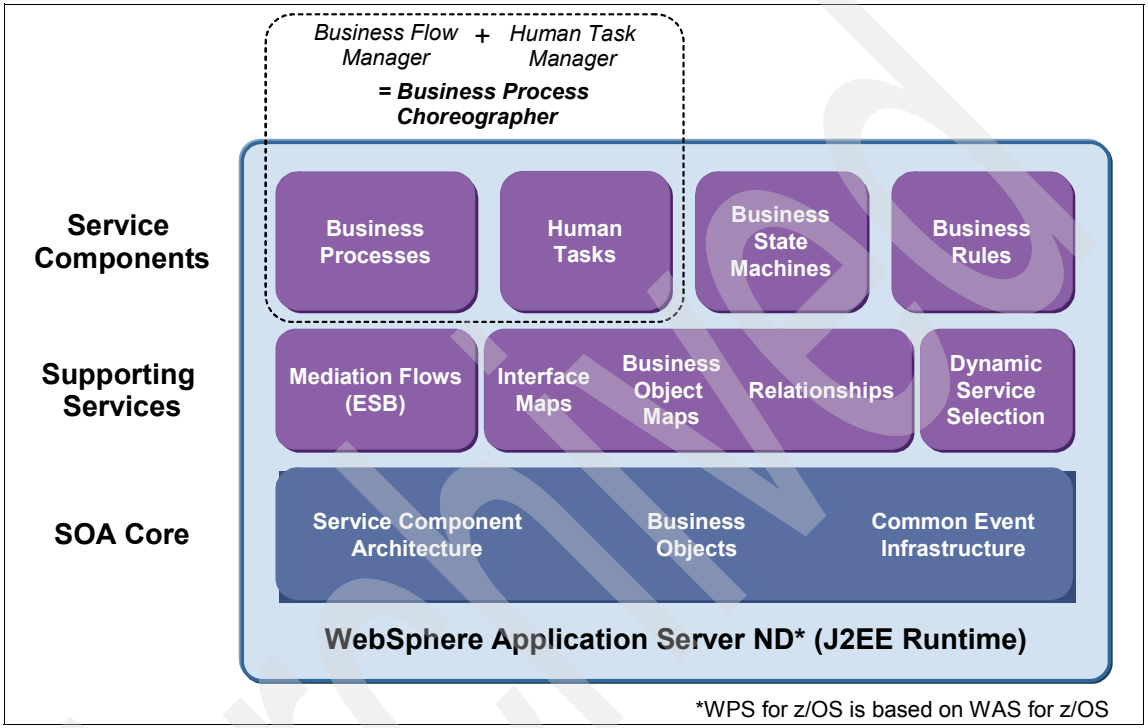


Figure 2-3 WebSphere Process Server architecture

At the bottom of Figure 2-3 we see the base infrastructure on which WebSphere Process Server V6.1 is built: WebSphere Application Server Network Deployment (ND) or WebSphere Application Server for z/OS V6.1, which itself is based on a J2EE 1.5 runtime.

Service-oriented architecture (SOA) core

Above this infrastructure layer there is the service-oriented architecture core consisting of the Service Component Architecture (SCA), business objects (BOs), and the Common Event Infrastructure (CEI).

Service Component Architecture

In a service-oriented architecture where many different services must be integrated a common invocation model as well as a common data model eases this task. Service Component Architecture is this invocation model. Every integration component is described through an interface and possibly through references. The interface describes how the component can be called and the references describe which other components this component calls. These services are then assembled in the Component Assembly editor, which is part of the WebSphere Integration Developer (WID), thus enabling a very flexible and encapsulated solution.

As depicted in Figure 2-4, a component can be anything from a business process described in WS-BPEL to a Plain Old Java Object (POJO). Multiple components must be wired together in order to form a deployable solution. All components are represented consistently and are invoked identically. This means that they can be reused for other assemblies as well.

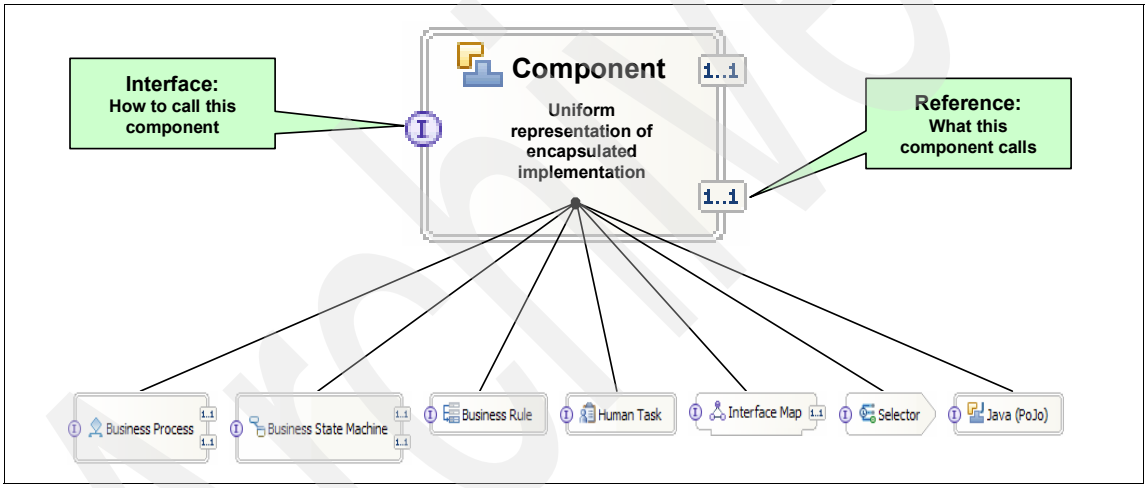


Figure 2-4 Service Component Architecture

The interface of a component describes how that component can be called from other components. The references of a component describe which other components are being called. References are being wired to interfaces.

A service interface is defined by a Java interface or a Web Services Description Language (WSDL) service, a so-called portType.

The service wiring resolves the references and is done by using the assembly editor. This wiring allows for the creation of SCA applications by component assembly.

Services can be packaged as larger units in a service module, which is the basic unit of deployment and administration in an SCA runtime. Figure 2-5 depicts such a module assembly.

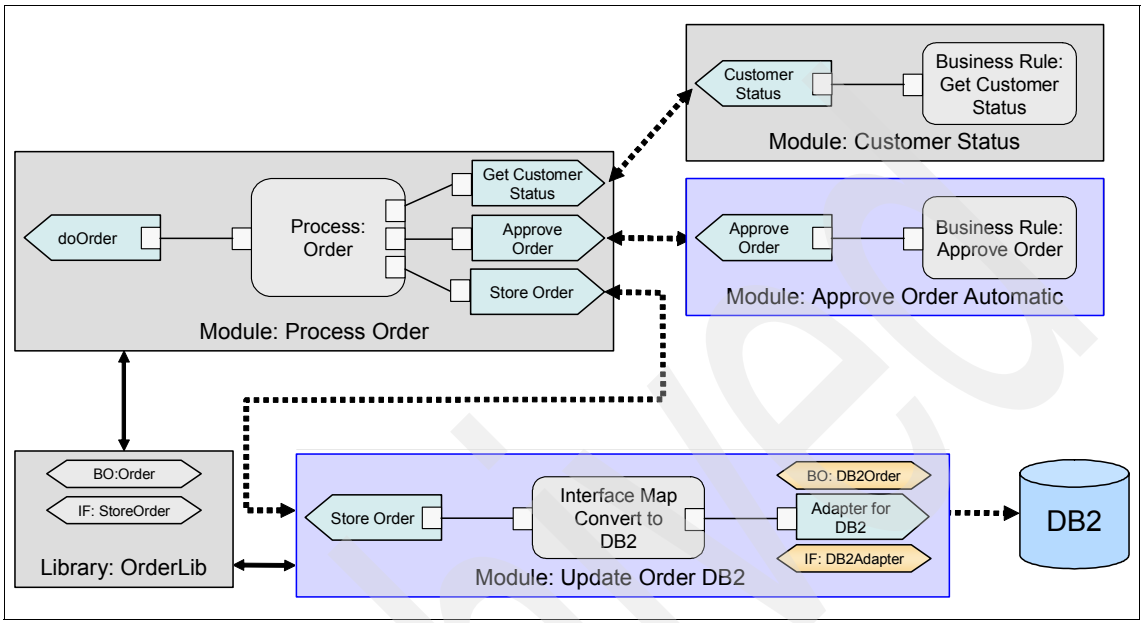


Figure 2-5 Multiple SCA modules

Looking closely at the *process order* in Figure 2-5, it becomes apparent that it exposes its service as portType *doOrder*. This is called the export of the process. It uses three other services for which references are needed:

- Get customer status
- Approve order
- Store order

These references are service imports. In Figure 2-5 we see that they are wired to the appropriate modules realizing these services. It now also becomes clear how a module can be easily exchanged with a different implementation.

An interface needs to specify the data structure that is passed to the service as well as the resulting data structure. This leads to the definition of business objects as the common data representation in WebSphere Process Server.

Business objects

Business objects are the universal data representation within WebSphere Process Server. They are used as data going in and out of services and are

based on the Service Data Object (SDO) standard (JSR253). Business objects decouple data definitions from their actual implementations. See Figure 2-6.

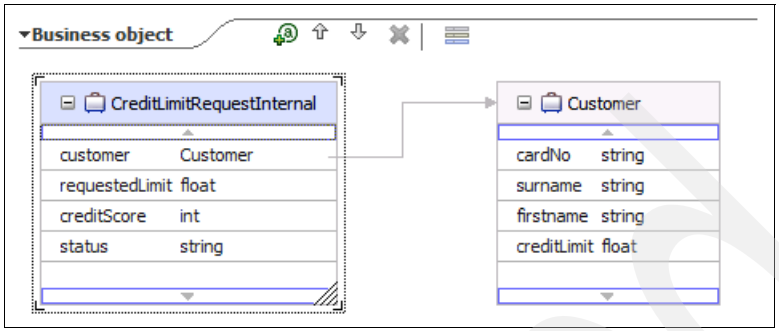


Figure 2-6 Business object definition

Business objects can be represented as XSDs and as Java classes. Each data item in a business object has a name and a type. The type can be either simple or complex, scalar, or of type array.

Common Event Infrastructure (CEI)

The Common Event Infrastructure is the foundation for monitoring applications. IBM uses this infrastructure throughout its product portfolio, and monitoring products from Tivoli® as well as WebSphere (WebSphere Monitor 6.1) exploit it. The event definition (Common Base Event, CBE) is being standardized through the OASIS standards body so that other companies as well as customers can use the same infrastructure to monitor their environment. The WebSphere Integration Developer (WID) tooling allows for a very fine-grained setting of the events that should be emitted from any given component. Figure 2-7 shows the overall picture of CEI

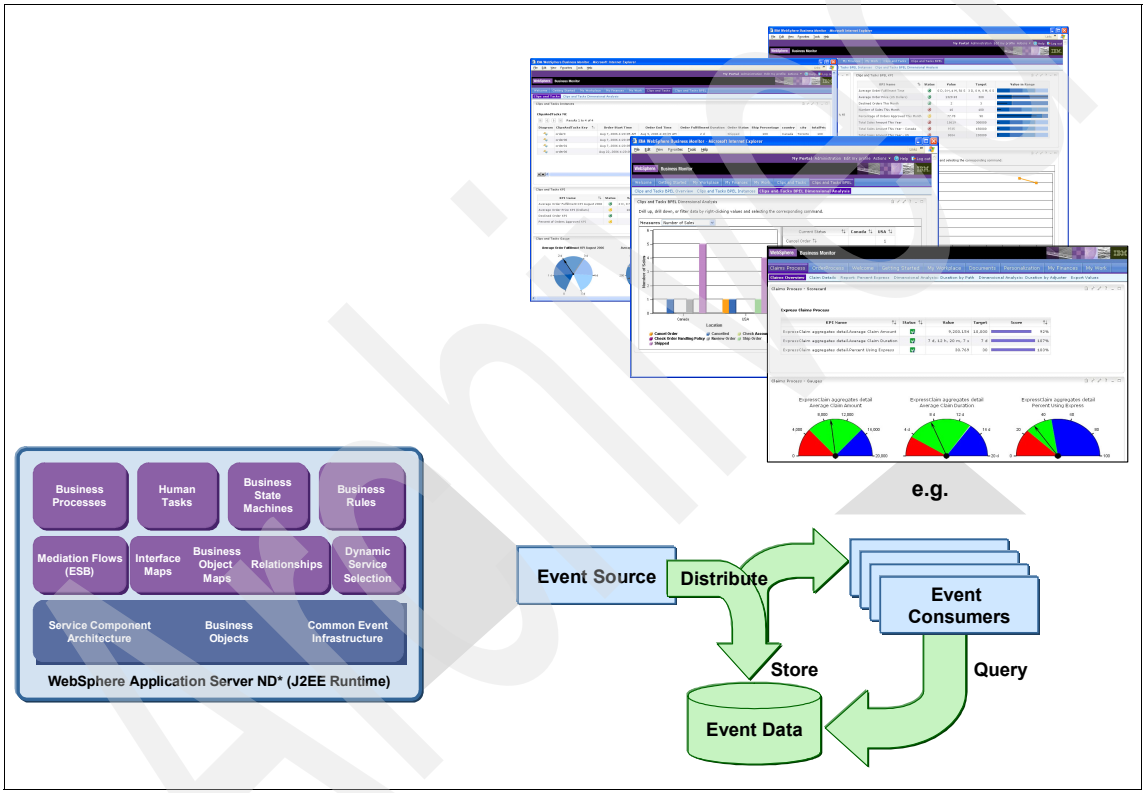


Figure 2-7 CEI event emitters and consumers

On the left-hand side of Figure 2-3 on page 16 all of the components of WebSphere Process Server are shown as event sources. WebSphere Integration Developer is used to configure which events are emitted by WebSphere Process Server. Be aware that other components than WebSphere Process Server can emit events as well. All emitted events can be stored in a database and can be distributed to event consumers. The event consumers can query the event data

store for different purposes, for example, to aggregate and then display historical events using different kinds of graphs. Typical data that are often displayed are, for instance, the number of processes per unit of time, how many processes needed manual intervention, and how many processes sent out an e-mail notification.

The built-in Business Process Choreographer Observer already allows us to monitor the health of the business process engine, number of active processes, and number of finished processes per day. It also allows for drill-down capabilities to retrieve statistical data on processes and activities.

Supporting services

Above the SOA core there are the supporting services. There are the mediations from the Enterprise Service Bus (ESB) and different kinds of mapping and selectors.

WebSphere Process Server V6.1 contains an ESB, a layer that allows different services to exchange data. When the data are in different formats a mediation is necessary that transforms the output of one service such that another service can digest it. This level is called mediation.

As transformations happen quite frequently in any given integration project, WebSphere Process Server V6.1 supports these four kinds of mediations:

- Maps: These transform one kind of business object into another.
- Relationships: Key management for same objects in different data stores (for example, a customer record has a different identifier in SAP® than in Siebel®).
- Interface mediation: To translate between semantically identical but syntactically different services, for example, updateCustomer (Customer) to updateCustomerInSAP (SAPCustomer).
- Dynamic service selection: This allows for the invocation of a different component (with the same interface) based on business rules.

These supporting services allow you to concentrate on the task of flexibly orchestrating services using different modeling techniques.

Service components

On this level four different elements for orchestrating a business process are shown:

- Business processes
- Human tasks
- Business state machines
- Business rules

Business processes

Business processes are modeled in WebSphere Integration Developer (WID). Once modeled they can be elements of other business processes. By using the common service invocation architecture a business process can call any kind of service, Java code, a human task, and any service that is described by a Web Service Description Language (WSDL) expression. Web Services Business Process Execution Language (WS-BPEL) offers many primitives out of which a business process can be assembled. A complete list can be found in the specification of WS-BPEL2.0. Besides the primitives WebSphere Process Server offers additional elements that are not part of the WS-BPEL specification, most notably:

- Human tasks
- Compensation
- Java snippets

On the left hand side of Figure 2-8 is a sample business process modeled in BPEL.

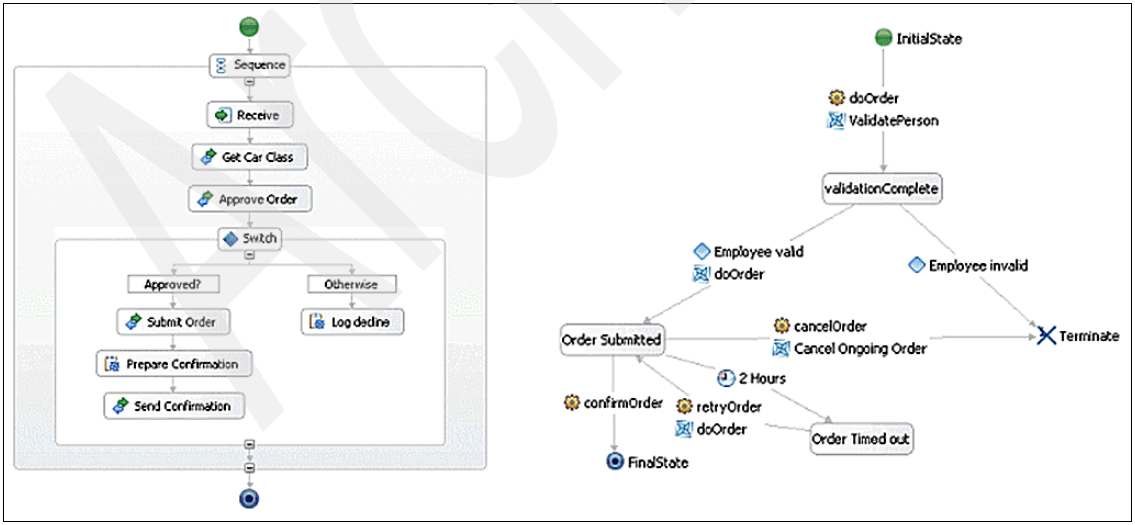


Figure 2-8 BPEL process and business state machine modeling

Human tasks

Human tasks are another element of a business process. Not all tasks can be done by an automated service that can be realized as a program. Some services, for instance, the review of an article, must be performed by a human being. This can still be modeled in a business process using the means for modeling human tasks provided in WebSphere Process Server.

Two kinds of human tasks are depicted in Figure 2-9. *Verify data* and *approve* are so-called inline To-do tasks. The *rework,* on the other hand, is a standalone task as an SCA component that is called from an invoke that is a BPEL primitive.

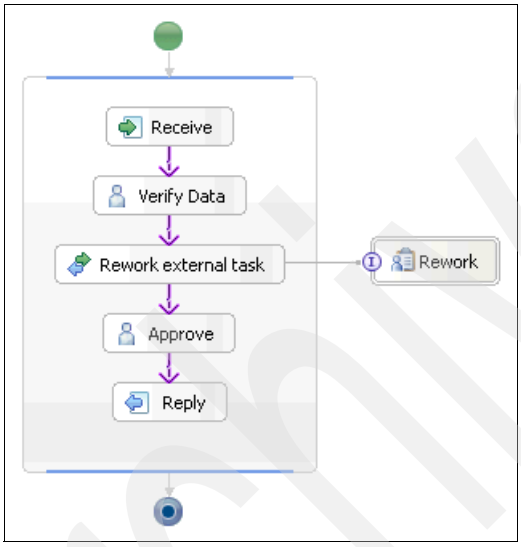


Figure 2-9 Inline to-do and standalone tasks

Inline to-do tasks have access to the process context, that is, all the variables that are defined in a business process. Standalone tasks as SCA components do not have this privilege. Four different kinds of human tasks are distinguished:

- To-do tasks: This is the classic scenario in which the machine creates a task for a human, for example, a review tasks for a reviewer.
- Invocation tasks: This is the typical invocation of a business process or any other automated task or service such as calling a SOAP service or a CICS transaction.
- Administration tasks support human administration of business processes and their activities.
- Collaboration tasks: In this scenario a person creates a task for another person (human), for example, a travel approval request for the person's manager.

Figure 2-10 shows three types of human-to-human tasks, and their differences are explained.

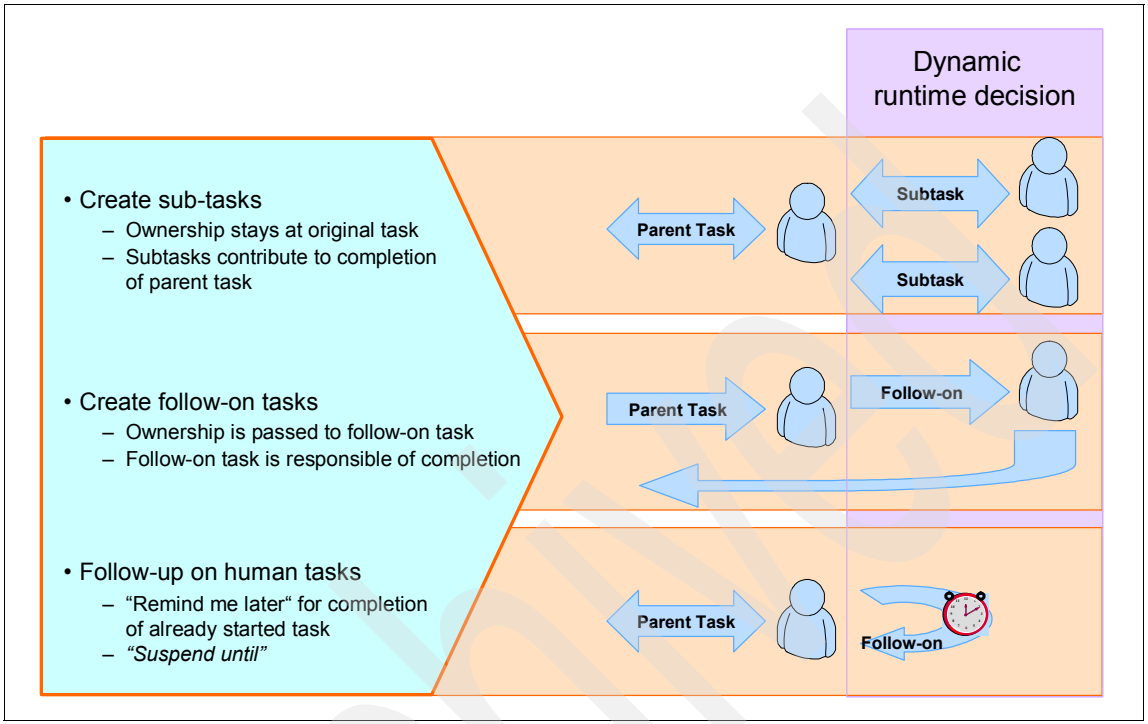


Figure 2-10 Collaboration scenarios

WebSphere Process Server V6.1 supports escalation, authorization, the ad-hoc creation of new tasks (Figure 2-10), dynamic assignments, and substitution. The latter is the ability to define a substitute for a person who would normally do a specific task. If that person becomes unavailable, for instance, because of illness, the task automatically goes to a substitute.

Business rules

Business rules can be used in orchestrating services to business processes. They can be if/then rules or decision tables. A Web client is included where the parameters of these rules can be changed by a business user. A natural language specification of these rules (for example, if the car is size <xxx> then the price is <yyy>) is allowed, meaning that the business person does not have to understand the implementation of the rules. Business rules can be changed via this Web client and the changes take effect without the business process having to be redeployed.

Business state machines

Business state machines are another way of modeling a business process. This is shown on the right-hand side of Figure 2-10 on page 24. There are some processes that are highly event-driven, for example, an order could be cancelled at any point during the order process. It is difficult to model these kinds of processes in a “traditional, sequential business process. Therefore, another service is offered where UML state machine diagrams can be used to describe such event-driven business processes. Note that under the covers these are implemented as WS-BPEL processes.

2.2.2 System topologies

Figure 2-11 shows the three main topologies that account for a large number of customer setups. The installation of these topologies is supported by templates and wizards in the administration console.

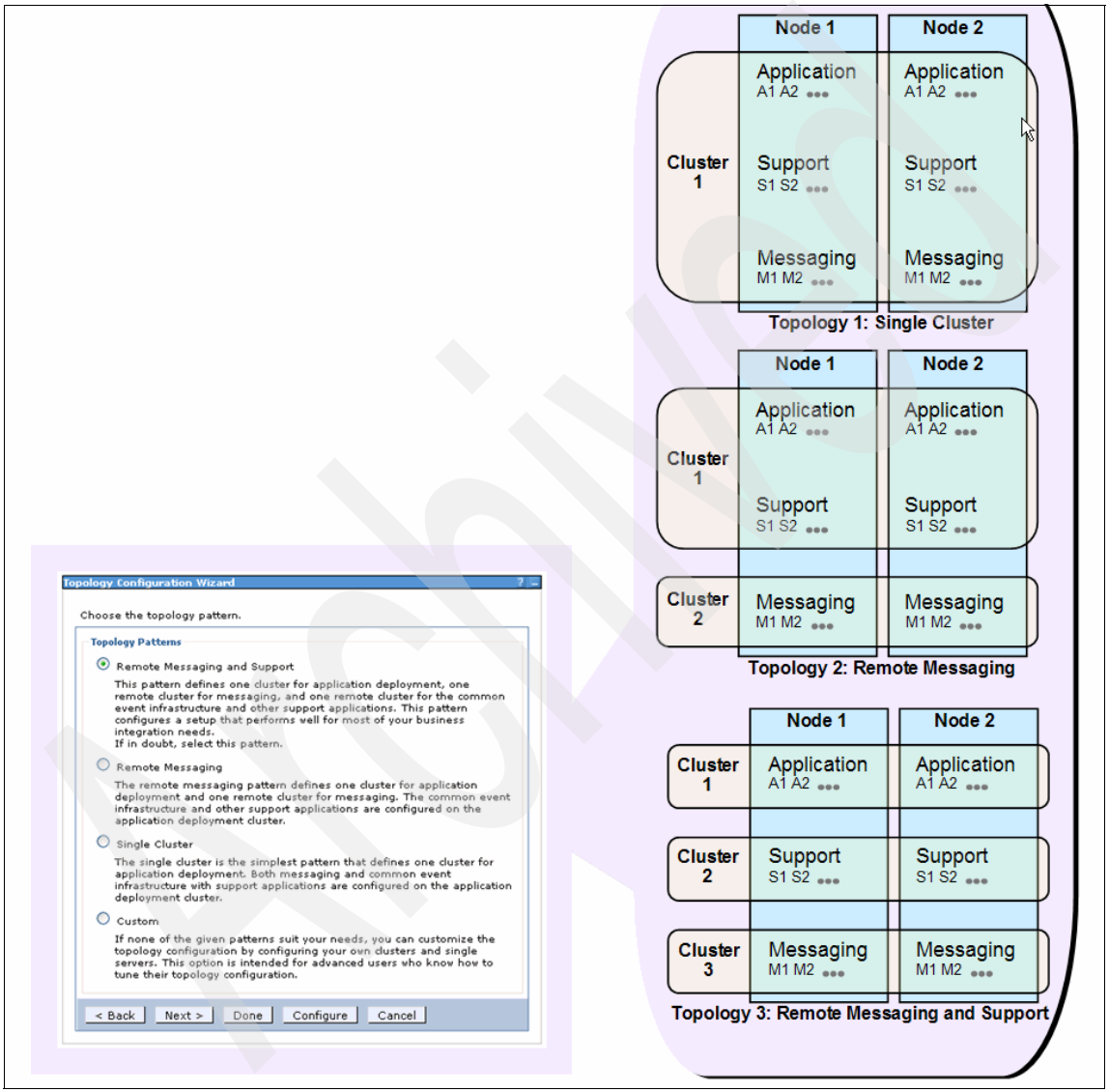


Figure 2-11 System topologies

This template-based clustering reduces the amount of time needed to set up these complex environments. Topology one is a single cluster, topology two clusters the messaging engines separately, and topology three consists of an application cluster, a support cluster, and a messaging cluster.

2.2.3 Clients

WebSphere Process Server offers two out-of-the-box clients. The *Business Process Choreographer Explorer* allows us to administer processes. The included graphical process instance viewer yields a real-time snapshot of a process and its activities at runtime.

The Business Process Choreographer Observer monitors the health of a business process by showing, for example, the number of instances of a process or the number of finished processes per day.

Besides these out-of-the-box clients, WebSphere Process Server also offers client-generating capabilities for the human tasks. The clients aim at the business user. These *generators* are part of the WebSphere Integration Developer and can create JavaServer™ Faces (JSF) based clients, IBM Lotus® Forms-based clients, or portlet-based clients for WebSphere Portal.

In addition, customers can develop a number of *customized clients*:

- Customized Business Process Choreographer Explorer instances to address the business needs of particular user groups
- Custom clients using the rich set of API functions provided by the Business Flow Manager and Human Task Manager
- Custom applications to address other specific business needs, for example, an automated batch application to start business processes

3

Transition concepts

This chapter discusses the following topics:

- Transition concepts overview and approaches available.
- Business process concepts in WebSphere MQ Workflow, and how they can be mapped to corresponding concepts in WebSphere Process Server. In addition, it describes enhanced functionality available only in WebSphere Process Server.
- Features mapping and migration approach, describing how to map the major conceptual features of WebSphere MQ Workflow to equivalent concepts in WebSphere Process Server. It also provides a strategy for transition.
- Fault and exception handling in business processes and tasks.

3.1 Transition concepts overview

This chapter describes the different approaches for converting the business process to the WebSphere Process Server environment.

3.1.1 Process conversion concepts

A tool is required for the creation of the technical execution model. Since the resource performing this role is different from the business analyst who created the business model, a different tool is required for the technical resource. The business analyst specifies what needs to happen and the technical team specifies how to make it happen. In addition, a non-technical analyst is not going to have the skills needed for integration, testing, and deployment. These different requirements again point to the need for different tools.

IBM tools

The offering from IBM for creating the business model is WebSphere Business Modeler V6. The middleware software enabling the business processes is WebSphere MQ Workflow V3.6 and WebSphere Process Server V6.1 as the current process engine.

The development tool used for WebSphere MQ Workflow is called Buildtime. In addition, a development tool was needed for clients. Windows clients would be developed using Microsoft® toolsets, while Web clients were developed using tools such as WebSphere Studio Application Developer Integration Edition. For WebSphere Process Server, the tool is WebSphere Integration Developer, which includes both the tooling for the business process, along with the tooling for creating Web clients, portal clients, Lotus Forms clients, and adapters, business rules, and other constructs that were not available in WebSphere MQ Workflow.

Execution models

WebSphere MQ Workflow uses MQ Workflow Definition Language (FDL) for its execution model. FDL is based on an early attempt at standards from the Workflow Management Coalition (WfMC). This standard did not gain broad industry acceptance, however. WebSphere Process Server V6.1 uses WS-BPEL, the Business Process Execution Language. WS-BPEL has broad industry support and has been ratified as a standard by its OASIS technical committee.

WS-BPEL has many similarities to FDL, but there are unique constructs in each. When attempting a transition from FDL to WS-BPEL, these differences must be considered.

Tool import capability

WebSphere Business Modeler and WebSphere Integration Developer offer different options for transitioning models from WebSphere MQ Workflow and FDL to WebSphere Process Server and WS-BPEL. There are different options because some customers start with a business model, while others do not.

WebSphere Process Server V6.1 can import a .ORG file from Websphere Business Integration Workbench V4.2.4. This capability enables customers to move their business models from the old environment to the new one. In addition, WebSphere Business Modeler V6.1 can also import FDL files from WebSphere MQ Workflow. With these capabilities, a new business model can be created based on the old system. The business model can be enhanced with technical details, then exported to WS-BPEL, WSDL, XSD, and other related artifacts.

WebSphere Integration Developer can then be used to edit the project exported from WebSphere Business Modeler. In addition, WebSphere Integration Developer can directly import FDL. The tool for this import capability is called FDL2BPEL, and the description can be downloaded from this location:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

Usually, after creating the artifacts in one of the ways described above, you should expect that manual rework is necessary because of these reasons:

- Some capabilities of WebSphere MQ Workflow cannot be directly mapped to a WebSphere Process Server capability. Instead, a work-around is necessary in order to have a similar capability.
- The generated artifacts can be further optimized to have a better execution performance.
- The structure of the generated artifacts can be simplified, for example, by consolidating activities or reusing variables. Such a refacturing would simplify later changes in the process model.

3.1.2 Considerations for long-running processes

Long-running processes are interruptible processes, which can include human interaction and execution in parallel threads. They persist their state in the WebSphere MQ Workflow Runtime database. Long-running processes can last from seconds up to years, depending on the business model and requirements.

Running instances in WebSphere MQ Workflow cannot be automatically transitioned to WebSphere Process Server. Typically, they are completed in WebSphere MQ Workflow.

Assessing process instances in the WebSphere MQ Workflow environment

To choose the best transition approach for long-running process instances in the WebSphere MQ Workflow environment, their average and maximum duration should be determined. This helps to get an overview of the environment to be transitioned.

For every process that needs to be transitioned we recommend capturing the following, if available:

- Number of completed process instances
- Average duration of a completed process instance
- Maximum duration of a completed process instance
- Number of active running process instances
- Average duration of an active process instance
- Age of oldest active process instance

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

- Business operations knows what duration to expect for a process instance of a certain type. There also might be custom logging of values usable for determining the duration of a process in the application.
- A technical evaluation of the data found in the WebSphere MQ Workflow audit trail database can be performed. Events for process start and finish can be used to calculate the average and maximum duration based on these results. Auditing must be enabled for the process in order to use this data.

We recommend using both the business and the technical approach described above. This helps to get a complete picture of the existing environment, especially if discrepancies are observed between the results of the two approaches.

Having obtained these numbers, we know:

- How long the process instances typically take until completed

  This determines how much time would be needed to finish the running process instances in the WebSphere MQ Workflow environment.
- What is a reasonable time after which a process instance can be terminated

  It might be the case that some processes take even longer than the expected maximum time. A plan for what to do in this case must be created as well. For example, consider a bank running a processes for loan approvals. There is a maximum time for this process. Processes taking longer than this time might not be valid any longer, as the loan approval applicant does not wait forever to get a loan. So, it might be possible to terminate these process instances.

There can be cases in which the processing of a business operation requires multiple processes. For example, let us say that the business operation for loan approval consists of three parts, each represented by a business process. Only the completion of these three process instances completes the business operation.

In this case, the duration of the business operation must be calculated as well. This is more difficult to determine. One approach to follow here is to get the durations for the individual processes as described above, and then calculate the minimum, average, and maximum durations for the entire business operation based on the values of the individual business process instances. Figure 3-1 shows how process duration is calculated.

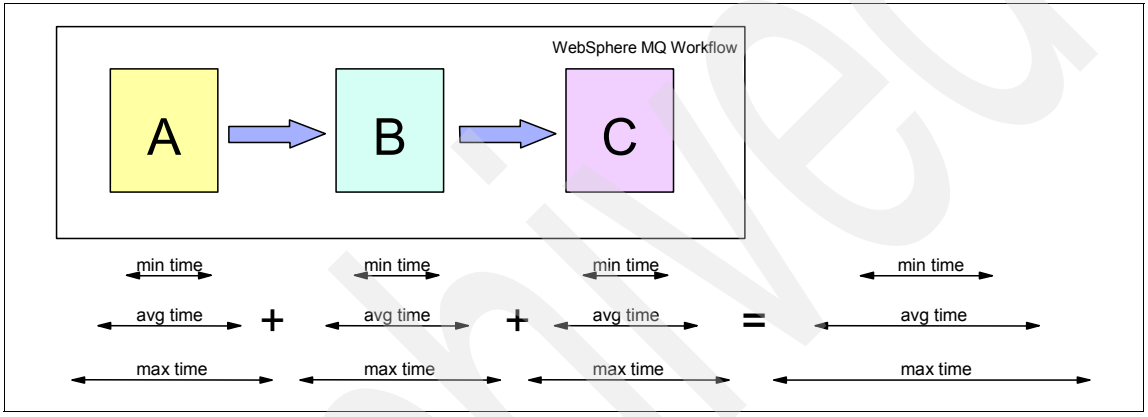


Figure 3-1 Process durations

Coexistence approach

Coexistence is the means of running both environments, WebSphere MQ Workflow and WebSphere Process Server, in parallel for a defined period of time. This ensures that instances of long-running business processes can finish in the old environment, while new processes are started in the new environment. This approach requires adapting your client to manage calls to both systems for the time of coexistence, as shown in Figure 3-2.

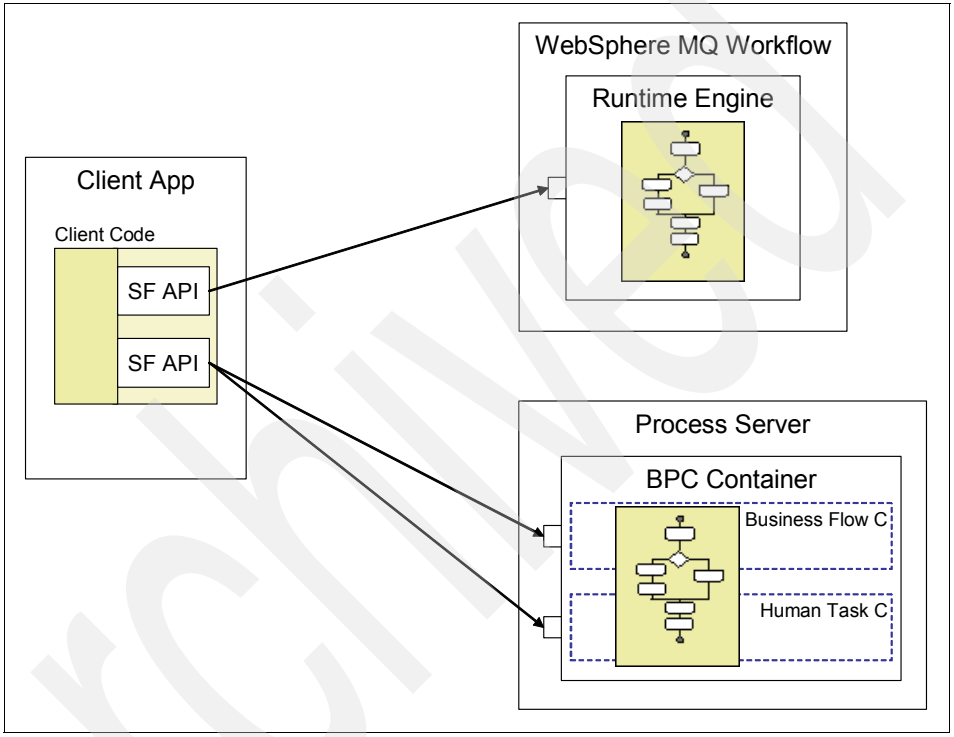


Figure 3-2 Coexistence approach

In some cases an alternative could be to run the two clients concurrently for the period of transition. This might be an acceptable approach if the number of human tasks in the active instances is limited and there is a short transition phase.

Using this approach you can build up the WebSphere Process Server system, deploy and test your transitioned application, test a new version of the client talking to both systems, and, at some point in time, go live with both systems by changing the client to direct new process starts to WebSphere Process Server while still handling requests for the WebSphere MQ Workflow process instances.

The disadvantage of this approach is that there probably is the need for additional hardware to run the new environment, as well as additional administrative and maintenance efforts during the time both environments are running in parallel. It may be a potentially high investment to make a client work that federates across WebSphere Process Server and WebSphere MQ Workflow.

In order to validate whether the coexistence approach is the best way to go, how long the WebSphere MQ Workflow processes will take until completion should be calculated, as described in “Assessing process instances in the WebSphere MQ Workflow environment” on page 32. This defines the necessary time of running both environments in parallel. The time for which these two systems and the extra effort must be spent should be sized and part of the transition planning to make sure that hardware and human resources are available during this phase of coexistence.

If there are process instances of several templates to be transitioned, such as a loan approval process and a travel booking process running on the same WebSphere MQ Workflow environment, the order of transition should be determined. Process instances with the longest duration should be transitioned first and processes with short duration last. This minimizes the period of coexistence of WebSphere MQ Workflow and WebSphere Process Server.

If all are transitioned at once, the period of coexistence is the time to complete an entire business operation. The second possibility, transition of these processes one after another, has the advantage that it could reduce the time required for coexistence, especially when business processes at the end of the call chain have a rather long duration.

Figure 3-3 shows the *migrate the longest last* approach.

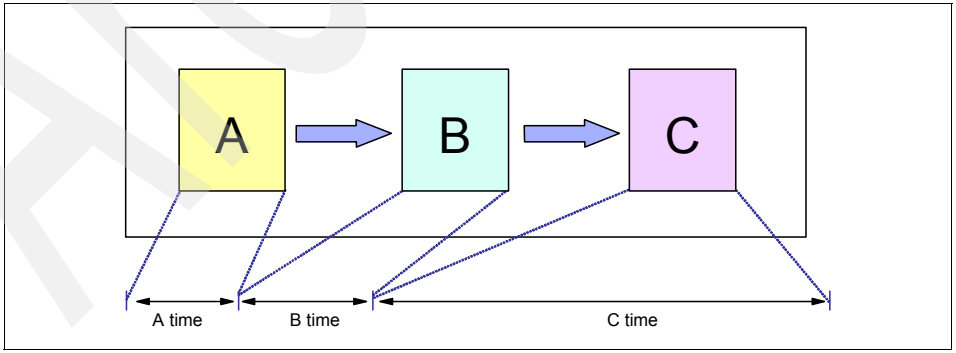


Figure 3-3 Scenario: Migrate the longest last

This approach starts moving the last process in the call chain to WebSphere Process Server, followed by the preceding one, until the first process in the call chain can be transitioned to WebSphere Process Server. The disadvantage of this approach is that it requires modifications in the remaining WebSphere MQ Workflow processes in the old environment to call the processes already running on WebSphere Process Server.

Adopt client applications

When WebSphere MQ Workflow and WebSphere Process Server environments are running in parallel, you must ensure that new processes are started only in the new environment, while the user must be able to work on staff activities or human tasks of both systems.

Therefore, the client applications interacting with WebSphere MQ Workflow must be enhanced to cope with both systems for the time of coexistence. The Java APIs for client interaction are different in the new environment. This means that the client must be changed as well be able to interact with the WebSphere Process Server system.

In addition to the client transition, the client applications must be enabled to cope with the two environments during the period of coexistence. The client needs to determine where to direct the requests it gets on a per-request basis.

A process start request for a transitioned process instance needs to always be directed to WebSphere Process Server.

If the period of coexistence is short, it may be justifiable for users to use two client applications instead of spending development effort to provide a common view of WebSphere Process Server and WebSphere MQ Workflow environments. This saves development and test time. The WebSphere MQ Workflow client can be phased out together with WebSphere MQ Workflow at the end of the period of coexistence. A decision for this approach depends on the client functionality and the kind of users to which the client is exposed.

Move-at-once approach

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

The only additional development effort that is needed is to provide a tool that supports the transition of the process instances to the new environment. On a high level, this tool needs to accomplish the following steps for each process instance to migrate:

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

This approach is applicable to very few business processes, as usually it is not possible to repeat steps. Too much work might be required for a redo of work performed through human interaction. Back-end systems participating in the process would need to be updated during this transition operation as well. These disadvantages may make it inadvisable to use this transition approach. This approach could, however, be applicable in the following situations:

- For processes requiring a short period of time until completion. Resubmitting these process instances in the new environment would create a backlog of work of, at maximum, this time on the new system and for the users.
- The number of running process instances is small and enables handling of the processes manually after resubmitting them on WebSphere Process Server. An administrator could manually complete the human tasks that have been performed in the WebSphere MQ Workflow process to get the new Process Server process instance to the same state it was at in WebSphere MQ Workflow before the process instance was suspended.
- The processes are structured in a way that enables them to be resubmitted without causing conflicts. An example of such a process is one requiring one or more approval steps before executing work. The work then is all automatic. Completion is fast and without wait times. For such process instances it can usually be expected that they are waiting at one of the staff activities for human interaction. Suspending them and resubmitting them on the new system would then be possible and at the most, some staff activities must be repeated.

However, before choosing this move-at-once approach, a very careful case-by-case analysis must be done to ensure that this is an acceptable option.

3.1.3 Considerations for process instance transition

If no running process instances must be transitioned, a move from WebSphere MQ Workflow V3.6 to WebSphere Process Server V6.1 has some advantages.

It can be done in a short time period and avoids parallel installations of the old and new environment. It requires only a minimal system-down time and it can be done without data loss.

Dry-out (sunset) approach

This approach does not require a transition of process instances. WebSphere MQ Workflow and WebSphere Process Server are running in parallel:

- New process models are created in WebSphere Process Server or source artifact transition is used.
- The WebSphere MQ Workflow environment is switched off when all long-running process instances have completed.

Concerns when using this approach are:

- The period where both environments are running in parallel might be very long (due to long-running process instances).
- WebSphere MQ Workflow cannot be sunset and two parallel systems must be maintained.

3.2 Business process concepts

For a transition from WebSphere MQ Workflow to WebSphere Process Server it is important to understand that WebSphere Process Server is a conceptual progression and not a simple product upgrade from WebSphere MQ Workflow.

3.2.1 Assessing WebSphere MQ Workflow conceptual constructs and mapping them to WebSphere Process Server

Both products have very different base architectures and it is impossible to do feature-to-feature mapping for all functionalities. Furthermore, there are some features of WebSphere MQ Workflow that have no direct implementation in WebSphere Process Server.

Constructs

Table 3-1 lists the conceptual constructs of WebSphere MQ Workflow and maps them to the equivalent constructs in WebSphere Process Server (programming model/constructs). For the most current description of the mapping constructs see SupportPac WA73, which can be downloaded from this location:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

Table 3-1 Constructs

	WebSphere MQ Workflow: FDL construct	**WebSphere Process Server WS: BPEL, BPEL extension, or human task construct**
Data	Data structure	Business object: Specified by XML schema definition
	Input/output container of activity	Variables: Used to specify the input/output of involved activity
	Global data container	Variable with query properties
Interfaces	Source and sink	Receive and reply activities with corresponding variables storing process input/output
	Process interface (inbound interface) Interaction with programs and/or processes (outbound interface)	Partner links
Activities	Program activity	Invoke activity or human task activity (invoke service or human task)
	Process activity	Invoke activity (invoke subprocess)
	Empty activity	Empty activity (point of synchronization)
	Block activity	Flow activity
Data row	Data connector	Assign activity or Java Snippet (initialization of variables, copy data between variables)

	WebSphere MQ Workflow: FDL construct	WebSphere Process Server WS: BPEL, BPEL extension, or human task construct
Control row	Process with activities	► Process: Long-running ► Flow (concurrent activities) ► Sequence (sequence of activities) ► While (repeated activity)
	Control connector	Link (synchronization dependency between two activities)
	Staff assignment	Staff verb (for example, specifying potential owners of *human task*)
	Transition condition	Transition condition (of link construct)
	Start condition of activity	Join condition of activity
	Exit condition of activity	Condition that controls a While activity
Topology	Domain	N/A: WebSphere Application Server ND/XD
	System group	
	System	
	Various types of servers	
	Queue manager	
	Program execution agent	N/A
	Program	N/A → SCA Import
Staff/ organization	Person	N/A → Staff schema/staff directory (for example, LDAP)
	Organization	
	Role	
	Level	
	Process category	N/A
Process Types	Process	Microflows Long-running processes: Inclusive specification of transactional behavior
Fault handling	Fault handling modeled explicitly (for example, _RC)	Fault Handler, continueOnError flag BPEL compensation for long-running processes, Microflow compensation

	WebSphere MQ Workflow: FDL construct	WebSphere Process Server WS: BPEL, BPEL extension, or human task construct
Event handling interaction with existing process instances	Not supported in first fashion Could be implemented via UPES (for example, using Message Broker)	Event handler Pick activity, receive activity
Monitoring specification	Audit trail specification Audit trail via MQ or DB	Monitoring specifications: CBEs (messages/DB) and/or audit trail (DB)
Layout of process model	Layout specification	Layout is specified in BPELX file: A separate file
Manual activity start/exit	Activity specification	Modeled using preceding/succeeding inline human task activity
Expiration	Activity specification notification mode (run/hold) specified at process level	Specified for invoke activity Expiration cannot be put on hold during process suspension yet
Priority	Activity specification	Human task specification Automatic priority increase upon escalation possible Priority cannot be taken from context/variable yet
Dynamic service invocation	Not supported in first fashion except for dynamic invocation of subprocesses	Support for dynamic assignment of endpoint references to partner links Dynamic invocation of subprocesses

Staff support

Table 3-2 lists the conceptual staff support of WebSphere MQ Workflow and maps each to the equivalent staff support in WebSphere Process Server.

Table 3-2 Staff support

	WebSphere MQ Workflow	WebSphere Process Server
Human task support	Staff assignment definitions for activity and process Manual start/exit for activity	Human tasks inline as part of processes, or standalone. Different kinds of human tasks: ► Participating ► Originating ► Purely human ► Administrative tasks Ad hoc collaboration support (for example, via subtasks, follow-on tasks.
Staff directory	Integral part of WebSphere MQ Workflow Can be synchronized with external LDAP directory using WebSphere MQ Workflow LDAP bridge	Native support for external directories (for example, LDAP, local OS, custom).
Staff schema	Fixed	Support for arbitrary staff schemes. For LDAP by default IBM Tivoli Directory Server schema is used.
Staff assignment/resolution	Based on predefined WebSphere MQ Workflow staff model	Specified (in TEL) using staff verbs. Predefined set of staff verbs. Definition of custom staff verbs possible, that is, defining verbs and XSLT files for their mapping to the (custom) staff schema.
Substitution	Integral part of staff schema	Extension for VMM.

Staff schema: Sample Staff Plug-in configuration

Table 3-3 lists the conceptual staff schema support of WebSphere MQ Workflow and maps each to the equivalent staff schema support in WebSphere Process Server.

Table 3-3 Staff schema

	WebSphere MQ Workflow	WebSphere Process Server
Person	Person	Person.
Organization	Organization	Not applicable. Use department instead.
Role	Role	Groups.
Level	Level	Not applicable. Could be provided using a custom schema. Additional verb definitions needed, too.

Staff resolution

Table 3-4 lists the conceptual staff resolution support of WebSphere MQ Workflow and maps each to the equivalent staff resolution support in WebSphere Process Server.

Table 3-4 Staff resolution

	WebSphere MQ Workflow	WebSphere Process Server
Everybody	ALL	Verb Everybody
Person	PERSON PersonName | from container	Verb Users by user ID
Organization	ORGANIZATION orgName | from container including child orgs, reporting managers, members only	Verb Group Members (including child orgs, members only) Reporting managers via custom verb
Coordinator of role	COORDINATOR OF ROLE roleName | from container	Via custom verb (LDAP/VMM)
Manager of organization	MANAGER OF ORGANIZATION orgName | from container	Via custom verb (LDAP/VMM)
Member of role	MEMEBER OF ROLE roleName | from container	Verb Group Members

Level	LEVEL x [...y] from container	Via custom verb (LDAP/VMM)
Process administrator	PROCESS_ADMINISTRATOR	Verb Users by user ID, using context variables in staff verb
Activity starter	STARTER_OF_ACTIVITY	Verb Users by user ID
Manager of activity starter	MANAGER OF STARTER_OF_ACTIVITY	Verb Manager of Employee by User ID, and using context variables in staff verb
Process starter	PROCESS_STARTER	Verb Users by user ID
Manager of Process Starter	MANAGER_OF PROCESS_STARTER	Verb Manager of Employee by User ID, and using context variables in staff verb
Four eyes principle	EXCLUDE STARTER_OF_ACTIVITY	Verb Users by user ID without Named Users, and using context variables in staff verb

Standard WebSphere MQ clients

Table 3-5 lists the WebSphere MQ Workflow interfaces/clients and maps them to the equivalent WebSphere Process Server interfaces/clients. See also Chapter 6, “Planning for clients based on application programming interfaces” on page 115.

Table 3-5 Standard WebSphere MQ clients

	WebSphere MQ Workflow	WebSphere Process Server
JSP/servlet-based Web client (including custom JSPs)	Web client	BPC Explorer, based on reusable JSF components
Portlet client	Portal client	► My Task Portlet by WebSphere Portal V5.1/6.0 ► BPC JSF components supported in the Portal environment
Fat client (non-Web based)	ActiveX client	N/A
Interfaces/client for process statistics	Audit trail view/queue	► Business Process Choreographer Observer ► Audit trail views ► Common Base Events (CBEs)
Client generation	Rapid deployment wizard	► Web client generation (JSF/JSP based) ► integrated in WebSphere Integration Developer

Interfaces

Table 3-6 lists the WebSphere MQ Workflow interfaces and maps them to the equivalent WebSphere Process Server interfaces.

Table 3-6 Interfaces

	WebSphere MQ Workflow	WebSphere Process Server
Process interfaces and renderings	Generic interfaces: ► Java ► C ► C++ ► ActiveX ► COBOL/390 ► XML ► MQ ► JMS WebSphere MQ Workflow: Web Services Process Management Toolkit (WA07)	Generic interfaces: ► EJB™ ► Web Services ► JMS Specific interfaces via SCA bindings (for example, Web Services, JMS, MQ native, MQ JMS) Business Process Choreographer JSF components for user interface clients
Data handling interfaces	Container APIs	Business object/SDO interface
Database views	Audit trail, process template, process instance, activity template, program activity instance, work item	Audit trail, process template, process instance, activity template, activity, work item, and others
Staff Administration API	Yes	Not applicable → external staff directory
External authentication	Authentication exit	Not applicable → WebSphere authentication

API functionality

Table 3-7 lists the WebSphere MQ Workflow API functionality and maps each to the equivalent WebSphere Process Server functionality.

Table 3-7 API functionality

	WebSphere MQ Workflow	WebSphere Process Server
SuspendUntil process instance	Yes	Yes suspend (duration or calendar)
Set description/name of activity, of process instance	Yes	-
Process instantiation	Create, Start CreateAndStart	- Initiate, call, sendMessage
Process instance monitoring API	Yes	-

Application/service integration

Table 3-8 lists the WebSphere MQ Workflow application/service integration and maps each to the equivalent WebSphere Process Server application/service integration.

Table 3-8 Application/service integration

	WebSphere MQ Workflow	WebSphere Process Server
Invocation model	Via UPES/XML (JMS/MQ), UPES Framework	SCA as the common invocation model Writing of SCA components (such as human tasks, business process, business rules), various SCA export/Import bindings (WS, MQ, MQ JMS) to interact with the outside, WebSphere Adapters, WebSphere Enterprise Service Bus
Invocation on z/OS: CICS & IMS Invocation	PES on z/OS	JCA adapters for CICS and IMS, or via Web Services interface
Web service invocation	SupportPac WA07: Web services	Full Web services support
Java invocation	Via UPES/UPES Framework (see SupportPac WA05) or by Program Execution Agent	Native support of Java
Client-side application Invocation	Programexecution agent	-

Tooling

Table 3-9 lists the WebSphere MQ Workflow tooling and maps each tooling environment to the equivalent one in WebSphere Process Server.

Table 3-9 Tooling

	WebSphere MQ Workflow	WebSphere Process Server
Integration development tooling	Buildtime	WebSphere Integration Developer comprises, for example, BPEL Process Editor, Human Task Editor, Assembly Editor
Application development tooling	Rapid Application Development (RAD)	WebSphere Integration Developer, RAD
Client generation	Rapid Deployment Wizard (SupportPac)	Web client generation (JSF/JSP based), integrated in WebSphere Integration Developer
Test environment	Development/test environment via dedicated set of runtime servers	Test environment integrated into WebSphere Integration Developer: WebSphere Test Environment Test client allows starting SCA service components and simulating services Debugging capabilities for step-by-step execution
Source artifacts	FDL	SCDLs, WSDLs/XSDs, BPELs, TELs, and others for additional component types

Supported environments

Table 3-10 lists the supported WebSphere MQ Workflow environments and those supported in WebSphere Process Server.

Table 3-10 Supported environments

	WebSphere MQ Workflow	WebSphere Process Server
Supported platforms	Windows, AIX®, Sun™ Solaris™, HP-UX, z/OS, Linux/iSeries (Client APIs only)	Windows, AIX, Sun Solaris, HP-UX, z/OS, Linux®, i5/OS®
Supported databases	DB2®, Oracle Database and server must be on the same operating system	DB2, Oracle Cloudscape, Informix®, SQL Server® Database and server may be on different operating systems
Server environment	Standalone server Clustering possible	Enterprise application running on WebSphere Application Server Network Deployment V6 Clustering support for WebSphere Process Server via WebSphere Application Server Network Deployment
Messaging infrastructure	 WebSphere MQ	WebSphere Platform Messaging WebSphere MQ connectivity via various SCA MQ bindings

3.2.2 Determining WebSphere Process Server enhanced functionality

WebSphere Process Server offers enhanced functionality that was not available in a WebSphere MQ Workflow environment:

- It natively supports the latest Internet standards for process execution.
- It supports open standards for defining business processes (BPEL), business objects (SDO), and invocation model (SCA).
- It is part of a converged SOA platform for process and application integration and Java application serving. A single integrated platform supports a vast array of features needed for business integration including process choreography, business state machines, business rules, Enterprise Service Bus (ESB), Common Event Infrastructure (CEI), adapters, and custom Java programs.

- It supports sub-tasks, follow-on tasks, and ad-hoc insertion of additional tasks.
- It supports undo operations of a process or parts of a process via compensation.

In addition, WebSphere Process Server has the following advantages:

- One programming model
- One integrated platform
- One integrated development environment

As well as enhanced functionality:

- Combines process integration capabilities
- Single platform for all types of business processes
- Wiring WS-BPEL processes with service components (business rules, human tasks, event state transitions, mediations, business objects, selectors, relationships) for composite process integration solutions
- Tooling: WebSphere Integration Developer

For more information about the benefits of WebSphere Process Server, refer to the WebSphere Process Server V6.1 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm

3.3 Features mapping and migration approach

Table 3-11 lists the major conceptual features of WebSphere MQ Workflow, maps them to equivalent concepts in WebSphere Process Server, and provides a strategy for transition.

Table 3-11 Features comparison and transition approach

WebSphere MQ Workflow concept	WebSphere Process Server equivalent	How to transition/references available
Business processes are defined via MQ Workflow Definition Language (FDL). WebSphere MQ Workflow V3.6 uses FDL.	Business processes are defined via BPEL. WebSphere Process Server V6.1 natively supports WS-BPEL 2.0.	Use FDL2BPEL Conversion tool. Additional manual steps to overcome limitations and optimize the model.

WebSphere MQ Workflow concept	WebSphere Process Server equivalent	How to transition/references available
In-transit data is defined via data structures and data containers.	Business Objects and variables that are based on Service Data Objects. Data definition via XML Schema Definition (XSD).	Use FDL2BPEL Conversion tool. Additional manual steps to overcome limitations and optimize the model.
Queue-based invocation model for external programs: User-defined program execution server (UPES).	JMS/MQ bindings can be used wherever queue connectivity is needed.	For WebSphere MQ Workflow messages of type ActivityImplInvoke, ActivityImplInvokeResponse, and GeneralError use FDL2BPEL Conversion tool. For other types of WebSphere MQ Workflow messages, manual modifications are necessary. The UPES Implementation programs must be reconfigured so that the reply is not sent to WebSphere MQ Workflow EXEXMLINPUTQ queue. Instead, for every UPES a separate response queue would need to be created. Also, the UPES implementation must handle the correlation IDs differently. Instead of the correlation ID being part of the message payload. WebSphere Process Server expects the correlation ID in MQMD or RFH2 headers.
Invocation of non-UPES external programs (PEA and PES).	No direct equivalence.	Not possible. Design change recommended.

WebSphere MQ Workflow concept	WebSphere Process Server equivalent	How to transition/references available
Clients: ► Default standard client available with product ► Default thin client available with product that uses WebSphere MQ Workflow Java API Custom client that uses WebSphere MQ Workflow Java API. Clients connecting using XML messages (non-UPES). Portal-based client shipped with product.	► None ► Default thin client (BPC Explorer) available with product that uses BPC Java API Custom client that uses Business Process Choreographer Java API. None. None.	► Not possible. ► Use the Business Process Choreographer Explorer. Use API documentation to manually change the WebSphere MQ Workflow Java API calls to equivalent Business Process Choreographer Java API calls. Not possible. Implement Portal client using Business Process Choreographer Java APIs.
Control and data flow.	Yes.	Use FDL2BPEL Conversion tool. Additional manual steps to overcome limitations and optimize the model.
Audit log database.	Yes.	Manually configure settings.
Expiration.	Yes.	Use the FDL2BPEL Conversion tool. Additional manual steps to add a fault handler where the timeout exception is handled.
Notification.	Partial.	Use the FDL2BPEL Conversion tool. Only human task activity notification can be migrated.
Process administration.	Yes.	Use the Business Process Choreographer Explorer.

WebSphere MQ Workflow concept	WebSphere Process Server equivalent	How to transition/references available
Exception handling.	Yes.	Use Fault Handlers, Compensation Handlers, and Failed Event Manager.
Internal staff database.	None. Uses a Staff Plugin Provider, which can access staffing data from LDAP files, the local operating system user list, or from a custom staff provider.	If the internal staff database of WebSphere MQ Workflow was populated using LDAP bridge, then configure WebSphere Process Server to access the source LDAP director.y If the Internal Staff Database of WebSphere MQ Workflow was manually created, then the data needs to be manually put into a custom user registry or an LDAP directory for use by WebSphere Process Server.

WebSphere MQ Workflow concept	WebSphere Process Server equivalent	How to transition/references available
Dynamic staffing/staff resolution.	Partial.	The FDL2BPEL Conversion tool translates WebSphere MQ Workflow staff queries to Task Execution Language (TEL) default staff queries as specified in the staffVerb.xml. Use TEL documentation to manually map complex WebSphere MQ Workflow staff query to customized TEL staff verbs. However, not all WebSphere MQ Workflow staff resolution behaviors can be achieved via TEL staff queries because of the following differences: ► TEL allows staff assignment for staff activities only, which are exclusively human tasks. ► Supports ad hoc human tasks. ► TEL staff queries cannot inherit from other contexts. ► TEL staff resolution is always based on a single query. ► Staff resolution strategies cannot be mapped to TEL (except for *include child organizations* and *organization members only*). ► Level is an unknown concept in TEL.
Global data container.	Yes, it is possible to query process level variables.	Use the FDL2BPEL Conversion tool.
Dynamic invocation of sub-process.	Yes.	Use the FDL2BPEL Conversion tool.

3.4 Fault and exception handling in business processes

During modeling of a process, faults and exceptions must be considered and, if not handled, can lead to unexpected conditions or results. There are a number of ways of dealing with potential faults and exceptions in business processes and tasks.

In WebSphere MQ Workflow the only method of handling unexpected faults and exceptions is via a return code. When modeling transition conditions and activity exits, a return code is set and then surrounding solutions can be built.

As an enhancement over WebSphere MQ Workflow, WebSphere Process Server offers value-add built-in capabilities such as compensation and fault/event handling.

Details on the specific topics can be found in the IWebSphere Integration Developer V6.1 information center. Refer to Chapter 12, “Implementing clients based on application programming interfaces” on page 275, for further information.

3.4.1 Continue on error setting

The *continue on error* setting determines how the process should proceed when a fault (on either an invoke or a snippet activity) is not caught on the enclosing scope or handled through a local fault handler.

When continue on error is checked, any faults that occur are sent to the fault handler of the scope enclosing the activity. If that fault handler cannot deal with the fault, it is thrown back to the next higher level. If the fault reaches the outermost scope and it is still not handled, the process terminates. This behavior is different from WebSphere MQ Workflow.

If continue on error is unchecked, the behavior is much closer to WebSphere MQ Workflow. If a fault is thrown and there is no fault handler for the enclosing scope, then the process transitions to a stopped state and a work item is created for the process administrator. For details see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/tasks/cconerr.html

For more detailed information see Chapter 9, “Business Process conversion” on page 187.

3.4.2 Business process compensation

Compensation halts the execution of a process and redresses the operations designed to be compensated that have already taken place.

Compensation has at times been described as a means of undoing an action, but this is not necessarily accurate. More specifically, it is a service that is executed when a state is reached in your process that you have deemed to be undesirable. The goal is not always to return to a previous condition, but instead to maintain a balanced and consistent state and to compensate for any committed operations that conflict with this state.

There are two types of compensation:

- Business compensation, which occurs outside of a transaction
- Technical compensation that occurs within

For more information refer to the section “Fault handling and compensation handling in business processes” in the WebSphere Process Server V6.1information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm

4

Planning for human interaction in business processes

For systems like WebSphere MQ Workflow or WebSphere Process Server, the people working with the system must be defined and administered appropriately.

At first, for both products, an introduction to the following areas is discussed:

- WebSphere MQ Workflow staff architecture
  - Staff mode

    WebSphere MQ Workflow and WebSphere Process Server have different capabilities to store and process staff-related information.
  - Authorization

    In WebSphere MQ Workflow and WebSphere Process Server, it is possible to define what each user is allowed to do in the system on a detailed level.
  - Staff resolution
- WebSphere Process Server people assignment

  A core function of WebSphere MQ Workflow and WebSphere Process Server is the assignment of work items or tasks according to rules defined in the

process model. For this, not only the available people must be known to the system but also properties of these people that serve as criteria for the staff resolution. For example, if a work item or task should be assigned to all members of a specific department, the members of the department must be known to the system.

- Map WebSphere MQ Workflow concepts to WebSphere Process Server

  The mapping of staff-related information from WebSphere MQ Workflow to WebSphere Process Server.

4.1 WebSphere MQ Workflow staff architecture

This section introduces the WebSphere MQ Workflow staff architecture. Special attention is given to aspects where the WebSphere MQ Workflow architecture differs from the WebSphere Process Server architecture.

4.1.1 People interaction with activities in WebSphere MQ Workflow

In WebSphere MQ Workflow, depending on the attributes of a program or process activity, the activity is designed for interaction with a person, or not. Typically, if the activity is started manually, it could be considered as a staff activity. If it is started automatically, the activity is performed without interaction with a person. In the latter case, often staff resolution is defined in a way that the activity is performed, for example, on behalf of the process administrator.

4.1.2 The fixed, built-in staff repository of WebSphere MQ Workflow

In WebSphere MQ Workflow, information about people is held in the Runtime database. The available attributes of people, organizations, and roles are built-in and cannot be extended by the user. Figure 4-1 shows the available relations between these types.

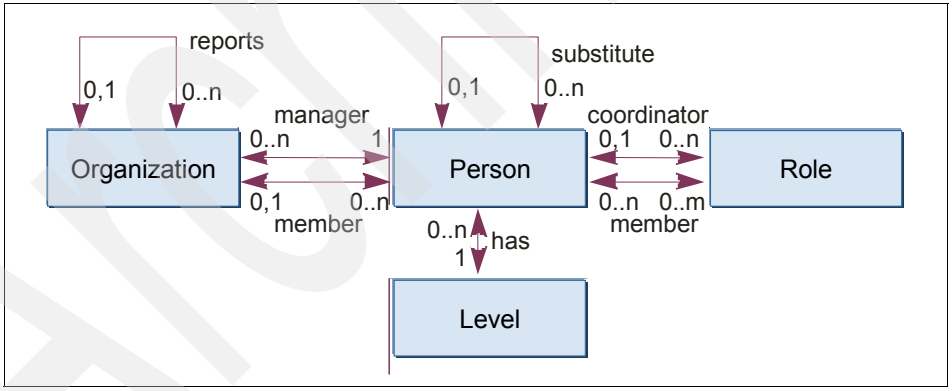


Figure 4-1 WebSphere MQ Workflow staff schema

All these relationships can be used to define staff resolution definitions.

In contrast to WebSphere Process Server, this staff model cannot be adapted to customer needs and cannot be easily integrated into other applications.

4.1.3 Tools to access and modify the WebSphere MQ Workflow staff repository

To query and update the staff repository, a variety of tools is available:

- MQ Workflow Definition Language (FDL) import/export
- WebSphere MQ Workflow client API
- WebSphere MQ Workflow Staff Administration API
- Lightweight Directory Access Protocol (LDAP) Bridge

MQ Workflow Definition Language import/export

FDL is a file exchange format to export data from and import data into a WebSphere MQ Workflow system, as shown in Figure 4-2. With the Runtime import/export tool fmcibie, you can import and export staff-related data. Figure 4-2 shows the WebSphere MQ Workflow FDL import and export.

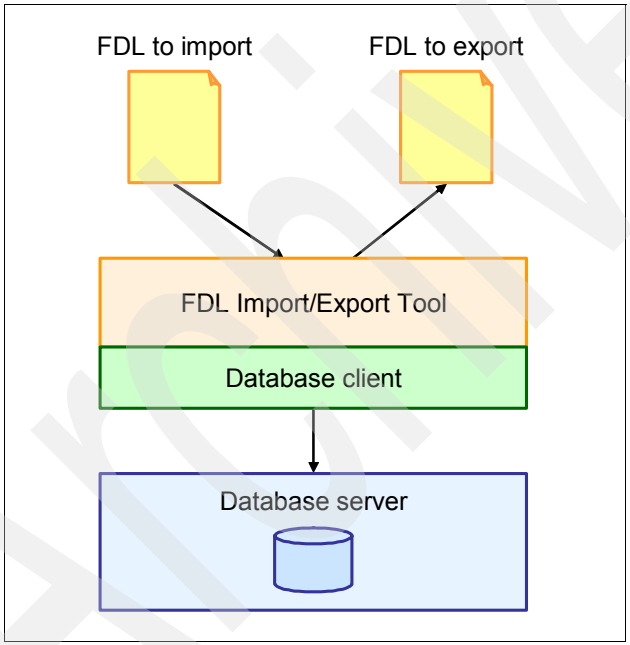


Figure 4-2 WebSphere MQ Workflow FDL import/export

For further information about FDL, refer to the publication *IBM WebSphere MQ Workflow Getting Started with Buildtime,* SH12-6286.

WebSphere MQ Workflow client API

In addition to the ability to log on and log off the WebSphere MQ Workflow APIs, a limited set of other capabilities is available:

- Setting the substitute and absent flag of a person
- Reading the attributes of a person record

Figure 4-3 shows the WebSphere MQ Workflow client API.

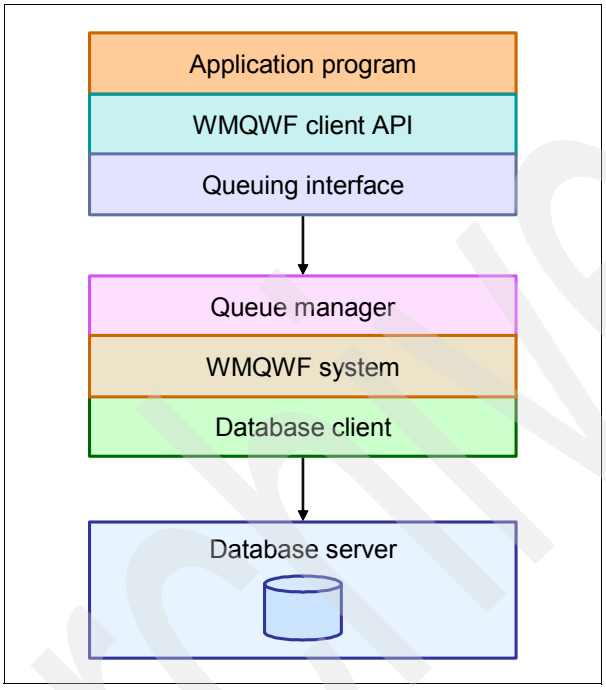


Figure 4-3 WebSphere MQ Workflow client API

For further information about the client API, refer to the *IBM WebSphere MQ Workflow Programming Guide*, SH12-6291.

WebSphere MQ Workflow Staff Administration API

For a programmatic access of the staff repository, this API allows reading and writing of staff-related data to an extent similar to FDL import and export. The API is optimized for performance even when you are processing large amounts of data.

Figure 4-4 shows the WebSphere MQ Workflow Staff Administration API.

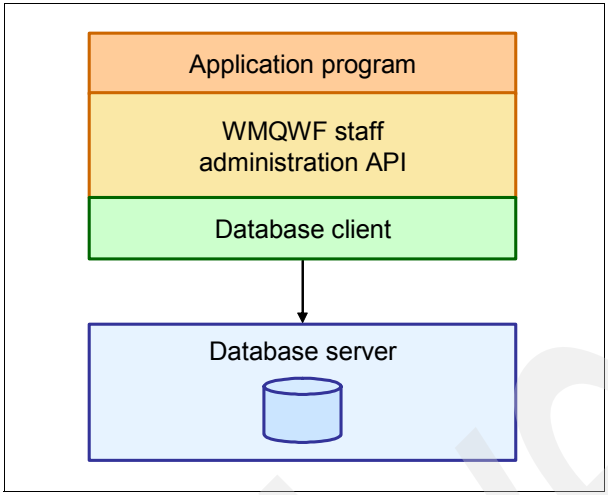


Figure 4-4 WebSphere MQ Workflow Staff Administration API

For further information about the Staff Administration API, refer to the *IBM WebSphere MQ Workflow Programming Guide*, SH12-6291.

LDAP Bridge

The primary usage of the LDAP Bridge is to access an external LDAP staff repository to update the WebSphere MQ Workflow staff repository. The intention is to use this tool, for example, each night to propagate into the WebSphere MQ Workflow staff repository changes made in the LDAP directory.

The LDAP Bridge can use various interfaces to read and write staff information, as shown in Figure 4-5. To access WebSphere MQ Workflow staff, the Staff Administration API and FDL format can be used. To access the LDAP directory, the client interface via Java Naming and Directory Interface (JNDI) and the LDAP directory information format (LDIF) interface can be used. For further information about the LDAP Bridge, refer to the *IBM WebSphere MQ Workflow Administration Guide*, SH12-6289.

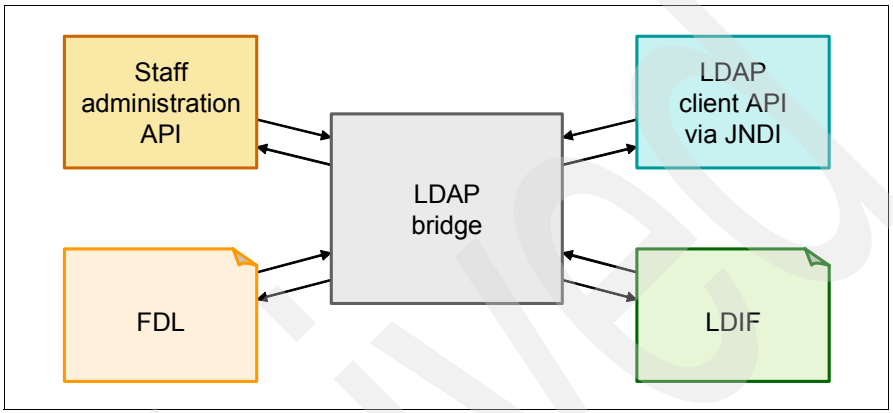


Figure 4-5 WebSphere MQ Workflow LDAP Bridge

4.1.4 WebSphere MQ Workflow authorization concepts

In WebSphere MQ Workflow, authorization and staff resolution are different concepts. In WebSphere MQ Workflow, authorizations can be granted in two ways:

- Explicitly

  The persons record contains a set of authorization attributes that can be selected or deselected. For example, the person who is assigned as process administrator of a process instance has the rights necessary to be able to administer this process instance.

- Implicitly

  Depending on previous actions, a person can have certain rights. For example, the person who is assigned as process administrator has the rights necessary to be able to administer the process instance.

The various authorization attributes are not explained here. For an explanation of the authorization settings available, refer to the *ÌBM WebSphere MQ Workflow Programming Guide,* SH12-6291. For more details on the FDL definitions of authorization attributes, refer to the *IBM WebSphere MQ Workflow Getting Started with Buildtime*, SH12-6286.

4.1.5 WebSphere MQ Workflow staff resolution concepts

Staff resolution is the process of selecting a group of people for a specific purpose according to a set of rules that have been defined earlier. In WebSphere MQ Workflow, staff resolution occurs in the following places:

- The person who is assigned as process administrator of a process instance, for example, has the rights necessary to be able to administer this process instance. The concept of a staff activity is not known in WebSphere MQ Workflow. Instead, for each program and process activity, staff resolution can be defined. In case of automatic activities, often the process administrator is assigned the activities work item. In contrast to WebSphere Process Server, after an activity got ready and staff resolution has been performed, the set of people assigned to this activity remains unchanged unless actions such as transfer item or delete item are performed.
- In WebSphere MQ Workflow, program activities and process activities have the same staff resolution capabilities. A direct equivalent for the staff resolution of process activities does not exist in WebSphere Process Server. Since the FDL2BPEL Conversion tool cannot convert this, you must model this explicitly in WebSphere Process Server. Two cases must be considered:
  - In WebSphere MQ Workflow, the process activity is started manually. In this case, before the invoke activity, a staff activity should be placed with the correct staff resolution.
  - In WebSphere MQ Workflow, the starter of a subprocess can be assigned the process administrator role of this subprocess. In WebSphere Process Server, the starter of the subprocess is the authority that invoked the subprocess. This is by default the starter of the parent process.
- For program and process activities and also for process instances, maximum time intervals for execution can be defined. If such a time interval is exceeded, a *notification item* is created as a reminder. The set of people getting notification items can correlate with the work item staff resolution of its activity. For example, if roles or organizations have been specified for the work item staff resolution, the notification items can be sent to the role coordinators or managers of organizations involved. In WebSphere Process Server, this concept is called escalation.
- When a new process is created, a process administrator is defined. This is a person who should feel responsible for the process instance. For example, if a staff resolution initially returns no person, then the process administrator receives the work item as a last resort.

Staff resolution definitions can be found on the following places in the WebSphere MQ Workflow Buildtime:

- The activity's Staff1 and Staff2 pages. This is the main control point for staff resolution.
- The staff page of the process settings. Here organizations and roles can be set that could influence the activity's staff resolution.
- Policy settings that can be set in both the network (in the system group, for example) and in the process and contained activities. An inheritance schema defines how these settings are evaluated.

Notification settings can be defined at the following places in the WebSphere MQ Workflow Buildtime:

- The notification page of an activity.
- The control page of a process.

The staff resolution for process administrator is set in the process properties in WebSphere MQ Workflow Buildtime.

Frequently used concepts are the so-called group work lists. Figure 4-6 shows a WebSphere MQ Workflow group work list.

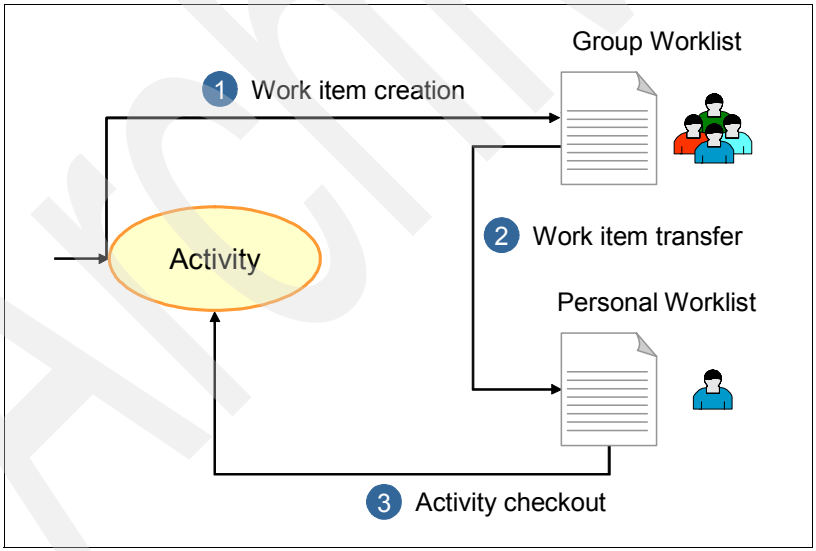


Figure 4-6 WebSphere MQ Workflow group work list

Group work lists can be characterized as follows:

- They are used especially if the number of work items per activity is large, or if the set of people who should get work items is highly dynamic.
- Instead of assigning work items to a group of people, a special person record is defined that represents this group, a so-called virtual user. Then, with staff resolution, only one work item is created for the virtual user.
- All members of this group are allowed to see the work items of this special person. After a member has selected a work item, the work item is transferred to this person and can then be checked out as usual.

For further information see *IBM WebSphere MQ Workflow Getting Started with Buildtime*, SH12-6286.

4.2 WebSphere Process Server people assignment

The human interaction for WebSphere Process Server is coordinated by its Human Task Manager, a component of Business Process Choreographer. Humans interact with the business process using human tasks.

4.2.1 Human tasks

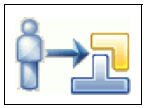
Human tasks support people when they interact with Web services and business processes. To-do tasks represent a to-do for a person or for a group of people. Human task activities of a business process represent to-do tasks. Furthermore, to-do tasks can be used outside the context of a business process to involve people in service-based applications.

To-do tasks enable people to perform some work in the context of a human task activity of a business process, or as an independent task. In the former case, to-do tasks are modeled as part of a business process and are created and started automatically by a process engine.

Task modeling includes the definition of people assignments for various task roles. The *potential creators* role enables users to create tasks. For tasks initiated by a process engine, this role is not used. The *potential owners* role grants working access to the task. One of the potential owners claims the task and becomes the task owner. The task owner performs some work on the task, potentially saves it, and completes it when finished.

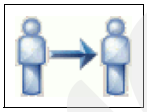
After task completion, the business process engine continues navigating the process. In addition to the two roles mentioned above, a to-do task supports the following roles:

- administrator, which can perform any permitted action on the task
- Reader, which can see all task data, but cannot change it
- Editor, which can work on a task, but cannot claim and complete it.



Invocation tasks support people when they want to invoke a (Web) service like retrieving the stock price of a company or invoking a business process that is exposed as a service. They deal with user authorization, invocation, and result-handling for the associated service. Via GUI clients of the Human Task Manager they provide business users with a comfortable interface to a Web service.

An invocation task is associated with various roles. People associated with the potential creators and potential starters roles can create the task and start it, respectively. The person creating the task becomes the task originator and the person starting the task becomes the task starter. When the task is started, it automatically invokes the service and waits for its result. When the service result is available, the invocation task stores it and the starter can retrieve it.



Collaboration tasks are intended for communication between people and are thus created and started by a human being who requests that this piece of work be done. This person becomes the originator of the task. Once started, a collaboration task behaves like a to-do task, but returns its result to the task originator instead of a client application.

Administration tasks support people in administering business processes and their activities. You can model administration tasks for three purposes in a business process. An overall administration task defines authorization for the

business process. A default administration task can be modeled to administer process activities.

This task model applies for all activities that need an administration task and that have no specific administration task assigned. You can also specify an explicit administration task for each invoke and snippet activity. If no administration task model is available, a default administration task is created ad hoc whenever one is needed by the business process.

All four kinds of human tasks support escalations that correspond to the WebSphere MQ Workflow notifications. You can model an unlimited set of escalations for a human task, escalating various actions like claiming a task in time, completing it in time, or that the invoked Web service returns in time. Possible escalation actions are the creation of work items for the escalation receivers (that is, they get access to the escalated task), sending e-mails, or sending custom events. When a task is escalated, its priority can be increased automatically.

Note that besides the task kind, Human Task Manager distinguishes between additional categories of human tasks:

- It supports inline tasks that are tightly coupled with the business process, as well as stand-alone tasks that follow the SOA patterns and are loosely coupled components.
- Human tasks cannot only be used as single tasks, you can also create an entire collaboration network using human tasks that you use as subtasks or follow-on tasks of the originally modeled human task.
- It also supports both tasks created using templates modeled in the WebSphere Integration Developer, as well as tasks defined and created ad hoc using runtime API methods.

For more details on inline, stand-alone, subtasks, follow-on, and ad hoc tasks, see the WebSphere Process Server Version 6.1 information center topics “Stand-alone and inline tasks,” “Subtasks,” “Follow-on tasks,” and “Creating task templates and task instances at runtime.”

4.2.2 Human Task Manager people resolution

Human Task Manager provides open access to various kinds of people directories. Unlike WebSphere MQ Workflow, it does not provide a people repository of its own, but leverages WebSphere Virtual Member Manager (VMM), the federated user repository within WebSphere Application Server, to interact with external directories like LDAP, operating system user repositories, or even proprietary custom repositories.

Repositories

Human Task Manager accesses the external people directory using the people assignment service that delegates people resolution to one of the people resolution plug-ins. These plug-ins are represented as the people directory provider in the runtime configuration. For each provider, Human Task Manager ships with a default people directory configuration that you can use out-of-the-box, or create your own customized configuration. Figure 4-7 shows the supported people resolution.

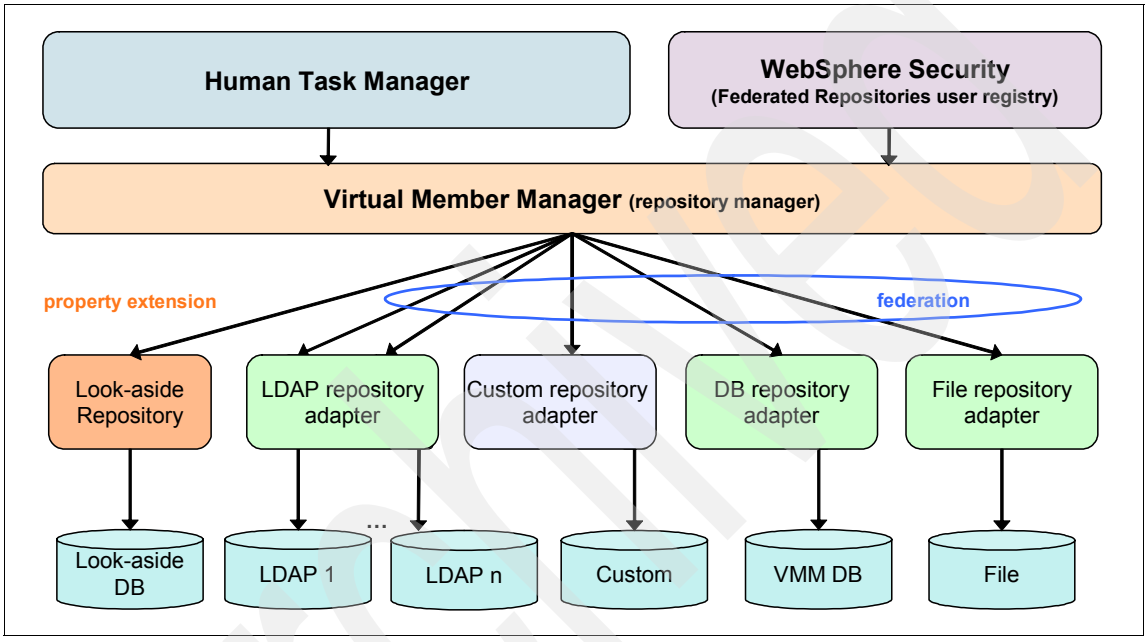


Figure 4-7 WebSphere Process Server Human Task Manager: People Directory Service

You can access the local operating system people management repository using the user registry people directory provider. An LDAP directory can be plugged in using either the LDAP provider or the VMM provider when Virtual Member Manager is configured for LDAP. To access your custom directory, we recommend implementing the RepositoryAdapter service provider interface for Virtual Member Manager. You can also store your people information in the VMM database. For test purposes, you can also use the VMM file repository or the system people directory provider.

For production environments, we recommend using the Virtual Member Manager people directory provider and configuring VMM for LDAP, or in case you do not exploit the substitution feature, you can also use the LDAP provider and access your LDAP directory directly.

Using the VMM people directory provider is the default way for people resolution in Human Task Manager. It enables access to one or more LDAP directories, database-based directories, or custom implementations of the Repository Adapter of the VMM interface. For test purposes, you can also use the built-in VMM file repository. The VMM people directory provider is a prerequisite to employing the Human Task Manager people substitution feature.

In addition to the VMM people directory provider other people directory providers exist. The user registry people directory provider offers access to the WebSphere user registry, which itself can be configured to access an LDAP directory, the local operating system repository, or a custom registry implementation. The WebSphere user registry interface does not include support for attributes other than the user ID. Human Task Manager features such as user people substitution or e-mail notification for escalations do not work with this people directory provider.

For details on people resolution and other aspects of people assignment in Human Task Manager, refer to the WebSphere Process Server information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/ctaskassign.html

4.2.3 Tools and APIs

Since Human Task Manager provides access to various people directories, it does not provide an API or user interface to manage users and groups. The administration of people information is performed using the native administration components of the people directory configured for Human Task Manager.

- When using Virtual Member Manager (VMM), you will either use the administration tools of the people directory configured for VMM or you can administer the users and groups using the tools recommended by VMM. See the WebSphere Application Server information center for more details about the VMM administration and configuration.

  When using LDAP underneath, you can use the WebSphere administrative console to perform basic user and group management and the tools provided by the respective LDAP supplier for more sophisticated operations.
- For an LDAP directory, you use the LDAP client of your directory server supplier or an open LDAP client.
- For the local operating system, you manage your users and groups with the operating system tools. When using a custom people directory, you can best administer it using its native administration interfaces.

- You can manage the Human Task Manager people directory configurations in the integrated solutions console under Resources → People Directory Provider → select provider of your choice → People directory configuration.

For details on how to manage people directory configurations, refer to the WebSphere Process Server V6.1 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/ctaskassign.html

Since Human Task Manager does not manage people information itself, it is not notified about changes in the underlying people directory. In order to keep its authorization management information up-to-date, Human Task Manager provides the people assignment refresh daemon that refreshes expired people query results on a periodic basis.

For its substitution feature, Human Task Manager provides a set of API methods to administer substitution data, and the Business Process Choreographer Explorer provides a graphical user interface based on this API. You can maintain the list of substitutes for a person using the substitutes' user IDs using the setSubstitutes() API method.

You can administer the following substitution data based on these APIs:

- You can maintain the list of substitutes for a person using the substitutes user IDs using the setSubstitutes() API method.
- You can maintain the absence flag of a user using the setAbsence() API method.

By default, users can only maintain their own absence and substitutes list. Only administrators are allowed to administer everybody's absence and substitutes. But by changing the value of the *restrict substitute management to administrators* Human Task Container flag to false in the integrated solutions console, you can allow every user to manage the absence and substitution for others as well. See the WebSphere Process Server information center for further details on the substitution feature.

To test the people assignment criteria you have modeled with WebSphere Integration Developer, a new feature has been added with Version 6.1. This feature allows for each people assignment criteria modeled in WebSphere Integration Developer to execute it against any of the WebSphere Process Server servers configured in WebSphere Integration Developer.

4.2.4 Authorization concepts

Business Process Choreographer is embedded in the security context of the WebSphere Application Server and thus leverages many of the application server security concepts.

Figure 4-8 shows how the Business Process Choreographer components integrate with WebSphere security.

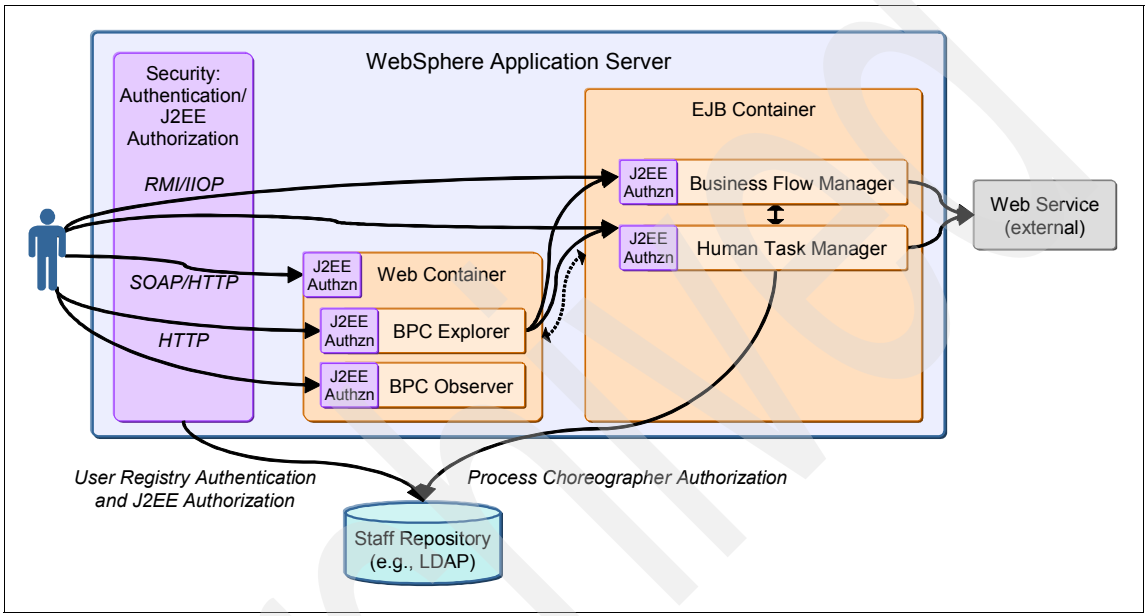


Figure 4-8 Business Process Choreographer components integrated with WebSphere security

User authentication is performed by the WebSphere Application Server Web container or EJB container, depending on the method used to invoke the API. Human Task Manager also exploits the subject information established by WebSphere authentication for its instance-based authorization.

Authorization for Business Process Choreographer is twofold. It relies on the J2EE authorization performed by the application server using a set of J2EE roles, as well as performing a sophisticated instance-based authorization that is able to enforce authorization rules that can consider the business context of a process or human task. The instance-based authorization is performed by Human Task Manager for all other Business Process Choreographer components.

Note that we strongly recommend configuring the same repository for WebSphere security (user registry) as for Human Task Manager people

resolution and thus authorization. Proper authorization relies on the users and groups, as well as the user-to-group relationships to be known identically for both WebSphere security and Human Task Manager people resolution.

J2EE authorization

The J2EE authorization for Business Process Choreographer consists of a set of J2EE roles established on its various components:

- The Business Process Choreographer Explorer supports the role WebClientUser, which mainly has the purpose of enforcing authentication before accessing the explorer. It is set to all authenticated users by default, since the explorer relies on the more sophisticated authorization performed by Human Task Manager. You can also use this role to restrict the set of users allowed to access the explorer. In this case you would restrict the role from *all authenticated* to a set of groups or users.
- The Business Process Choreographer Observer supports the role ObserverUser that has the default of *all authenticated*. For the observer we recommend restricting the set of users for this role, since it does not benefit from the instance-based authorization performed by Human Task Manager.
- Business Flow Manager supports three roles for authorization:
  - BPESystemAdministrator is the overall administration role and grants its members the right to invoke every API method that is available and to see all data. This role overrides restrictions of the Human Task Manager instance-based authorization.
  - BPESystemMonitor grants its members read rights for all items (processes, activities, and so on) managed by Business Flow Manager. This role overrides restrictions of the Human Task Manager instance-based authorization.
  - BPEAPIUser is mainly used to enforce authentication for users who access the Business Flow Manager API. We recommend leaving its value as all authenticated.

  The WebClientUser and JMSAPIUser roles are technical roles not used for authorization.
- Human Task Manager supports three roles for authorization:
  - TaskSystemAdministrator is the overall administration role and grants its members the right to invoke every API method that is available, and to see all data. This role overrides restrictions of the Human Task Manager instance-based authorization.
  - TaskSystemMonitor grants its members read rights for all items (tasks, escalations, and so on) managed by Human Task Manager. This role

overrides restrictions of the Human Task Manager instance-based authorization.

- TaskAPIUser is mainly used to enforce authentication for users who access the Human Task Manager API. We recommend leaving its value as all authenticated.
- The role EscalationUser is a technical role not used for authorization.

Note that to keep API authorization consistent, we strongly recommend keeping the user-to-role mappings for the following roles identical:

- BPESystemAdministrator and TaskSystemAdministrator
- BPESystemMonitor and TaskSystemMonitor
- BPEAPIUser and TaskAPIUser

Instance-based authorization

One of the features of Business Process Choreographer is its ability to provide an instance-based authorization for business processes and human tasks. This allows you to model sophisticated authorization rules like the separation of duties (also known as the *four-eyes principle*) in review or approval processes. This is a major enhancement of the standard J2EE authorization approach, which only supports the definition of static authorization based on methods without considering the business context.

The instance-based authorization relies on people resolution being performed in order to assign people to various roles for later authorization checks. Although authorization rules are modeled for a task template, they can behave differently for each human task or business process instance based on that template, since they are able to consider business context data when assigning people to an authorization role.

The instance-based authorization is performed by Human Task Manager and is based on work items, as shown in Figure 4-9.

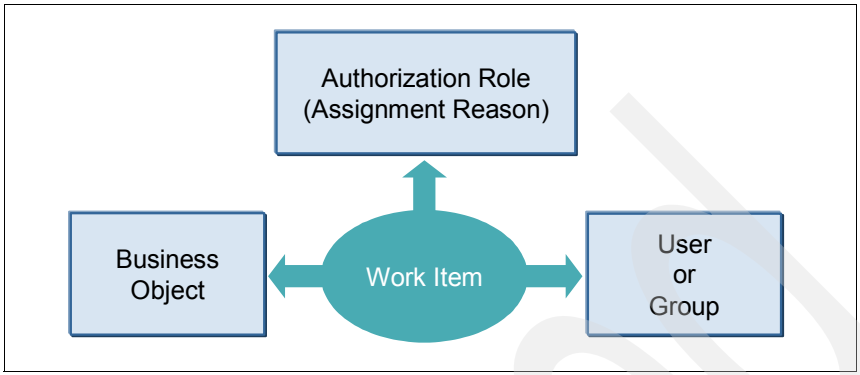


Figure 4-9 WebSphere Process Server work item relations

Work items associate users with a certain business object, like a task or process instance, and simultaneously one of the predefined authorization roles of the task or process, like reader or administrator. Each predefined role grants authorization for a fixed set of actions that a user is allowed to perform on this business entity, like claiming a task or getting the process instance data. Thus, authorization is granted by associating a person with a role on a business object.

There are three kinds of work items:

- *Everybody* work items grant authorization to every (authenticated) user associated with the work item role (assignment reason).
- *User* work items grant role authorization to the user whose user ID is associated with the work item.
- *Group* work items grant role authorization to an entire group of people. Authorization is not checked using the user ID, but by verifying the list of group memberships associated with the user's login context information.

Work items are created based on the result of people resolution, which provides the set of users or the group to authorize. They cache the people query results and thus provide a major performance advantage, since people queries do not need to be executed for every authorization check, which would cause a heavy load on the people repository in use. However, cached information has the drawback of potentially becoming outdated.

Business Process Choreographer supports refreshing the work item cache manually in the WebSphere Application Server administrative console, as well as automatically, using a people query result refresh daemon that refreshes expired results. If the original result has been modified manually using the work item

transfer functionality, a refresh of the original result considers the transfer history and does not override manual changes. Note that only user work items are refreshed, as described above.

Authorization based on group work items is refreshed automatically every time that the user logs onto WebSphere Application Server. People resolution is described in more detail in the next section.

Table 4-1 lists the authorization roles that are supported by human tasks, depending on the task kind.

Table 4-1 Authorization roles supported by human tasks

	To-do task	Invoc. task	Collab. task	Admin. task	Comment
Potential creator	X	X	X	-	People allowed to create task instances
Originator	X	X	X	-	The person who created the task
Potential owner	X	-	X	-	People who can claim and work with tasks
Owner	X	-	X	-	The person who claimed the task
Potential starter	-	X	-	-	People allowed to start a task
Starter	-	X	-	-	The person who started the task
Administrator	X	X	X	X	People allowed to administer a task
Editor	X	-	X	-	People allowed to edit task data
Reader	X	X	X	X	People allowed to see task data
Escalation receiver					People who receive an escalation

Note: The escalation receiver role does not apply to human tasks, but to their escalations.

Table 4-2 lists the roles supported by business processes.

Table 4-2 Roles supported by business processes

Role	Comment
Administrator	People allowed to administer a process
Reader	People allowed to see process data
Starter	Person who created and started the process instance

Note: The administrator of a process administration task corresponds to the process administrator on the administered process, and the administration task reader corresponds to the process reader.

Table 4-3 lists the authorization roles that are supported by process activities, depending on the kind of activity.

Table 4-3 Authorization roles supported by process activities

	Human task activity (staff activity)	Receive, receive-choice, event handler	Comment
Potential owner	X	-	People who can claim and work with tasks
Owner	X	-	The person who claimed the task
Potential starter	-	X	People allowed to start an event
Administrator	X	X	People allowed to administer a task
Editor	X	-	People allowed to edit task data
Reader	X	X	People allowed to see task data

As does WebSphere MQ Workflow, Business Process Choreographer also supports authorization inheritance. That is, the process administrator is allowed to perform administrative actions on sub-entities like activities of the process as well. The following authorization inheritance rules apply to Business Process Choreographer:

- A process administrator is administrator for every activity, inline task (including its sub-tasks and follow-on tasks), or task escalation of the process.
- A process reader is reader for every activity, inline task (including its sub-tasks and follow-on tasks), or task escalation of the process.

- Members of an inline to-do task authorization role have the same role on the corresponding human task activity.
- A potential starter of an inline invocation task is a potential starter of the associated receive or receive choice activity, or an *on event* Event Handler, too.
- A task administrator is administrator for every sub-task and follow-on task of this task, including escalations of one of these tasks.
- Members of each task role are readers for every sub-task and follow-on task of this task, including escalations of one of these tasks.
- Escalation receivers are readers for the escalated task, and for every sub-task and follow-on task of this task, including escalations of one of these tasks.

In addition to role inheritance, you also must consider the default rules that apply for people resolution in case it fails or returns empty results. For example, if people resolution for the potential owner role fails, the administrators become potential owners for this task.

For the full set of defaults, as well as further details on Business Process Choreographer authorization, see the WebSphere Process Server V6.1 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/cstaffdefaultusers.html

4.2.5 People assignment concepts

The architecture and concepts for Business Process Choreographerr people assignment are different for WebSphere MQ Workflow.

People resolution for Business Process Choreographer is performed by Human Task Manager using people directory provider plug-ins that query an external people directory like LDAP or Virtual Member Manager (VMM).

People resolution is performed to query, for the user and group, information needed to create the work items that are later used for authorization checks. Human Task Manager people resolution is based on authorization rules that are modeled as so-called people assignment criteria in WebSphere Integration Developer.

Modeling

As a task or process modeler, you create human tasks using the task editor of WebSphere Integration Developer. When you define human tasks, you select an

authorization role like *potential owner* and associate it with a people assignment criteria from the set offered by the task editor.

The criteria come with a set of parameters that must be populated with data. Parameter data can reference context variables like %wf:process.starter%. When you have finished populating the parameter values, the people assignment criteria is ready for deployment. The resulting parameterized people assignment criteria is stored as part of the human task model. Business Process Choreographer provides a default set of people assignment criteria that can be used for standard staff queries. You can extend the set of people assignment criteria to more closely match your needs.

For each task, you also specify a people directory configuration JNDI name, which refers to a people directory configuration to be used for people assignment criteria deployment and later execution of the deployed people queries at runtime. By default, the Virtual Member Manager people directory configuration is used.

If you want to exploit the Human Task Manager substitution feature that is available with WebSphere Process Server V6.1, you can also select a substitution policy per task model, which is then applied whenever people resolution is performed or refreshed for this human task.

The resulting task or process model is exported as a J2EE enterprise archive (EAR) file and can be deployed to a WebSphere Process Server installation.

Deployment

When the WebSphere Application Server administrator deploys the human task or business process model EAR file, the people resolution service transforms the parameterized people assignment criteria into plug-in-specific people queries.

When the people support service has finished transforming the parameterized people assignment criteria into a plug-in and people-repository-schema-specific query, the deployment artifacts are stored in the Business Process Choreographer database and available for people resolution at runtime.

Runtime

The actual people resolution happens at runtime, for example, to retrieve the list of potential owners of a human task when the task is started.

The authorization management retrieves the stored people query specification from the database, resolves the set of context variables referenced by this query, and invokes the people resolution service. The people resolution service selects the appropriate people resolution plug-in and delegates people resolution to it.

The people resolution plug-in invokes people directory API methods in order to retrieve lists of user IDs or a group ID.

The Human Task Manager authorization management is in charge of driving the people assignment process, which consists of steps for people resolution, people substitution, query result post-processing, and default user/inheritance rule handling. For every task role, the authorization management retrieves the stored people query specification and invokes the various people assignment steps. The resulting set of people or group is used to create user and group work items, one for each user group, respectively, enabled for a given task role.

The people resolution mechanism is described in detail in the WebSphere Process Server V6.1 information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/cstaffdefaultusers.html

4.3 Map WebSphere MQ Workflow concepts to WebSphere Process Server

This section discusses the WebSphere Process Server concepts that can be used when converting an application from WebSphere MQ Workflow to WebSphere Process Server. The following cases can occur:

- For a WebSphere MQ Workflow concept, an equivalent concept in WebSphere Process Server exists. In this case, a conversion is easy. Sometimes the naming of these concepts differs.
- For the WebSphere MQ Workflow concept, no direct equivalent concept exists in WebSphere Process Server. In this case, by looking at the use cases of these WebSphere MQ Workflow concepts, a WebSphere Process Server concept could be used instead. Often, depending on the usage of a concept in WebSphere MQ Workflow, a specific solution is chosen in WebSphere Process Server.
- Sometimes for a concept that could not easily be solved in WebSphere MQ Workflow, a more or less complicated work-around may have been implemented in WebSphere MQ Workflow. Possibly, this concept can be implemented more easily with WebSphere Process Server.

  Such work-around solutions are often specific to one customer. Therefore, the FDL2BPEL Conversion tool cannot identify them. This means that even when using the FDL2BPEL Conversion tool, you should check the resulting BPEL process in order to identify patterns that could be modeled better with WebSphere Process Server-specific constructs. To be able to do this, good knowledge of the WebSphere Process Server capabilities is required.

4.3.1 Mapping the staff model

If an LDAP directory is already used for WebSphere MQ Workflow, for example, together with the LDAP Bridge, then after connecting WebSphere Process Server to the LDAP directory, the staff directory can be used.

In the remaining chapter we assume that this is not the case. This means that staff information must be moved from the WebSphere MQ Workflow Runtime database to an LDAP directory, for example.

The WebSphere MQ Workflow staff repository is based on a schema with quite a number of attributes and relationships. Converting this schema into a staff repository schema for WebSphere Process Server would be quite a large task. In addition, the staff directory of a typical WebSphere Process Server installation is an LDAP directory with a customer-specific schema.

Therefore, a standard, out-of-the-box conversion of staff information is not possible. Instead, you should perform the following steps:

- Identify the attributes and relationships of the WebSphere MQ Workflow staff repository used by the customer and that must be converted to WebSphere Process Server.
- Decide how to map these attributes and relationships to the capabilities of WebSphere Process Server. Often, more than one alternative is possible here. You should make your decisions based on the customer situation and requirements.
- The last step is the conversion of the staff repository. This is explained in the following sections.

When comparing the WebSphere MQ Workflow staff entities and relationships with the staff entities and relationships known to WebSphere Process Server it becomes evident that WebSphere MQ Workflow explicitly considers a richer set. However, on a more abstract level, both WebSphere MQ Workflow and WebSphere Process Server determine sets of users and groups that they associate with their respective staff task roles. For defining such user sets, WebSphere MQ Workflow can specify queries based on its own entity types and relationships.

To evaluate how user data defined for WebSphere MQ Workflow can be reused in the context of WebSphere Process Server, consider the following questions:

- How can WebSphere MQ Workflow staff data be represented in the predefined user and group schema of an existing people directory?
- How can people queries in WebSphere Process Server be parameterized to delimit the correct sets of users and groups?

4.3.2 Mapping the staff repository data

After the decisions have been made about how the staff data is to be mapped from WebSphere MQ Workflow to the people directory employed by WebSphere Process Server, the data must be transferred.

Regarding the technical solution for a transition from WebSphere MQ Workflow to the people directory used by WebSphere Process Server, the most relevant question here is how many people are defined in the system. For example, it makes a difference whether you have to cope with 10 users or 10,000 users. In the first case, the people could be manually defined in the WebSphere Process Server people directory. In the latter case, tools support is essential to minimize manual steps during the conversion.

As the FDL2BPEL Conversion tool concentrates on the conversion of process models, other tools are necessary for the migration of staff-related information. In general, staff definitions are quite customer-specific. Therefore, depending on the circumstances, different approaches can be considered.

WebSphere MQ Workflow FDL export

A simple way to get the staff repository content in file format is the use of the FDL export tool *fmcibie*.

WebSphere MQ Workflow LDAP Bridge

In case an LDAP directory is used, the WebSphere MQ Workflow LDAP Bridge can be used to transfer the data from WebSphere MQ Workflow to LDAP.

The WebSphere MQ Workflow LDAP Bridge can be used to convert WebSphere MQ Workflow staff repository content into LDAP entries either by generating an LDAP Directory Interchange Format (LDIF) file or by updating the LDAP directory via the LDAP client interface. However, the capabilities of the WebSphere MQ Workflow LDAP Bridge are limited.

See 10.1, “Using the LDAP Bridge to map staff repository content from WebSphere MQ Workflow to WebSphere Process Server” on page 231, for an example.

WebSphere MQ Workflow Staff Administration API

Especially in situations were staff data is mapped by means of programs, using the WebSphere MQ Workflow Staff Administration API is appropriate. Writing a Java program using APIs to update the LDAP directory is usually a more complex task than configuring the LDAP Bridge appropriately. On the other hand, a Java program could handle special mapping requirements that could be difficult or impossible to achieve with the LDAP Bridge.

For an example, refer to 10.2, “Using APIs to map staff repository content from WebSphere MQ Workflow to WebSphere Process Server” on page 234, and 10.3, “Mapping substitution information from WebSphere MQ Workflow to WebSphere Process Server” on page 238.

The FDL2BPEL conversion

When converting process definitions, the FDL2BPEL Conversion tool also converts staff resolution and notification definitions from FDL to WebSphere Process Server definitions.

Transitioning query definitions

Query definitions must be manually converted from WebSphere MQ Workflow to WebSphere Process Server.

4.3.3 Mapping authorization information

We distinguish here between mapping of the explicit WebSphere MQ Workflow authorizations and implicit WebSphere MQ Workflow authorizations.

In WebSphere Process Server, authorizations are defined with specific assignment reasons of workitems. For one task and one person, several workitems can exist having different assignment reasons.

Note that a WebSphere MQ Workflow role corresponds to a group in WebSphere Process Server.

Mapping explicit WebSphere MQ Workflow authorizations

Figure 4-10 shows the available explicit authorization settings in WebSphere MQ Workflow Buildtime.

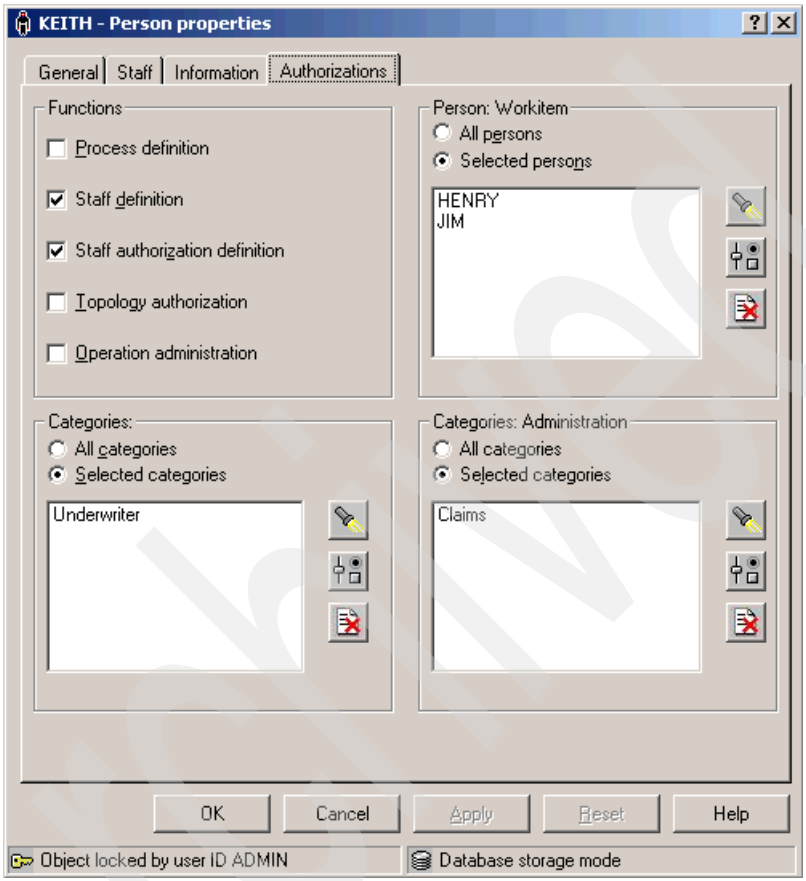


Figure 4-10 WebSphere MQ Workflow Buildtime authorization definitions

As WebSphere Process Server does not have a built-in staff repository, in WebSphere Process Server these authorization settings are mapped to authorization settings of different products:

- Process definition

  The right to import and translate new process models in the WebSphere MQ Workflow Runtime database. This is a right of WebSphere administrators. They can deploy SCA modules or J2EE applications.

- Staff definition

  This maps to the authorization to update the staff repository in WebSphere Process Server in case of LDAP to have the right to update the LDAP directory.

- Staff authorization definition

  As the WebSphere MQ Workflow authorization settings are mapped to different products, the same mapping applies to the right to administer these settings.

- Topology authorization and operation administration

  This maps to the right to do WebSphere Application Server system administration.

- Work item authorization

  In WebSphere MQ Workflow, this is the right to see work items either of all people or of specific people.

  A similar concept for the right to see work items of specific people does not exist in WebSphere Process Server. To get access to the work items of all people in WebSphere Process Server, you can use the J2EE roles BPESystemMonitor and TaskSystemMonitor, respectively, that have the right to see all process and task data.

- Authorizations for categories

  The concept of categories does not exist in WebSphere Process Server. To model something similar in WebSphere Process Server, you must assign people to your task models in the following way:

  – For a process start authorization you must assign people to an invocation task.

  – For the process administration you must assign people to a process administration task.

Mapping implicit WebSphere MQ Workflow authorizations

The following implicit WebSphere MQ Workflow authorizations exist:

- Process administrator

  In WebSphere Process Server, this is the process administrator authorization role.

- Process creator

  In WebSphere MQ Workflow the creation of a process can be done separately from starting the process. In WebSphere Process Server a similar behavior can be achieved by defining an invocation task that starts the process. This invocation task can be created and later started. When it is started, it triggers the start of the process.

- Work item owner

  In WebSphere Process Server, this is the (for example, to-do) task owner authorization role.

WebSphere Process Server supports more authorization roles than WebSphere MQ Workflow. In addition to the corresponding roles above, the authorization role *potential owner* is equivalent to the staff resolution result in WebSphere MQ Workflow.

4.3.4 Mapping staff-related definitions of process models

The FDL2BPEL Conversion tool maps most of the WebSphere MQ Workflow constructs correctly to the corresponding WebSphere Process Server constructs, but a small number of more complex constructs are not mapped. For detailed information see the documentation in SupportPac WA73, *WebSphere MQ Workflow - FDL2BPEL Conversion* available here:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

Nevertheless, the mapping is explained here so that in case you convert the process model manually you can check the conversion for completeness:

- Mapping of WebSphere MQ Workflow program activities

  With the FDL2BPEL Conversion tool, WebSphere MQ Workflow activities are converted to one of the following types:

  – Empty activity
  – Service invocation activity
  – Human task with inline to-do task

  The kind of human tasks that correspond most to the WebSphere MQ Workflow activities with staff interaction are the to-do tasks, where the

business process schedules some work for a set of humans, the potential owners. One of the potential owners claims the task and becomes the owner. Then the owner performs some work on the task and completes it when finished. After task completion, the business process engine continues navigating the process.

- Administrative access of processes and activities

  From time to time, errors can occur with process instances or activities. Often, the result is that process instances or activities are not in the correct state or variables must be changed. In WebSphere MQ Workflow, people with administration authority can initiate repair actions to correct this.

  In WebSphere Process Server, to do this it is necessary to create administration tasks for processes and activities. With these constructs you can also define who is allowed to administer the processes and activities. The FDL2BPEL Conversion tool creates administration tasks for processes, but not for activities.

- Complex staff resolution definitions

  WebSphere Process Server people queries must be defined to delimit sets of users and groups available in the people directory used by WebSphere Process Server. For semantics and parameterization of WebSphere Process Server people queries, see the WebSphere Process Server information center.

- Scope of the FDL2BPEL Conversion tool to convert staff resolution definitions

  For detailed information about which WebSphere MQ Workflow staff resolution can be mapped to WebSphere Process Server, see the FDL2BPEL Conversion documentation available here:

  http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

- Mapping of WebSphere MQ Workflow group work lists to WebSphere Process Server group work items

  In WebSphere MQ Workflow, group work lists are simply a usage pattern and not a built-in capability. You must therefore check the WebSphere MQ Workflow processes to verify whether such a construct is used. If yes, you can use the WebSphere Process Server group work item feature.

- Setting the process administrator of a subprocess

  In WebSphere MQ Workflow, a process activity can have the same staff resolution definition as a program activity. We consider this with two use cases:

  - The subprocess is started manually. In this case, in WebSphere Process Server, an inline to-do task could be inserted before the invoke activity with the appropriate staff resolution.
  - In WebSphere MQ Workflow, the subprocess starter is assigned the process administrator role of this subprocess. Again, an inline to-do task could be called before the invoke activity. The process administrator is set from an appropriate variable containing the starter of the to-do task.

- Mapping WebSphere MQ Workflow notification definitions to Human Task Manager escalation in WebSphere Process Server

  In WebSphere MQ Workflow, time limits can be defined for processes and activities. If the time to complete the process or activity reaches the limit, a notification is sent to the people responsible for the process. The staff resolution for this notification is often defined implicitly. If, for example, in WebSphere MQ Workflow an organization is involved in the staff resolution of an activity, then a notification for this activity can be sent to the manager of the organization involved.

  In WebSphere Process Server, this must be explicitly modeled. The Human Task Manager escalation concept even allows for multiple escalations in a chain and an unlimited number of escalation chains with different completion states to be monitored.

5

Planning for back-end application integration

This chapter discusses the following:

- WebSphere MQ Workflow back-end overview

  Information required to assess how back-end applications were integrated in the WebSphere MQ Workflow environment and to plan for corresponding functionality in WebSphere Process Server.

  We describe the three invocation styles available to integrate back-end implementations:

  - User-defined program execution server (UPES)
  - Program execution server (PES)
  - Program execution agent (PEA)

- WebSphere Process Server back-end overview

  To plan for a corresponding back-end integration of a WebSphere Process Server environment, we provide the following information:

  - Transition planning for user-defined program execution server implementations
  - Transition planning for program execution server implementations

5.1 WebSphere MQ Workflow back-end overview

The WebSphere MQ Workflow architecture offers three different invocation styles to integrate back-end implementations:

- User-defined program execution server (UPES)
- Program execution server (PES)
- Program execution agent (PEA)

These are represented by an *automatically starting program* activity of a process model.

All three back-end invocation styles are able to invoke and interact with external service implementations, such as updating a database or exchanging data with another system.

Figure 5-1 shows the WebSphere MQ Workflow architecture with the back-end implementation options circled.

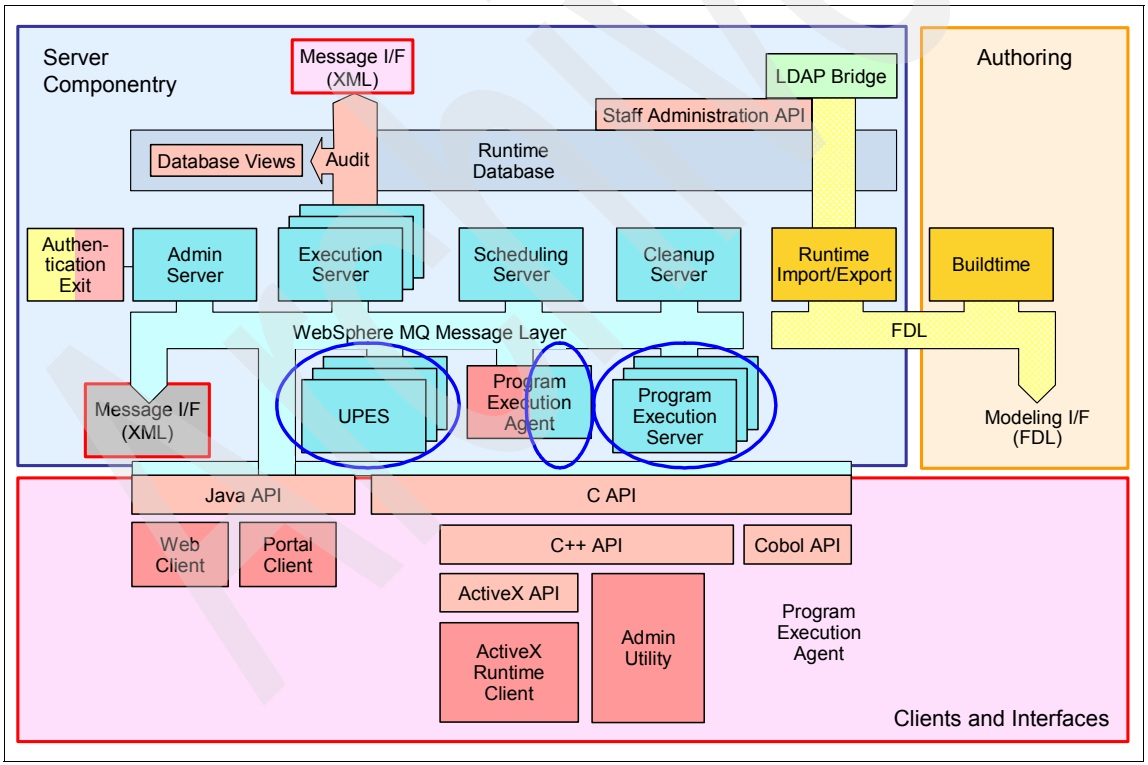


Figure 5-1 WebSphere MQ Workflow architecture with the back-end implementation options

Platform overview

Table 5-1 provides an overview of the supported platforms for the different back-end implementation options.

Table 5-1 Back-end implementation options and operating system platforms

Back-end integration options	Operation system platform	Remarks
User-defined program execution server	Any type of operating system that is supported by WebSphere MQ	Recommended as best practice
Program execution server	z/OS only	The PES provides the ability to: ► Perform transactional RPC calls to CICS/IMS. ► Perform message-based interactions with CICS/IMS. ► Reuse existing CICS/IMS programs without modification.
Program execution agent	Windows platforms supported by WebSphere MQ Workflow	A very limited back-end implementation solution. It does not scale well. It is typically used as a front-end application invocation mechanism in conjunction with human tasks.

5.1.1 User-defined program execution server (UPES)

WebSphere MQ Workflow has an XML interface to communicate with external systems. Incoming messages can make requests to the WebSphere MQ Workflow engine, while WebSphere MQ Workflow can send messages to request a program to perform an action in a user-defined program execution server implementation.

All XML messages for WebSphere MQ Workflow are carried by WebSphere MQ. Therefore, the message is actually the WebSphere MQ message header (MQMD) followed by the payload (XML-formatted data).

The UPES must be implemented as a custom program using the documented XML-based message formats that apply to it. To help implement a UPES application, MQ Workflow provides a UPES Framework library. It is a best practice to use this UPES Framework to develop a new UPES. The UPES Framework is a fully integrated and fully functional server framework.

With the UPES you can integrate any type of application and use this application to extend the management capabilities of WebSphere MQ Workflow. These applications can run on any type of operating system that is supported by WebSphere MQ. A UPES can be written in any programming language that has an interface to or is supported by WebSphere MQ, while the provided UPES Framework only supports implementations written in Java. For details on how to incorporate application programs, see the *IBM WebSphere MQ Workflow: Programming Guide,* SH12-6291, or *IBM WebSphere MQ Workflow for z/OS Programming Guide*, located at:

http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC33-7031-09

5.1.2 UPES implementations

To interface from WebSphere MQ Workflow to external systems, the WebSphere MQ Workflow API can be used within an external application. However, it is often not desirable to change an existing application. In addition, WebSphere MQ has become pervasive in the industry, which means that many applications are already MQ-enabled or have WebSphere MQ adapters available off-the-shelf.

Because of this, the WebSphere MQ Workflow XML interface is a good way to let WebSphere MQ Workflow interface with external systems. If WebSphere MQ Workflow wants to request an external program to perform work as a step of a process, it can send out an XML message to it. However, it is unlikely that the back-end application that is invoked understands the WebSphere MQ Workflow format of XML.

In this case, a custom adapter program or an add-on service, such as WebSphere MQ Integrator or WebSphere Message Broker, can be used to transform the message on-the-fly into the proper format so that the application does not have to be modified. If the request was synchronous, a reply message is sent back. Again, WebSphere MQ Integrator can transform the reply message back into the WebSphere MQ Workflow XML format.

WebSphere MQ Workflow calls such an interface a UPES. From a WebSphere MQ Workflow standpoint, a UPES is simply a queue to which the messages are sent. It is the job of the UPES implementation to read the XML messages from WebSphere MQ Workflow and deal with them.

There are three main messages that WebSphere MQ Workflow can send to a UPES:

- ActivityImplInvoke
- ActivityExpired
- TerminateProgram

UPES and load-balancing

If the WebSphere MQ Workflow environment is a clustered environment catering for availability and load-balancing, it is likely that the UPES implementation was set up to cater for these requirements as well. A UPES can run on the same machine as WebSphere MQ Workflow or elsewhere on the network. There can be one instance of it running or multiple instances. In this section we describe a few of the key configuration options.

The first option, a single UPES with multiple instances on the same box within a workflow system, allows multiple instances of the UPES to process the message, improving performance. The administrator can control how many instances are started to control the load on the system and throughput. The UPES runs on a remote server so that the workflow server has less system load.

Figure 5-2 shows the single UPES option with multiple instances on the same machine.

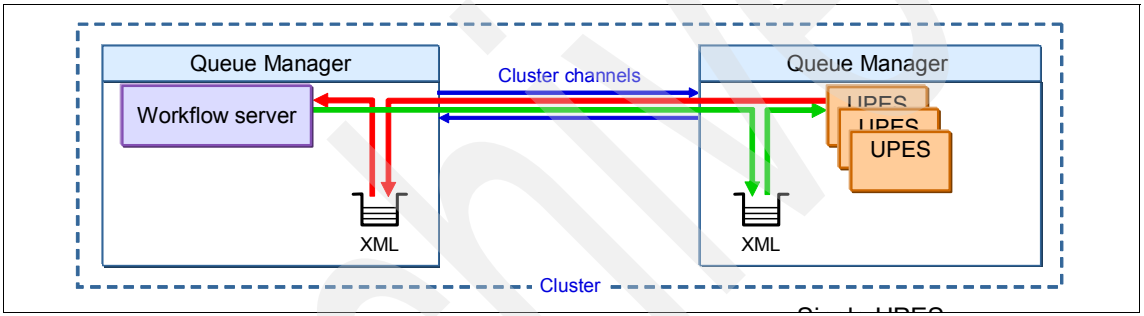


Figure 5-2 Single UPES: multiple instances on the same box

The second option is to set up two (or more) servers where the UPES implementations run within a Workflow system. This provides load-balancing, because the requests are performed by multiple servers. The queue managers are all in the same cluster.

Figure 5-3 shows the single UPES option with multiple instances on the multiple machines.

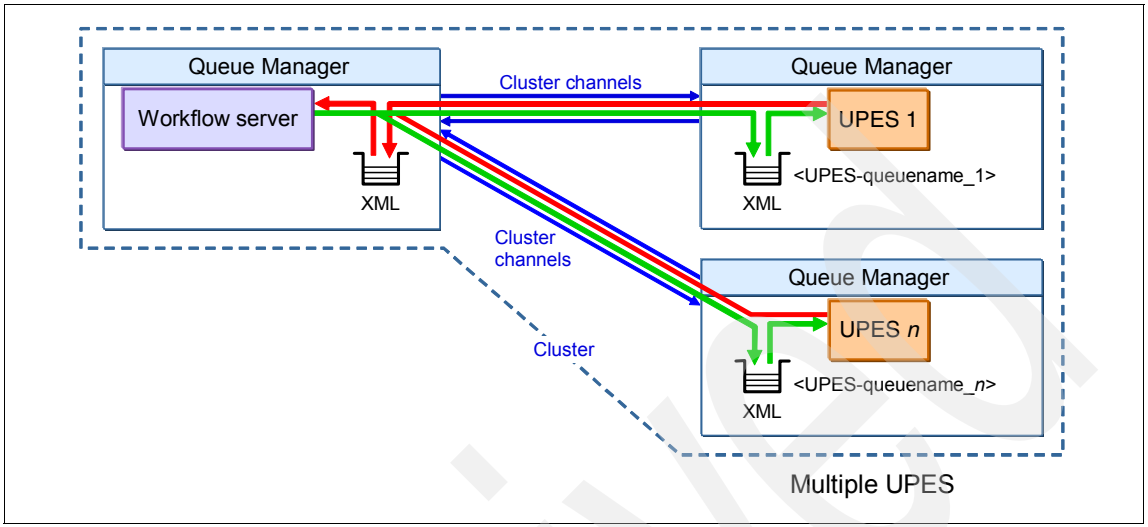


Figure 5-3 Single UPES: Multiple instances on multiple boxes

The third option is to set up multiple UPESs on a single box within a workflow system. In this configuration there is only one queue manager, but with multiple queues from which the UPES implementations read their messages. Different activities use the different queues.

Figure 5-4 shows multiple UPESs on a single machine.

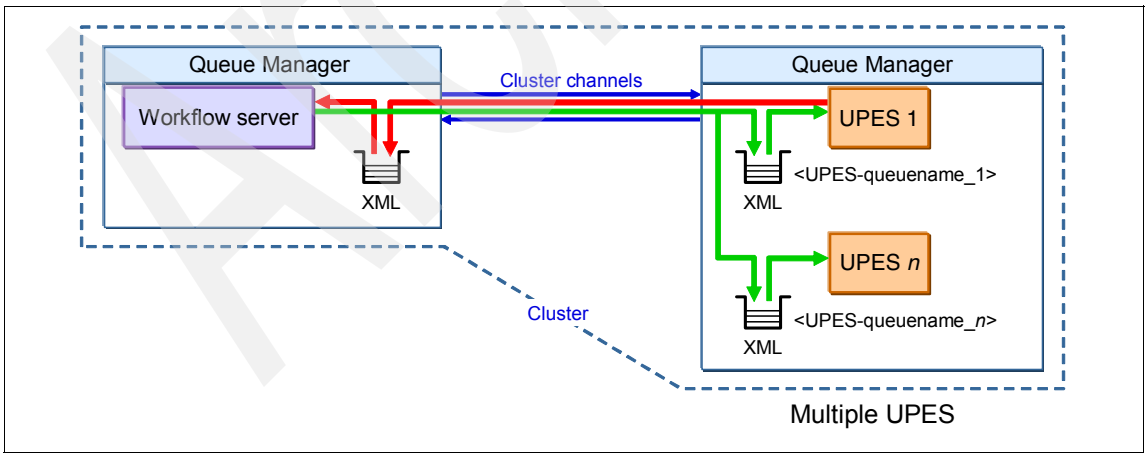


Figure 5-4 Multiple UPESs on a single box

The next scaling option is shown in Figure 5-5. It shows a configuration where each UPES implementation has its own unique queue, and each UPES implementation is on a different physical server within a workflow system. Figure 5-5 shows multiple UPESs on a multiple machines.

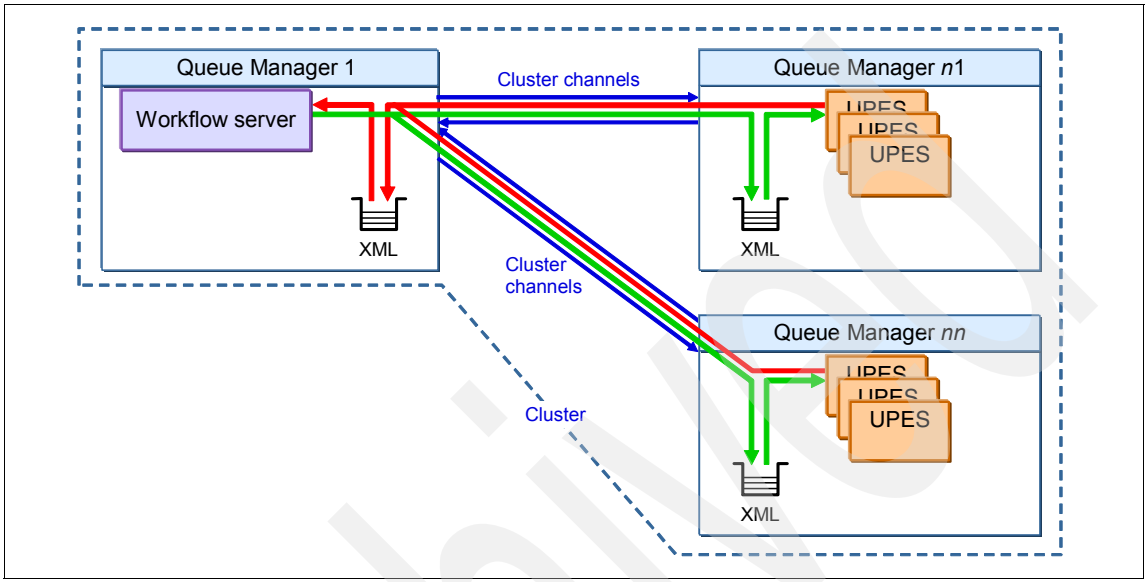


Figure 5-5 Multiple UPESs on multiple boxes

Figure 5-6 shows a fully load-balanced UPES implementation. WebSphere MQ Workflow uses a system group with multiple physical servers, each with its own queue manager in a WebSphere MQ cluster. There are also multiple physical servers for the UPES implementations, also in the same WebSphere MQ cluster.

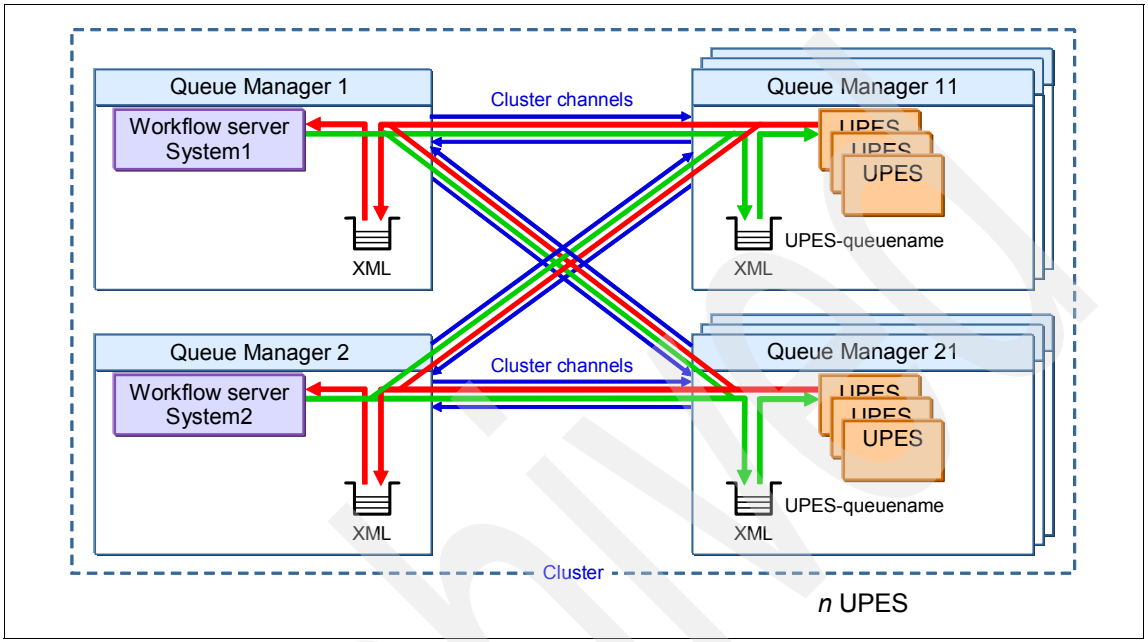


Figure 5-6 UPES: Load balancing within a workflow system group

A workflow can run on any of the servers in the system group. The message goes into the cluster where it can be processed by any of the UPES implementations on any of the boxes. For a large system with high throughput rates, this is the most desirable configuration.

Figure 5-6 shows UPES load-balancing within a WebSphere MQ Workflow system group.

5.1.3 Program execution server

The program execution server (PES) invokes application programs that you define in the workflow model. These programs run in CICS or IMS. A PES plug-in, the *invocation exit*, can be used to invoke other programs. These exits use invocation protocols for applications that are running in other environments.

The program execution server can map parameters to a format expected by an invoked program so that the invoked program does not have to be changed. This

applies if an application does not use the container API to access its input or output data, for example, a data structure that is passed in a CICS COMMAREA.

Programs invoked by the PES are always *unattended*. No user involvement is needed. The PES is designed to incorporate security checking, such as z/OS Security Server (RACF®). For details on how to incorporate application programs, see the *IBM WebSphere MQ Workflow for z/OS Programming Guide.*

5.1.4 Program execution agent

The program execution agent (PEA) invokes and manages task-relevant application programs or tools that you define in the workflow model. Application programs can run on a different operating system from the one used for the server components of MQ Workflow. The program execution agent is used to start attended programs from the client machine. You can, however, also start programs that run in unattended mode on platforms on which no program execution server is available (that is, all non-z/OS platforms). For details on how to incorporate application programs, see the *IBM WebSphere MQ Workflow: Programming Guide*, SH12-6291, available at:

http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SH12-6291-10

5.2 WebSphere Process Server back-end overview

WebSphere Process Server offers an invocation mechanism to invoke service components. This is based on the Service Component Architecture (SCA).

An SCA module contains components, imports, and exports. The focus of this section is on SCA imports with their binding options. A component implementation is, for example, a Java class or a BPEL process running within the J2EE environment. To connect an SCA module to a subsystem like CICS or IMS, SCA imports with J2C (J2EE connector architecture) bindings are used. For more information about the Service Component Architecture, refer to:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.prodovr.doc/topics/csrvcomparch.html

5.2.1 Service components

The building blocks of a business solution are Service Component Architecture (SCA) components wired together to form modules that can be deployed to the WebSphere Process Server.

A service component configures a service implementation. A component consists of an implementation (which is not shown in the assembly editor of WebSphere Integration Developer), one or more interfaces (which define its inputs, outputs, and faults), and zero or more references. A reference identifies the interface of another service or component that this component requires or consumes. An interface may be defined in one of two languages:

- WSDL port type
- Java

An interface supports synchronous and asynchronous interaction styles. A component's implementation can be in various languages.

Figure 5-7 illustrates the various implementation types, interface, and reference variants of a SCA component.

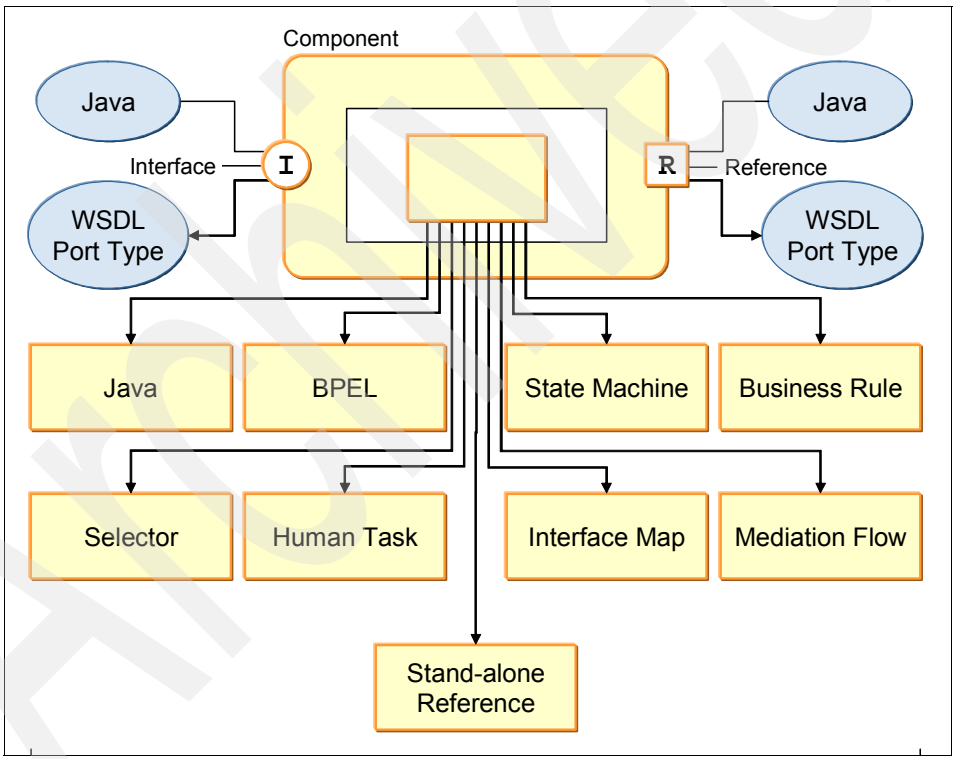


Figure 5-7 A service component

The assembly editor of the WebSphere Integration Developer lets you build applications by assembling the Service Component Architecture components. For more information about service components see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.prodovr.doc/topics/cservcomps.html

5.2.2 Accessing services

Imports allow modules to publish their services and to use services from other sources. Imports require binding information, which specifies the means of transporting the data from the modules. An import allows you to use functions that are not part of the module that you are assembling. Stand-alone references allow a Java program to invoke SCA components or imports.

Figure 5-8 shows how a business process component could be integrated with service implementations via various imports.

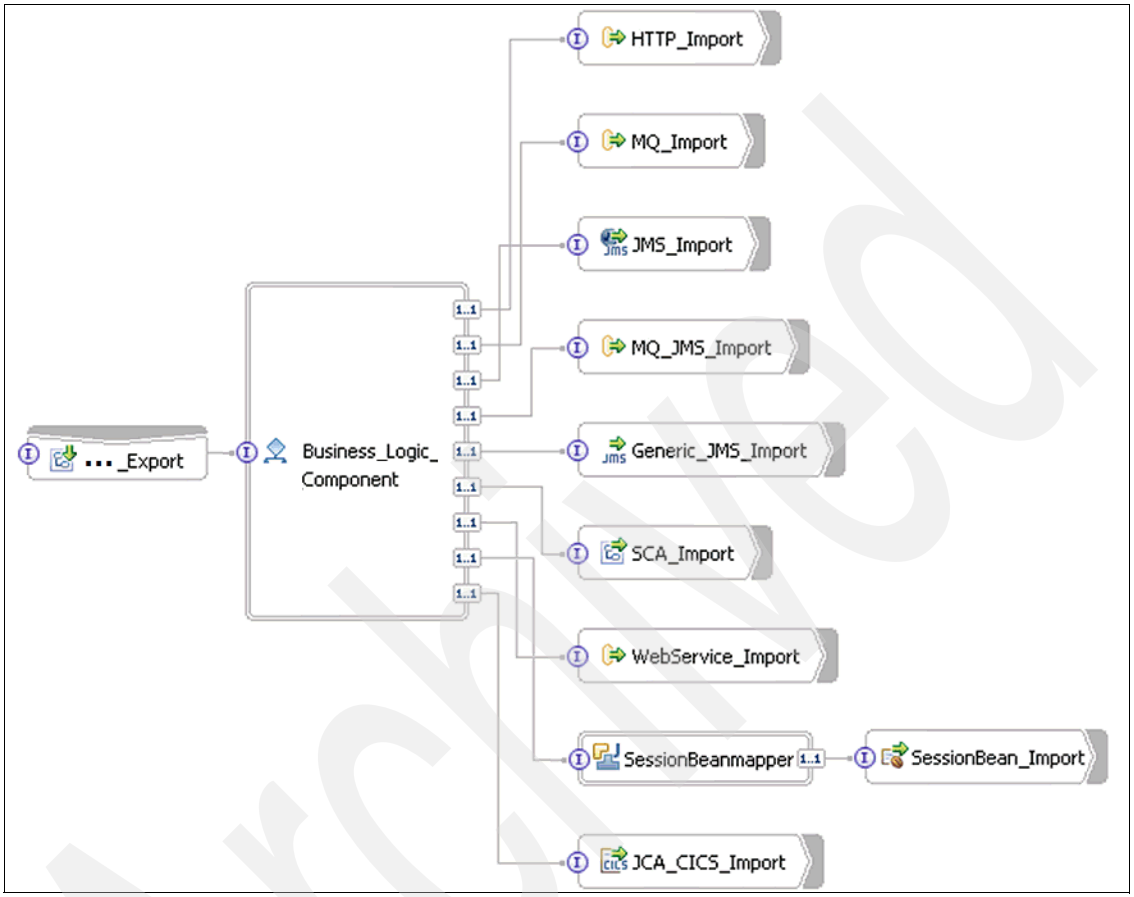


Figure 5-8 Module showing a business process component with various imports and exports

In addition, service components can interact with other applications on Enterprise Information Systems (EIS) with IBM WebSphere Adapters. For more information about accessing services see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.basics.doc/topics/aaccess1.html

Import bindings

Binding information determines how a service connects to and interacts with an application. Specifically, bindings are the protocols and transports assigned to imports and exports. Applications connect to services through a binding.

Import bindings are concrete definitions that specify the physical mechanism that SCA modules use to access an external service.

Figure 5-9 shows the WebSphere Process Server predefined bindings offered on an import component.

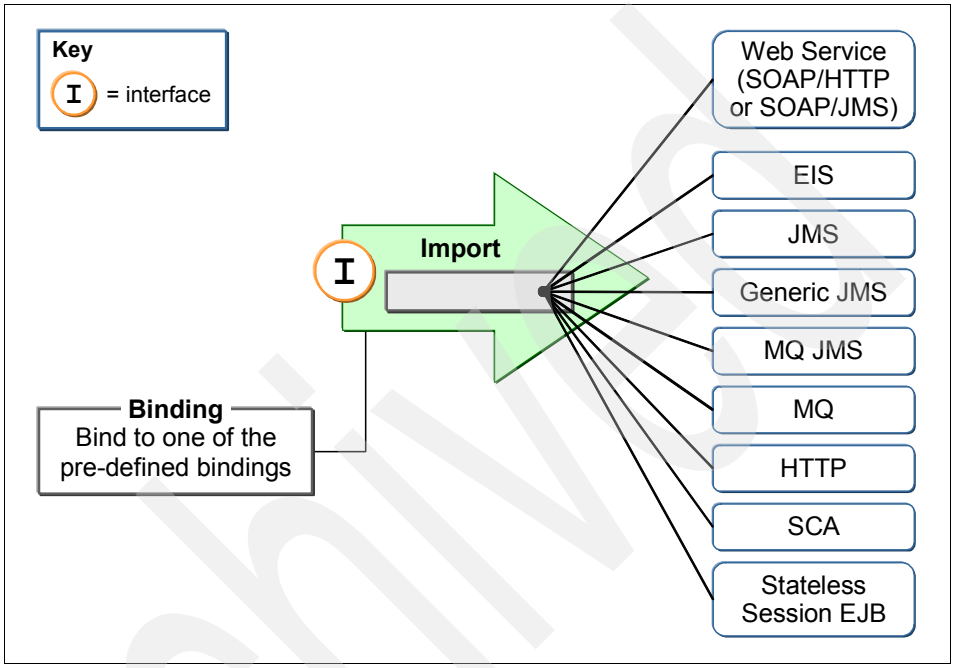


Figure 5-9 Bindings for imports

The following import bindings are offered by WebSphere Process Server:

- Web service binding

  Web service bindings allow components to invoke Web services. The supported protocols are SOAP/HTTP and SOAP/JMS.

- SCA binding

  SCA bindings connect SCA modules to other SCA modules. SCA bindings are also referred to as default bindings.

- Java Message Service binding

  Java Message Service (JMS) bindings allow interoperability with the WebSphere Application Server default messaging provider.

- WebSphere MQ JMS binding

  WebSphere MQ JMS bindings allow interoperability with WebSphere MQ-based JMS providers.

- WebSphere MQ binding

  WebSphere MQ bindings allow interoperability with WebSphere MQ.

- Generic JMS binding

  Generic JMS bindings allow interoperability with third-party JMS providers that integrate with the WebSphere Application Server using the JMS Application Server Facility (ASF).

- WebSphere Adapters binding

  WebSphere Adapters bindings enable interaction with Enterprise Information Systems (EIS).

- HTTP binding

  HTTP bindings allow you to access applications using the HTTP protocol.

- EJB binding

  EJB bindings allow you to access an imported service that is itself an EJB.

If you want to integrate Enterprise Information Systems, you generally use messaging bindings (JMS, generic JMS, MQ, and MQ JMS) or EIS bindings. EIS exports and imports are created with a particular resource adapter and let you interact with many EIS systems.

More information about bindings and on selecting the most appropriate one can be found here:

- http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.basics.doc/topics/cbindings.html
- http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.prodovr.doc/topics/cappbnd.html

Adapters

IBM WebSphere Adapters deliver generic technology and business application adapters with wizards that quickly and easily enable connection to Enterprise Information Systems.

There are two types of IBM adapters:

- WebSphere Adapters, also referred to as JCA adapters
  - WebSphere Adapters are based on the Java 2 Platform, Enterprise Edition (J2EE) Connector architecture (JCA), and are the recommended adapters to use with WebSphere ESB and WebSphere Process Server.
  - If an application is developed with a WebSphere Adapter embedded, the adapter is deployed with the application.
- WebSphere Business Integration (WBI) Adapters

  WebSphere Business Integration Adapters reside outside of WebSphere ESB or WebSphere Process Server. The runtime communicates with this type of adapter through a Java Message Service (JMS) transport layer.

IBM WebSphere Adapters allow for integration of existing EIS infrastructure and applications that are deployed on WebSphere Process Server. WebSphere Adapters enable you to quickly and easily create integrated processes that exchange information between enterprise resource planning, human resource, customer relationship management, and supply chain systems.

The following adapters can be configured to work with WebSphere Integration Developer.

- WebSphere Adapter for Email
- WebSphere Adapter for Flat Files
- WebSphere Adapter for FTP
- WebSphere Adapter for JDBC™
- WebSphere Adapter for JD Edwards® EnterpriseOne
- WebSphere Adapter for Oracle E-Business Suite
- WebSphere Adapter for PeopleSoft® Enterprise
- WebSphere Adapter for SAP Software
- WebSphere Adapter for Siebel Business Applications
- CICS ECI resource adapter

  The Customer Information Control System External Call Interface (CICS ECI) resource adapter lets you access programs on CICS servers. Working with the external service wizard, the CICS ECI resource adapter creates services that invoke CICS transactions on a CICS server.
- IMS TM resource adapter

  The IMS TM resource adapter is used by services to access IMS transactions running on host IMS systems. It can also be used for IMS transactions to access external services.

- Using WebSphere Business Integration Adapters

  These adapters do not comply with either the J2C architecture or the Enterprise Metadata Discovery specification, yet can be used with the external service wizard.

For more information see:

- Overview of adapters:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wbpm.scenarios.esb1.610.doc/concepts/cwesb_scenario_adapters.html

- Configuring and using adapters:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/tcreatecmps.html

- Developing services with adapters:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/coverview.html

- IMS TM resource adapter:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/ims_overview.html

- CICS ECI resource adapter:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/cicsoverview.html

CICS connection possibilities

To integrate Enterprise Information systems such as CICS, several connection possibilities are offered:

- Connect using a JCA adapter.
- Connect using a Web service.
- Connect using the WebSphere MQ-CICS bridge.

For more information see:

- How to connect to CICS

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpm.scenarios.esb1.610.doc/concepts/cwesb_scenario_cics.html

- IBM Redbook publication: *WebSphere for z/OS V6 Connectivity Handbook*, SG24-7064

  http://www.redbooks.ibm.com/abstracts/sg247064.html

IMS connection possibilities

To integrate Enterprise Information Systems such as IMS, several connection possibilities are offered:

- Connect using a JCA adapter.
- Connect using a Web service.
- Connect using the WebSphere MQ-IMS bridge.

For more information see:

- IMS TM resource adapter

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/ims_overview.html

- IBM Redbook Publication: *WebSphere for z/OS V6 Connectivity Handbook*, SG24-7064

  http://www.redbooks.ibm.com/abstracts/sg247064.html

5.2.3 Service invocation

A business process can consume Service Component Architecture services. The way in which data is exchanged between the SCA service and the process depends on how the process was modeled.

The consumption of a service in a business process is implemented using a Business Process Execution Language (BPEL) *invoke* activity in the process model. The data that is passed to the SCA service is retrieved from one or more BPEL variables. For more information about invocation scenarios for business processes, see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/cinvocation.html

5.3 Transition planning for user-defined program execution server implementations

To plan for a transition of user-defined program execution server (UPES) implementations from WebSphere MQ Workflow to WebSphere Process Server, you must analyze the implementation in place in the WebSphere MQ Workflow environment and find out what the invoked service does.

For more information about the analysis of the implementation, as well as information about how to transition a UPES program invocation from WebSphere MQ Workflow to WebSphere Process Server, refer to Chapter 11, “Integrating back-end applications” on page 251, and “UPES questionnaire” on page 426.

UPES program invocation

This section describes a UPES invocation framework, how WebSphere MQ Workflow does program invocation via a UPES, and shows a conceptual alternative in WebSphere Process Server.

WebSphere MQ Workflow

WebSphere MQ Workflow sends a WMQWF-XML-based message to a WebSphere MQ queue of a UPES invocation framework. The invocation framework reads the message, invokes the UPES application, and returns the invocation result to the WebSphere MQ Workflow execution server.

WebSphere Process Server

WebSphere Process Server uses an SCA import with an MQ JMS binding to connect to the UPES invocation framework queue. In order to create a WMQWF-XML-based message a WMQWF data binding is needed.

UPES invocation framework

The invocation framework could be any of the following:

- WebSphere MQ Workflow UPES Framework
- Custom UPES framework
- Another product such as WebSphere Message Broker

The invocation framework reads the message, invokes the UPES application, and returns the invocation result to the calling server, which is either the WebSphere MQ Workflow execution server or WebSphere Process Server.

Summary

Figure 5-10 summarizes the native UPES program invocation possibility of both products. Additional integration mechanisms to those shown below are possible. For more detailed information refer to Chapter 11, “Integrating back-end applications” on page 251.

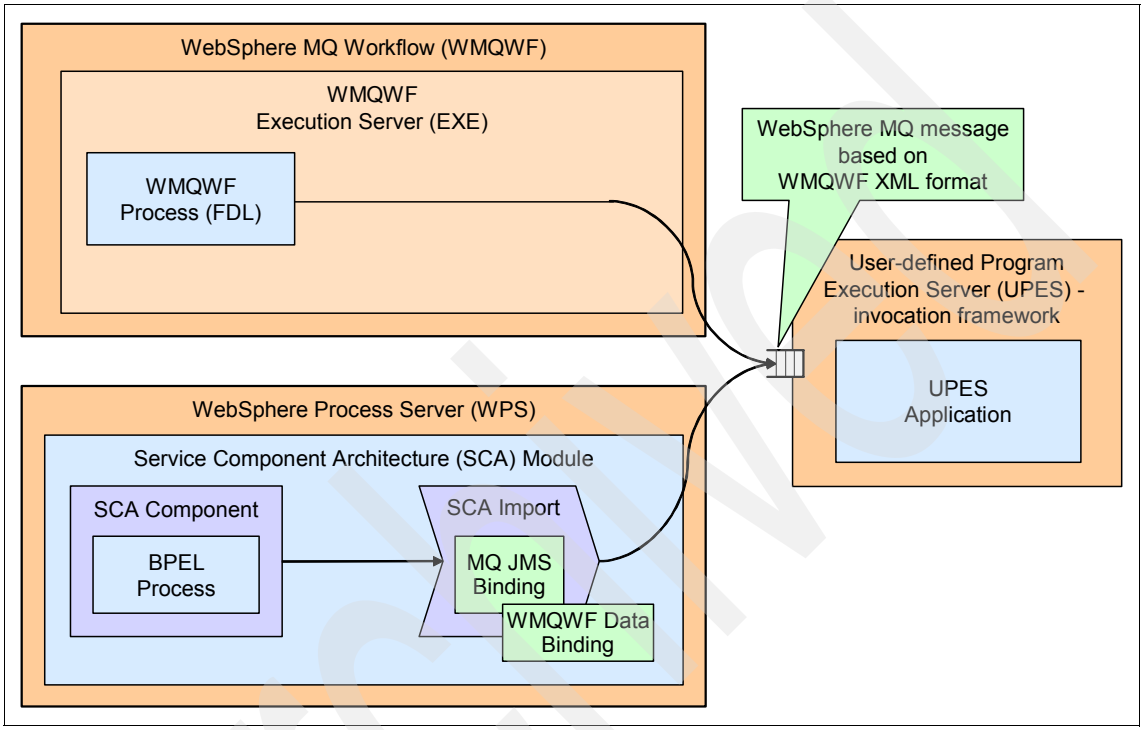


Figure 5-10 UPES program invocation

5.4 Transition planning for program execution server implementations

Transition planning for program execution server (PES) program invocations from WebSphere MQ Workflow to WebSphere Process Server requires an understanding of the current implementation in WebSphere MQ Workflow as well as some knowledge about WebSphere Process Server SCA modules, their binding possibilities, and integration using adapters.

For UPES program invocations to CICS and IMS, refer to the IBM Redbook Publication: *WebSphere for z/OS V6 Connectivity Handbook*, SG24-7064:

http://www.redbooks.ibm.com/abstracts/sg247064.html

The following sections provide an overview how a PES program invocation could be to transitioned from WebSphere MQ Workflow to WebSphere Process Server.

CICS/IMS program invocation

The PES of WebSphere MQ Workflow offers the following invocation types:

- Synchronous CICS program invocation
- Asynchronous CICS program invocation
- Synchronous IMS program invocation
- Asynchronous IMS program invocation

WebSphere Process Server allows similar invocations via an SCA import, specific binding, and an MQ message interface, or with a specific resource adapter.

Table 5-2 shows on a high level how PES invocation concepts map to WebSphere Process Server with SCA imports and their bindings.

Table 5-2 PES migration scenarios

Migration scenario	WebSphere MQ Workflow for z/OS PES (PES invocation exit)	WebSphere Process Server (SCA import binding)
Synchronous and transactional CICS program invocation via CICS Transaction Gateway (CTG)	External CICS interface (EXCI) call to the CICS Transaction Gateway	EIS binding for CICS adapter
Asynchronous message-based CICS program invocation via MQ Series CICS DPL Bridge (message with MQCIH header)	WebSphere MQ-CICS bridge	MQ message binding
Synchronous and transactional IMS program invocation via IMS Connect	CPIC call to OTMA	EIS binding for IMS adapter

Migration scenario	WebSphere MQ Workflow for z/OS PES (PES invocation exit)	WebSphere Process Server (SCA import binding)
Asynchronous message-based IMS program invocation via MQ Series IMS OTMA Bridge (message with MQIIH header)	WebSphere MQ-IMS bridge	MQ message binding

5.4.1 Synchronous CICS program invocation

This section describes and visualizes how WebSphere MQ Workflow does a synchronous CICS program invocation and shows a conceptual alternative in WebSphere Process Server.

WebSphere MQ Workflow

The PES of WebSphere MQ Workflow uses the external CICS interface (EXCI) to pass requests to a CICS program via the CICS Transaction Gateway.

WebSphere Process Server

WebSphere Process Server uses an SCA import with an Enterprise Information Systems (EIS) binding to connect an External Call Interface (CICS ECI) resource adapter. The CICS ECI resource adapter passes requests to a CICS program via the CICS Transaction Gateway.

Summary

Figure 5-11 summarizes the synchronous CICS program invocation possibility of both products.

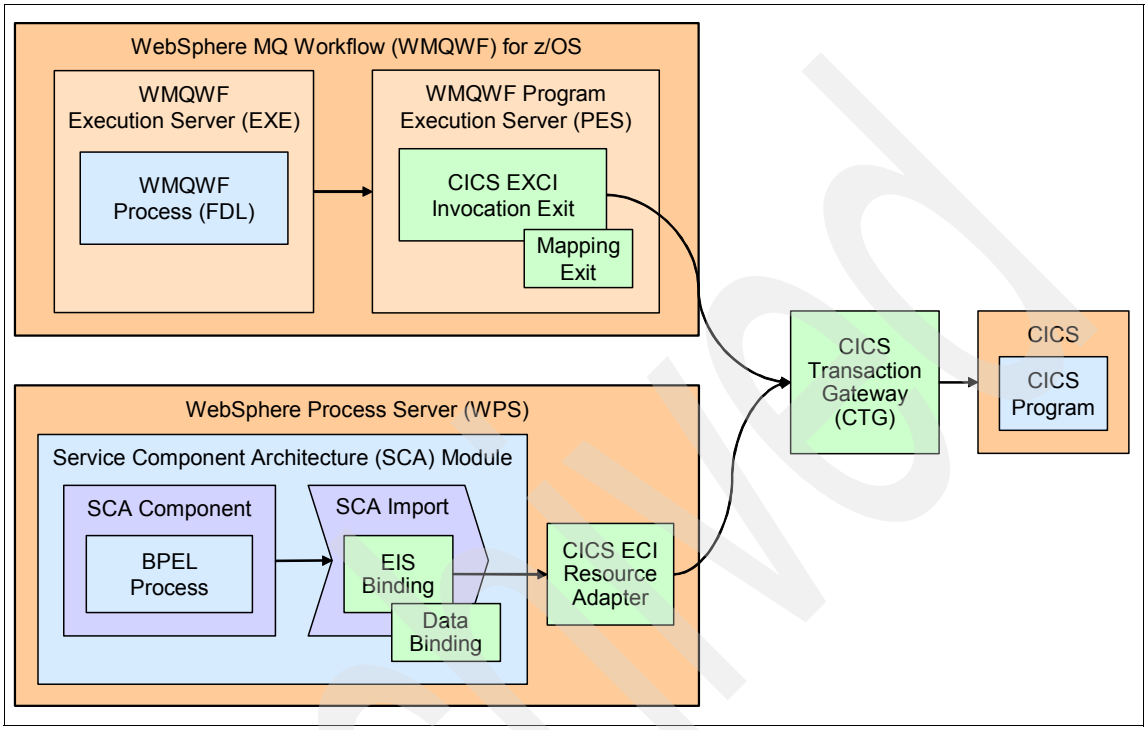


Figure 5-11 Synchronous CICS program invocation

5.4.2 Asynchronous CICS program invocation

This section describes how WebSphere MQ Workflow does an asynchronous CICS program invocation and shows a conceptual alternative in WebSphere Process Server.

WebSphere MQ Workflow

The PES of WebSphere MQ Workflow uses the MQ CICS invocation exit to pass requests to a CICS program via a WebSphere MQ-CICS (DPL) bridge.

WebSphere Process Server

WebSphere Process Server uses an SCA import with an MQ message binding to pass requests to a CICS program via a WebSphere MQ-CICS (DPL) bridge.

Summary

Figure 5-12 summarizes the asynchronous CICS program invocation possibility of both products.

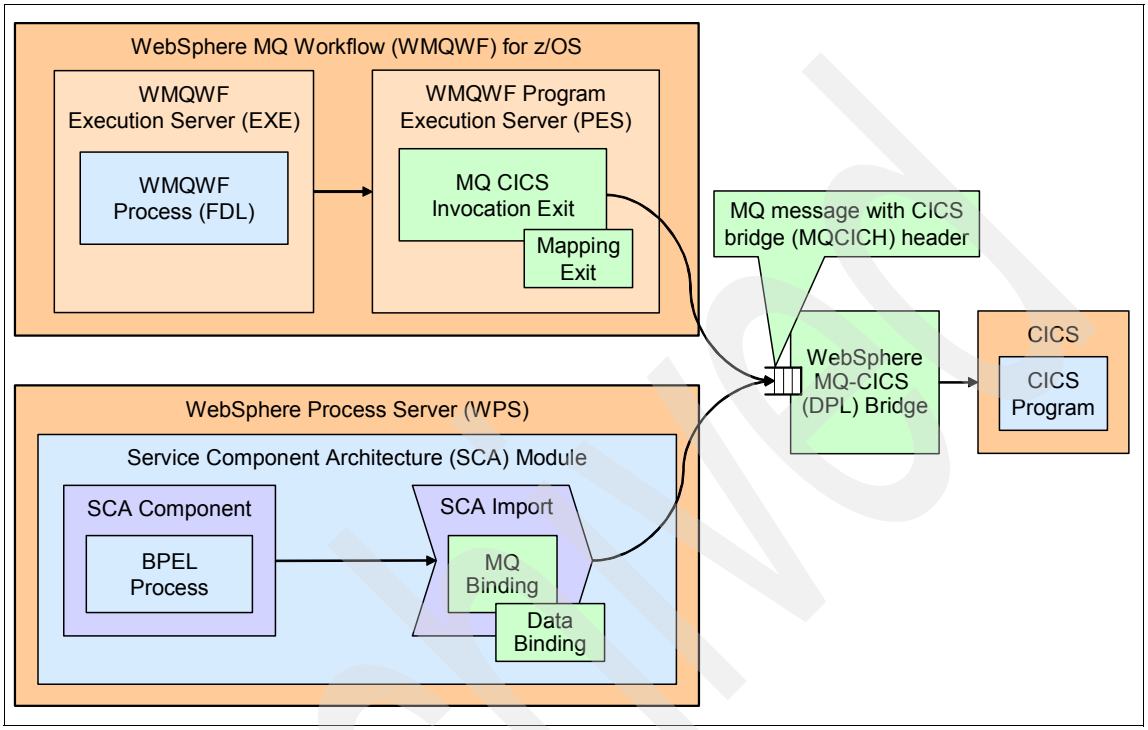


Figure 5-12 Asynchronous CICS program invocation

5.4.3 Synchronous IMS program invocation

This section describes how WebSphere MQ Workflow does a synchronous IMS program invocation and shows a conceptual alternative in WebSphere Process Server.

WebSphere MQ Workflow

The PES of WebSphere MQ Workflow uses the IMS CPIC invocation exit to pass requests to an IMS application via an IMS Connect.

WebSphere Process Server

WebSphere Process Server uses an SCA import with an Enterprise Information Systems binding to pass requests to an IMS application via an IMS Connect.

Summary

Figure 5-13 summarizes the synchronous IMS program invocation possibility of both products.

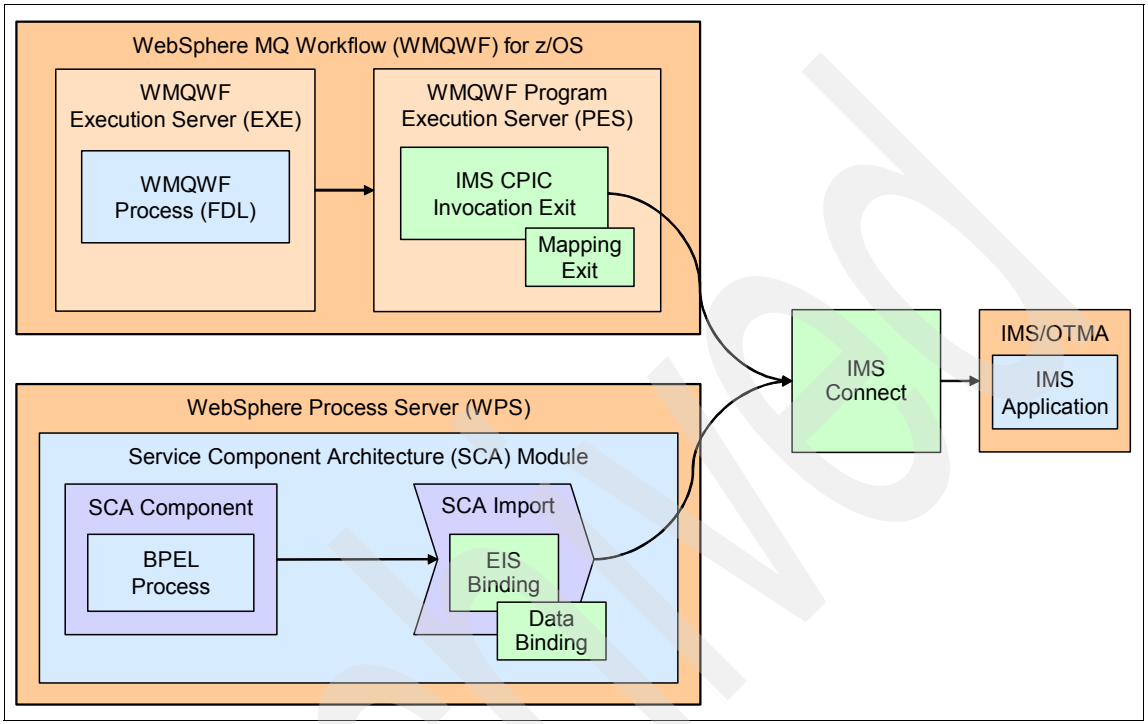


Figure 5-13 Synchronous IMS program invocation

5.4.4 Asynchronous IMS program invocation

This section describes how WebSphere MQ Workflow does an asynchronous IMS program invocation and shows a conceptual alternative in WebSphere Process Server.

WebSphere MQ Workflow

The PES of WebSphere MQ Workflow uses the MQ IMS invocation exit to pass requests to a IMS/OTMA application via a WebSphere MQ-IMS (OTMA) bridge.

WebSphere Process Server

WebSphere Process Server connects via an SCA import with an MQ message binding to pass requests to a IMS application via a WebSphere MQ-IMS (OTMA) bridge.

Summary

Figure 5-14 summarizes the asynchronous IMS program invocation possibility of both products.

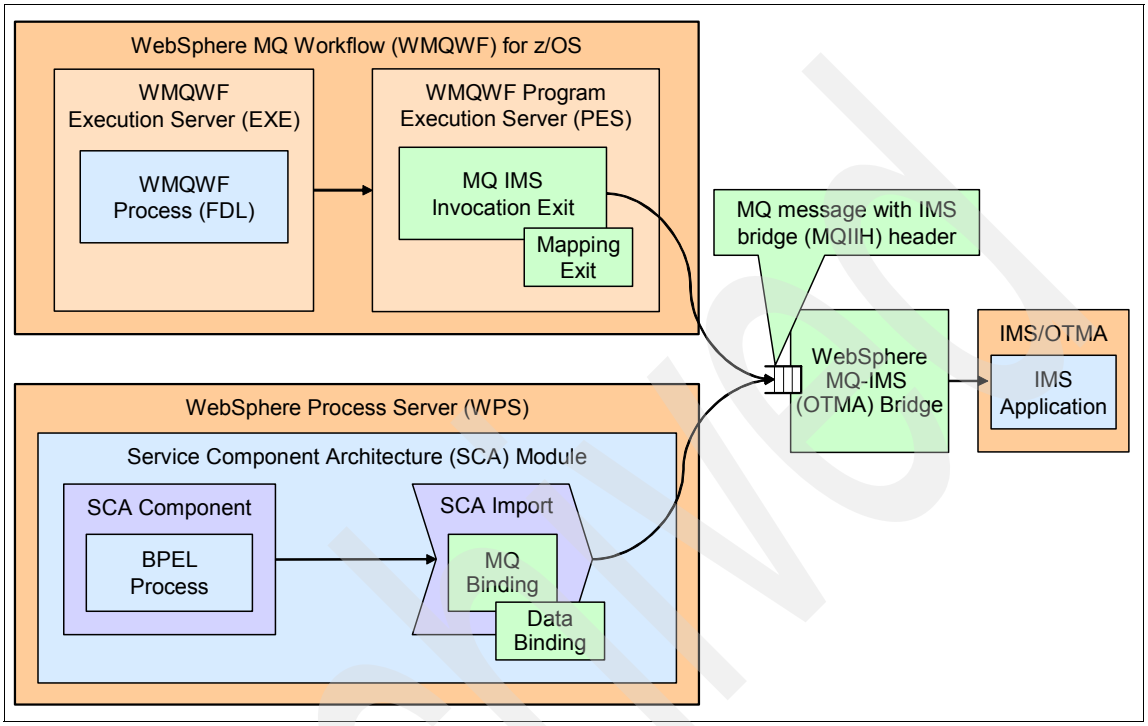


Figure 5-14 Asynchronous IMS program invocation

5.4.5 Data formats

For transition of WebSphere MQ Workflow data formats of the input and output containers to WebSphere Process Server business objects see Table 3-11 on page 50.

In order to map data from WebSphere Process Server business objects used within the SCA module to the format expected by invoked CICS/IMS programs (typically binary data structures defined using Cobol copybooks), data bindings for the SCA J2C imports must be generated from Cobol copybooks.

5.4.6 Security

No special considerations need to be made in WebSphere Process Server for WebSphere MQ Workflow z/OS PES security exit implementations. When using

the SCA J2C imports to invoke CICS/IMS programs, these invocations are automatically done on behalf of the authenticated WebSphere user.

5.5 Transition planning for program execution agent implementations

This topic is not covered in this book due to the following reasons:

- The WebSphere MQ Workflow best practices promoted the Java APIs and thin client technology for the front-end integration and the UPES technology for backend integration and not the program execution agent.
- In WebSphere Process Server no out-of-the-box functionality is available, which is comparable to the WebSphere MQ Workflow program execution agent.

6

Planning for clients based on application programming interfaces

This chapter discusses the following:

- Client options

  It provides an overview of the available clients options to interact within business processes. It describes the different client types of WebSphere MQ Workflow and WebSphere Process Server.

  It introduces task list handling implementation options to show the possibilities for WebSphere MQ Workflow and WebSphere Process Server. Task list handling is one of the most used usage pattern of clients.

- Application programming interface overview

  Since clients are based on public application programming interfaces (APIs). This provides a high-level API overview.

6.1 Client options

Human-based interaction within a business process is typically based on business and operational needs. These needs can be covered with a role-based human task model and with specific functions to interact with the process within a client implementation. The resulting human interaction scenarios define the feature content implemented in the clients.

A typical client categorization for the role-based process interactions is:

- Business user client: To work with the assigned tasks
- Administration client: To manage the life cycle of business processes and human tasks
- Web client with monitoring capabilities: To retrieve statistical information-based events and create reports on processes and activities
- Client application: To handle automated interactions

Client implementations also depend on the underlying technology, like J2EE, J2SE™, .net, Web Services, WebSphere MQ, and so on, and on the given environment such as the platform. Clients therefore must be based on different API languages. A typical client categorization for different technologies is:

- Web-based clients, also categorized as a thin clients
- Stand-alone clients, also categorized as thick clients

6.1.1 WebSphere MQ Workflow clients

WebSphere MQ Workflow offers the following clients:

- WebSphere MQ Workflow Web client: A Web application that implements a generic Web user interface for interacting with business processes and human tasks.
- Customized Web clients: You can customize the WebSphere MQ Workflow Web client instance to address the business needs of user groups that are assigned to this instance in various ways:
  - Creating your own Java Server Pages (JSPs) for displaying input and output container
  - Developing your own viewer to alter the look and feel of the application
  - Adding your own command handler to enhance/extend the functionality
  - Configuring client properties to modify the default behavior
- WebSphere MQ Workflow portal-based client: Allows you to access a WebSphere MQ Workflow system using workflow-specific portlets.

- WebSphere MQ Workflow standard client: Based on WebSphere MQ Workflow ActiveX API (Windows platform only).
- Custom clients: Creating custom clients allows you to address specific business needs of user groups.
- Custom application: A custom application allows you to address other specific business needs, for example, an automated batch application to start business processes.

6.1.2 WebSphere Process Server clients

WebSphere Process Server offers the following clients:

- Business Process Choreographer Explorer

  Business Process Choreographer Explorer is a Web application that implements a generic Web user interface for interacting with business processes and human tasks. This is an out-of-the-box Web client delivered by WebSphere Process Server.

  Depending on the user role users can use the Business Process Choreographer Explorer to perform the following tasks:

  - As a business administrator, you can manage the life cycle of business processes and can repair business processes. For example, you can restart or force the completion of single activities, or compensate the business process as a whole. If compensations fail, you can retry or skip the process instances and end whole compensation with stop. In addition, you can add and update custom properties for business processes and activities.
  - As a human task administrator, you can manage the life cycle of human tasks and manage work assignments. For example, you can assign responsibility to users or manage absence handling and substitutes for users. You can also change the priority and business category for human tasks and add or update custom properties.
  - As a business user, you can use Business Process Choreographer Explorer to work with your assigned tasks. For example, you can initiate business processes, services, and human tasks, and you can work on, edit, save, complete, or release human tasks. In addition, you can flag your absence and define substitutes.

  Furthermore, Business Process Choreographer Explorer offers a search function that you can use to discover business processes and their related activities and human tasks that need attention. For example, you can check the status of these instances, navigate between related instances and templates, and retrieve a graphical view of the process states that includes the associated activities and human tasks.

- Business Process Choreographer Observer

  The Business Process Choreographer Observer (BPC Observer) is a Web client providing out-of-the-box basic monitoring and reporting capabilities for processes and activities executing on WebSphere Process Server. While business processes and tasks are running, WebSphere Process Server can emit events that contain information about state changes of process instances and their related activities. Use Business Process Choreographer Observer to retrieve statistical information based on these events and create reports on processes and activities.

  You can define your own reports or use a drill-down approach to get more detailed information about specific process instances, activity instances, or inline human tasks. In addition, you can export the reported results for further external processing.

  Business Process Choreographer Observer bridges a gap between IT-level monitoring and business-level monitoring. By providing means for reporting on events in the Business Flow Manager component, it helps you to understand what is happening in Business Process Choreographer.

- Business user client generator

  Part of the WebSphere Integration Developer that is useful for quickly prototyping human task processes or as a starting point for creating a custom client.

  WebSphere Integration Developer supports the generation of the following types:

  – JavaServer Faces (JSF)-based client

    The JSF client is generated based on data described in the interface that the human task implements and does not need any input defined in the user interface settings.

  – IBM Lotus Forms-based client

    Lotus Forms allows you to easily integrate electronic forms with human tasks. You can generate a client based on forms that you specify in the user interface settings.

    A prerequisite is Lotus Forms API, Lotus Forms Viewer, and Lotus Forms Designer. The Lotus Forms API is needed to generate a client, Lotus Viewer to view a client, and Lotus Designer to edit a form.

– Portlet-based client for WebSphere Portal

  Select the portal client to point to an existing portlet by setting properties in the user interface settings. Or you can generate a portlet using the portlet generator.

In order to generate portlets, you need the Portal Toolkit. When you install WebSphere Integration Developer, you have the option to install the Portal Toolkit.

- Business Process Integration Extension for IBM WebSphere Portlet Factory

  This asset accelerates, simplifies, and reduces the skills required for the development of Web-based or portal-based user interfaces (UI) for human tasks in Business Process Management (BPM) solutions. Development tools involved are IBM WebSphere Portlet Factory and IBM WebSphere Integration Developer. Runtime products involved are IBM Websphere Portal and IBM WebSphere Process Server. For more details see the following articles:

  – Implementing a human-centric business process application using WebSphere Portlet Factory: Part 1: Solution overview

    http://www.ibm.com/developerworks/websphere/library/techarticles/0804_ng/0804_ng.html

  – Implementing a human-centric business process application using WebSphere Portlet Factory: Part 2: Developing a human-centric business process

    http://www.ibm.com/developerworks/websphere/library/techarticles/0804_kapadia/0804_kapadia.html

- Customized Business Process Choreographer Explorer

  You can configure multiple instances of the Business Process Choreographer Explorer and customize each instance to address the business needs of user groups.

  – Customizing the Business Process Choreographer Explorer interface for different user groups

    The navigation pane in the default Business Process Choreographer Explorer user interface contains a set of links to predefined views. The My To-dos view is the default view that is shown after you log in. If you have one of the Business Process Choreographer system administrator roles, you can customize the links that are shown in the navigation pane, the view that your users see after they log in, and the information that is shown in the views.

  – Personalizing the Business Process Choreographer Explorer interface

    The navigation pane in the default Business Process Choreographer Explorer user interface contains a set of links to predefined views and

views that are defined by your system administrator. You can add your own views to your navigation pane, for example, to monitor the progress of a specific task or process.

- Changing the appearance of the default Web application

  Business Process Choreographer Explorer provides a ready-to-use Web user interface based on JavaServer Pages (JSP) files and JavaServer Faces (JSF) components. A cascading style sheet (CSS) controls how the Web interface is rendered. You can modify the style sheet to adapt the user interface to fit a certain look and feel without writing any new code.

- Custom clients

  Creating custom clients allows you to address specific business needs of user groups. Use the rich set of API functions provided by the Business Flow Manager and Human Task Manager to develop custom clients.

  Useful starting points for information about custom client development are, for example:

  - The WebSphere Process Server V6.1 information center topic “Developing client applications for business processes and tasks”:

    http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6ovr.html

  - A generated business user client
  - The sample JSF-based client available on the Business Process Choreographer samples Web page at:

    http://publib.boulder.ibm.com/bpcsamp/

- Custom application

  A custom application allows you to address other specific business needs, for example, an automated batch application to start business processes.

  Useful starting points for a development of such an application could be the WebSphere Process Server V6.1 information center topic “Developing client applications for business processes and tasks”:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t6ovr.html

Summary

Table 6-1 presents an overview of how the different client types are represented in each product and gives an overview of the functional enhancements.

Table 6-1 Client type overview

Client type	Availability in WebSphere MQ Workflow	Availability in WebSphere Process Server or WebSphere Integration Developer
Business user client	Yes, via a: ► Customized Web client ► Custom client ► Custom portal client	Yes, via a: ► Client generation ► Customized client ► Custom client ► Portlet generation
Administration client	Yes, via a: ► Web client as part of the product ► Custom client	Yes, via a: ► Business Process Choreographer Explorer as part of the product ► Custom client
Web client with monitoring capabilities	Yes, via a Web client with a custom command handler	Yes, via a Business Process Choreographer Observer as part of the product
Client application	Yes, via a custom client application	Yes, via a custom client application

Recommended reading

It is beyond the scope of this book to provide all the detail of information about clients. For further information about WebSphere MQ Workflow clients see:

- *IBM WebSphere MQ Workflow: Concepts and Architecture*, GH12-6285
- *IBM WebSphere MQ Workflow: Programming Guide*, SH12-6291
- *IBM WebSphere MQ Workflow for z/OS: Programming Guide*, SC33-7031
- The documentation delivered with the install image <WMQWF_installation_directory>/doc/content.html

For further information about WebSphere Process Server clients see:

- Setting up a user interface for your human task

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.tel.ui.doc/topics/tnclient.html

- About Business Process Choreographer Explorer

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/c7webclt.html

- Getting started with Business Process Choreographer Explorer

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t7explorer_using.html

- Business Process Choreographer Explorer user interface

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/c7explorer_areas.html

- Planning for the Business Process Choreographer Explorer

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t1pl_explorer.html

- Developing client applications for business processes and tasks

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6ovr.html

6.2 Client implementation concepts

This section describes the basic task list programming model of WebSphere MQ Workflow and WebSphere Process Server. In addition, it discusses the enhanced task list programming models available in WebSphere Process Server.

6.2.1 Basic task list programming model

A typical client would have implemented a task list programming model. If we compare a WebSphere MQ Workflow client with a WebSphere Process Server client, we find a very similar basic programming model. From an high-level implementation concept mostly the technical terms have changed. This enables the client developer to do a straight-forward transition from WebSphere MQ Workflow to WebSphere Process Server. The similar programming model also makes it easier for the users who are working with the same or a similar user interface behavior of the client.

WebSphere MQ Workflow

In WebSphere MQ Workflow the task list programming model is:

1. Query for the to-do tasks (work items).
2. Display the query result as a list (work list) with the most important attributes.
3. The business or administrative user selects a preferred item and locks it (check-out) for further processing.
4. The business or administrative user works on it, based on the provided input/output data (get input container/get output container).
5. The server feeds back the result of the work (set output container).
6. Final completion of the work (check-in).

WebSphere Process Server

In WebSphere Process Server the task list programming model is:

1. Query for the to-do tasks.
2. Display the query result as a list (task list, task view) with the most important attributes.
3. The business or administrative user selects a preferred item and locks it (claim) for further processing.
4. The business or administrative user works on it, based on the provided input/output data (get input message/get output message).
5. The user feeds back the result of the work (create output message).
6. Final completion of the work (complete).

Summary

Figure 6-1 summarizes and compares the WebSphere MQ Workflow and WebSphere Process Server basic programming task list models.

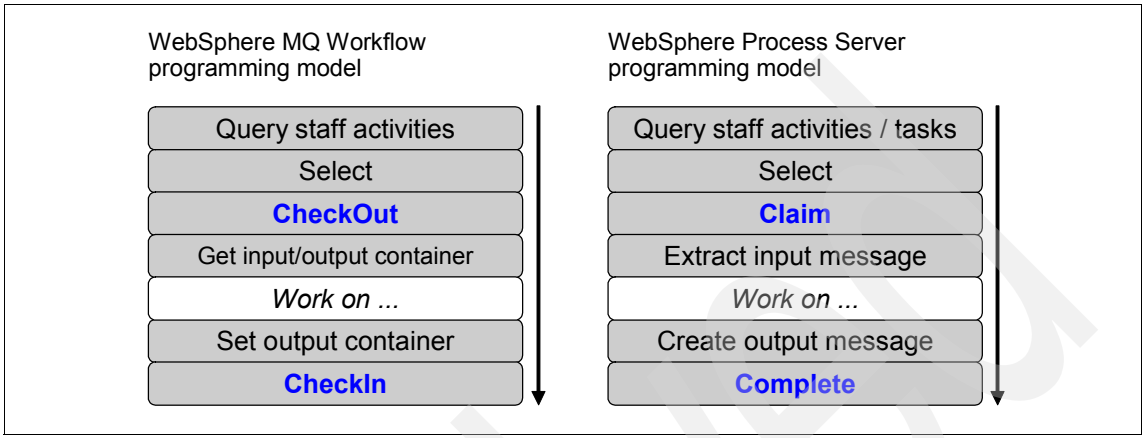


Figure 6-1 Basic task list programming model

Technology differences resulted in a different implementation in other WebSphere Process Server programming model areas such as exception handling. For more information and mapping details, refer to Chapter 12, “Implementing clients based on application programming interfaces” on page 275.

WebSphere Process Server offers additional enhanced functions, however. Some of these enhanced capabilities are described in the following sections.

6.2.2 Enhanced task list programming models

WebSphere Process Server offers additional enhanced features. Most of these new features were already requirements submitted by WebSphere MQ Workflow customers. Some of these customers circumvented the missing WebSphere MQ Workflow functionality by writing their own, sometimes complex, custom implementations.

The following sections describe some of the new enhanced programming models and features now available with WebSphere Process Server.

Enhanced task list programming model: Alternative 1

The high-level interaction steps of this enhanced programming model are:

1. Query for the to-do tasks (tasks in ready state).

2. Display the query result as a list (task list, task view) with the most important attributes.
3. The business or administrative user selects a preferred item and locks it (claim) for further processing.
4. The business or administrative user works on it, based on the provided input/output data (get input message/get output message).
5. The user saves a preliminary working result (create output message and save output message).

> **Note:** Between this and the following step, a long-term interruption may occur. Should a logout occur, steps 1 to 3 must be repeated. The previously stored intermediate result can, however, be re-used to do the final work completion.

6. Retrieve the saved data of step 5.
7. Complete the work based on the input data and the previously stored and retrieved output.
8. The user feeds back the result of the work (set parts of the output message).
9. Final completion of the work (complete).

Enhanced capability: Interrupting a unit of manual work with saving the preliminary working results.

Figure 6-2 illustrates the alternative described above.

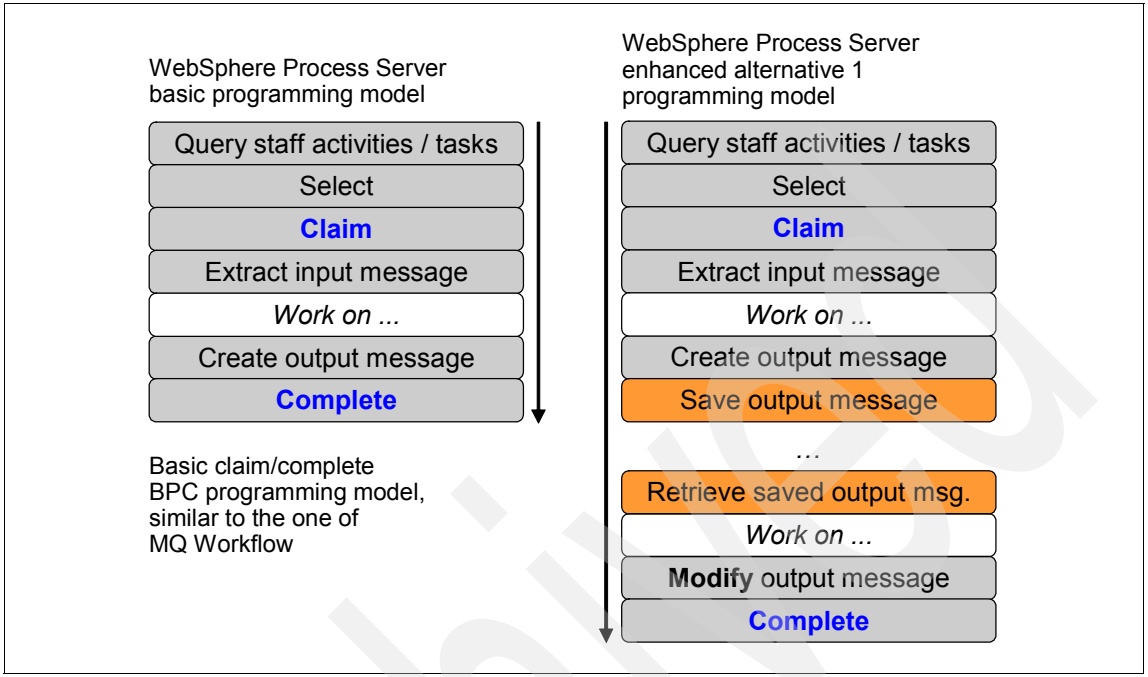


Figure 6-2 WebSphere Process Server enhanced task list programming model: Alternative 1

Enhanced task list programming model: Alternative 2

The high-level interaction steps of this enhanced programming model are:

1. Query for the to-do tasks (tasks in ready state).
2. Display the query result as a list (task list, task view) with the most important attributes.
3. The business or administrative user selects a preferred item and locks it (claim) for further processing.
4. The business or administrative user works on it based on the provided input/output data (get input message/get output message).
5. Save a preliminary working result (create output message and save output message).
6. Transfer the work to a user with the preliminary working results.

Note: The following steps are performed by the user to whom the work was transferred.

7. Query for the to-do tasks (tasks in claimed state).
8. Display the query result as a list (task list, task view) with the most important attributes.
9. The business or administrative user selects a preferred item and locks it (claim) for further processing.
10. The user works on it based on the provided input and on output data set by the previous user (get input message/get output message).
11. The server feeds back the result of the work (set parts of the output message).
12. Final completion of the work (complete).

Enhanced capability: More than one user can complete a unit of manual work.

Figure 6-3 illustrates the alternative described above.

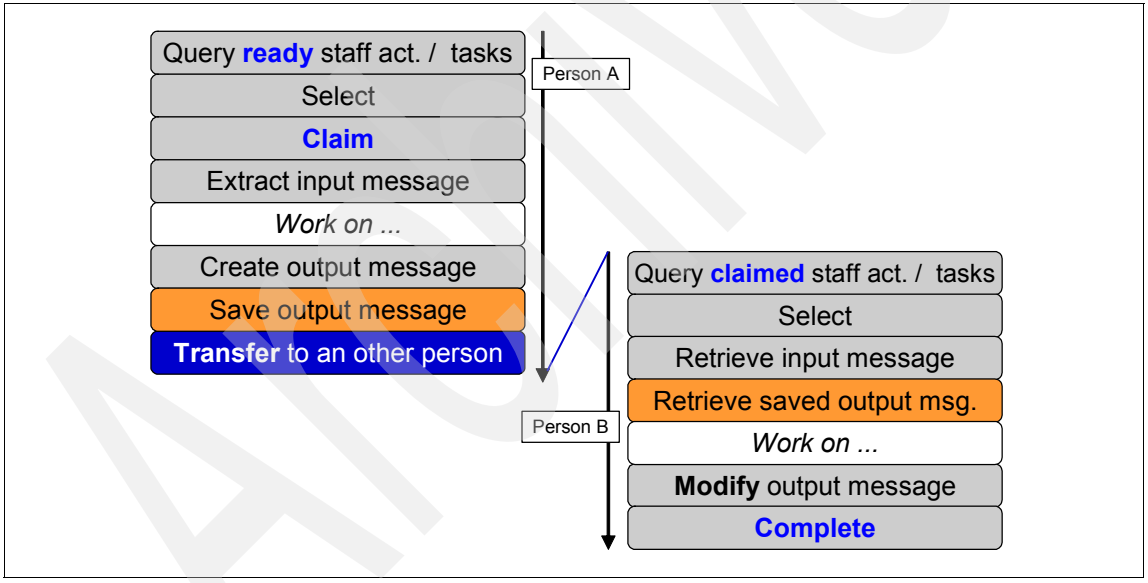


Figure 6-3 WebSphere Process Server enhanced task list programming model: Alternative 2

Enhanced task list programming model: Alternative 3

When two or more users try to claim the same human task, only one user succeeds. The other user is denied access.

Only one user can claim a human task. If several users attempt to work with the same human task at the same time, the probability of collision increases. Collisions cause delays because of lock waits on the database or rollbacks.

Some ways to avoid or reduce the incidence of collision are:

- If concurrent access is high, limit the number of users who can access a particular human task.
- Avoid unnecessary human task queries from clients by using intelligent claim mechanisms. For example, you might take one of the following steps:
  - Try to claim another item from the list if the first claim is unsuccessful.
  - Always claim a random human task.
- Reduce the number of potential owners for the task, for example, by assigning the task to a group with fewer members.

These concepts to reduce concurrent access to human tasks are known in both WebSphere MQ Workflow and in Business Process Choreographer. WebSphere Process Server offers an enhanced alternative to avoid concurrent access to human tasks. The high-level interaction steps of this enhanced programming model are:

1. Claim the next unit of work with a specific API method (a claim API method with an integrated query function returning exactly one item based on whereClause and orderByClause specifications).
2. The business or administrative user works on it based on the provided input/output data (get input message/get output message).
3. The server feeds back the result of the work (create output message).
4. Final completion of the work (complete).
5. Repeat steps 1 to 5 to process the next unit of work.

Enhanced capability: The combined claim and query functionality to grab the next available unit of work to avoid concurrent access to human tasks.

Figure 6-4 illustrates the alternative described above.

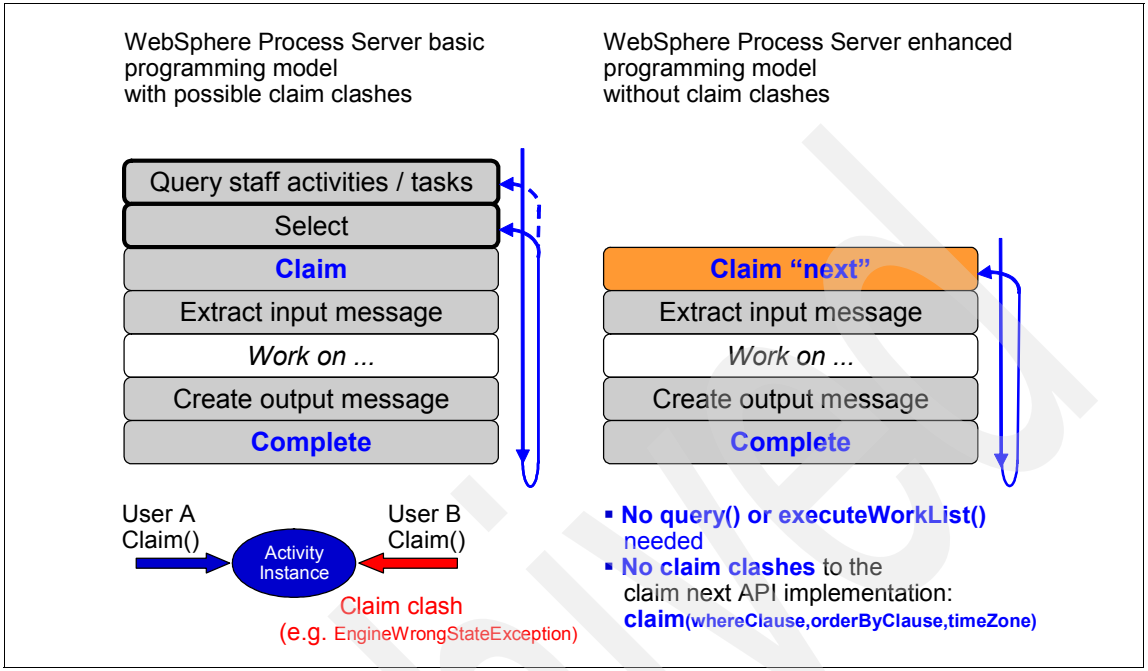


Figure 6-4 WebSphere Process Server enhanced task list programming model: Alternative 3

Enhanced task list programming model: Alternative 4

Some workflows are performed by only one person, for example, ordering books from an online bookstore. This type of workflow has no parallel paths. In an online bookstore the purchaser completes a sequence of actions to order a book. This sequence of actions can be implemented as a series of human task activities (to-do tasks). If the purchaser decides to order several books, this is equivalent to claiming the next human task activity. This type of workflow is also known as a page flow because user interface definitions are associated with the activities to control the flow of the dialogs in the user interface.

The Business Process Choreographer offers an API that supports the processing of this type of workflow.

The high-level interaction steps of this enhanced programming model are:

1. Query for the to-do tasks (tasks in ready state).
2. Display the query result as a list (task list, task view) with the most important attributes.
3. The business or administrative user selects a preferred item and locks it (claim) for further processing.

Note: Steps 1 to 3 are only the initial steps of a page flow.

4. The business or administrative user works on it, based on the provided input/output data (get input message/get output message).
5. The user feeds back the result of his work (create output message).
6. Final completion of the work and grabbing the next task within the page flow as one transaction (complete and claim successor).
7. Repeat steps 4 to 7 until the page flow ends.

Enhanced capability: API support for server-controlled page flows.

Figure 6-5 illustrates the alternative described above.

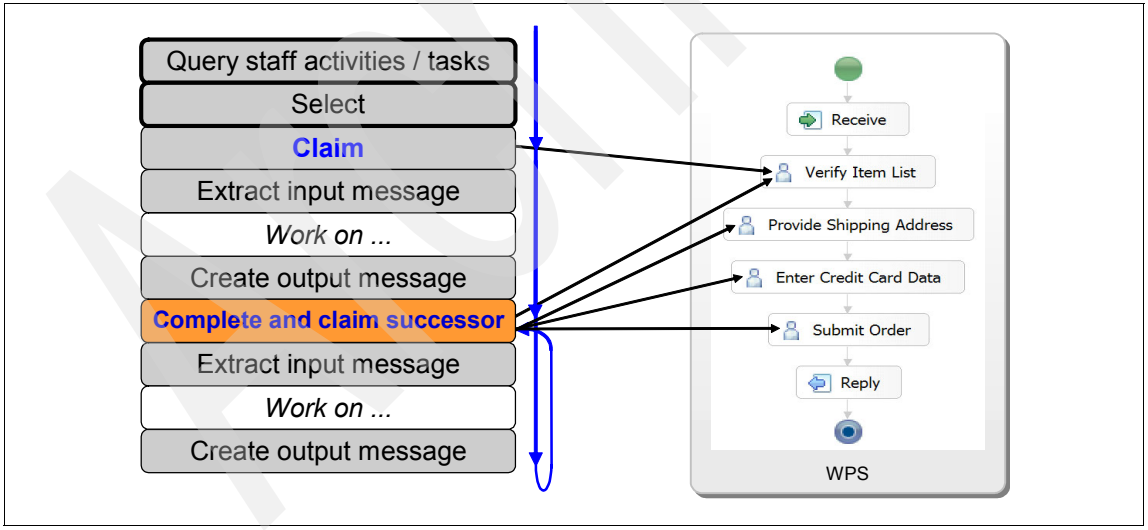


Figure 6-5 WebSphere Process Server enhanced task list programming model: Alternative 4

Enhanced task list programming model: Alternative 5

The task list programming allows users to give up ownership of a task. They can cancel the claim of the task. The task is put into the ready state again and it can be claimed by one of the potential owners. In WebSphere MQ Workflow, a similar concept (cancel checkout) is available. WebSphere Process Server offers an additional API parameter (keepData*)* that allows you to specify whether data saved for the claim activity instance is to be kept. A value of true states that any output or fault message set is to be kept. A value of false states that any output or fault message set is to be deleted.

In WebSphere Process Server the task list programming model is:

1. Query for the to-do tasks.
2. Display the query result as a list (task list, task view) with the most important attributes.
3. The business or administrative user selects a preferred item and locks it (claim) for further processing.
4. The business or administrative user works on it based on the provided input/output data (get input message/get output message).
5. The user feeds back the result of the work (create output message).
6. Instead of a final completion of the work the user decides to cancel the claim (give up ownership of a task).

   At this point the user can specify to keep any saved output or fault data (message).

Enhanced capability: Giving back the ownership of a task in combination with keeping any saved output or fault data.

Figure 6-6 illustrates the alternative described above.

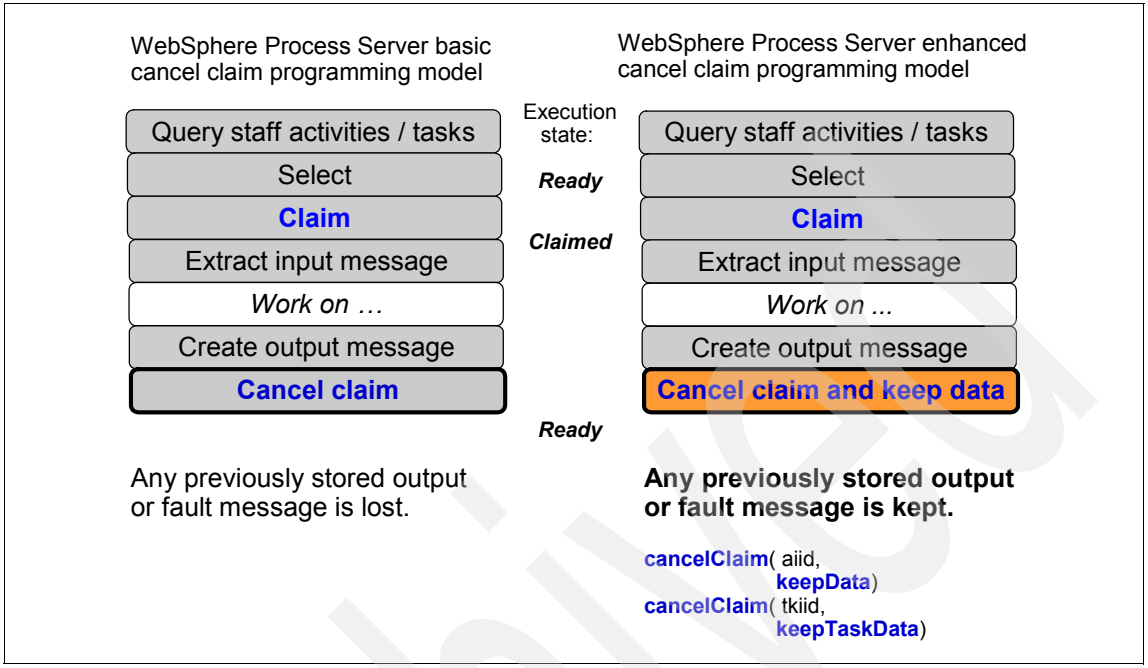


Figure 6-6 WebSphere Process Server enhanced task list programming model: Alternative 5

6.3 Application programming interface overview

WebSphere MQ Workflow, as well as WebSphere Process Server, offer APIs to interact with processes, human tasks, and so on. These APIs include the following conceptual areas:

- Create and start processes.
- Create lists for various tasks.
- Perform human tasks.
- Initiate work.
- Cancel or suspend work.
- Reassign work.
- Collaborate to achieve work.
- Handle out-of-office.
- Check status and analyze situations.
- Repair processes.
- Manage life cycle.

- Manage work assignments.
- Deal with data objects.
- Authenticate users.

Due to the differences of the architectures, component structure, and the enhancements, the APIs are grouped into different interfaces. Table 6-2 provides an overview over how the API groups are mapped.

Table 6-2 Application programming interfaces mapping

Application programming interface area	WebSphere MQ Workflow	WebSphere Process Server
Processes, activities	WebSphere MQ Workflow runtime application programming interface	Business Flow Manager application programming interface
Human tasks		Human Task Manager application programming interface
Data		► Service Data Object application programming interface ► Business Object application programming interface
Authentication		WebSphere Application Server foundation services based on Java Authentication and Authorization Service (JAAS)
	Authentication exit application programming interface	Java Authentication and Authorization Service exit
Service Component Architecture		Service Component Architecture application programming interface
Message driven interface	Inbound MQ-based XML interface	Java Message Service application programming interface
Staff administration	Staff administration application programming interface	API of the various people directory providers
Web client customizing	Web client framework application programming interface	Process Explorer framework application programming interface
Portal client customization	Portal client framework	► Portlet generator ► Portlet factory
Back-end interfaces	► User-defined program execution server framework ► PES framework	► Service Component Architecture application programming interface ► WebSphere Adapters

For more details about application programming interfaces see Chapter 12, “Implementing clients based on application programming interfaces” on page 275, and the related product documentation.

7

Planning for operational aspects

In addition to the business processes and client implementations, operational aspects must be considered when transitioning from WebSphere MQ Workflow to WebSphere Process Server. Deployment topology, high availability and scalability of the environment, security considerations, and performance targets must be assessed in the WebSphere MQ Workflow environment and corresponding target technology needs to be set up with WebSphere Process Server.

This chapter discusses the following at a conceptual level:

- Assessing the WebSphere MQ Workflow topology
  - Understanding the topology options available in WebSphere Process Server topology.
  - Some scenarios on how to map from the old to the new environment.

  Such an assessment is a complex task.
- Planning the WebSphere Process Server topology

  In addition, the architecture as well as the deployment topology of WebSphere MQ Workflow are different from the architecture and topology of WebSphere Process Server.

- Capacity planning

  This section provides information about how to plan the capacity requirements for various transition scenarios.

- WebSphere MQ Workflow to WebSphere Process Server transition recommendations

This chapter does not provide an exact project plan, but rather an overview of the relevant information of both products, as well as a set of hints and considerations that help you plan your target environment.

For detailed information about performing the actual assessment of the WebSphere MQ Workflow environment and on how to implement a corresponding environment in WebSphere Process Server, refer to Chapter 13, “Implementing operational aspects” on page 357. In addition, we recommend using the planning worksheets provided for the various transition aspects. They are located in Appendix A, “Transition planning worksheets” on page 411.

7.1 Assessing the WebSphere MQ Workflow topology

This section gives you a basic understanding of the components and requirements of the environment in place in WebSphere MQ Workflow. It describes the following concepts:

- The WebSphere MQ Workflow architecture and hierarchical system structure
- The server components
- The client architecture and components
- The client connection types
- The components for program execution
- Scalability, high availability, and workload management
- Architecture of integrating applications
- Authorization and staff resolution
- Operational setup for monitoring and auditing
- Cleanup of processes

7.1.1 The architecture and hierarchical system structure

To assess the WebSphere MQ Workflow setup in place, a basic understanding is required of the underlying architecture concepts, the hierarchical system structure, the components that belong to a system, and the communication between the components, which is based on WebSphere MQ message queuing.

Figure 7-1 shows an example of the system hierarchy.

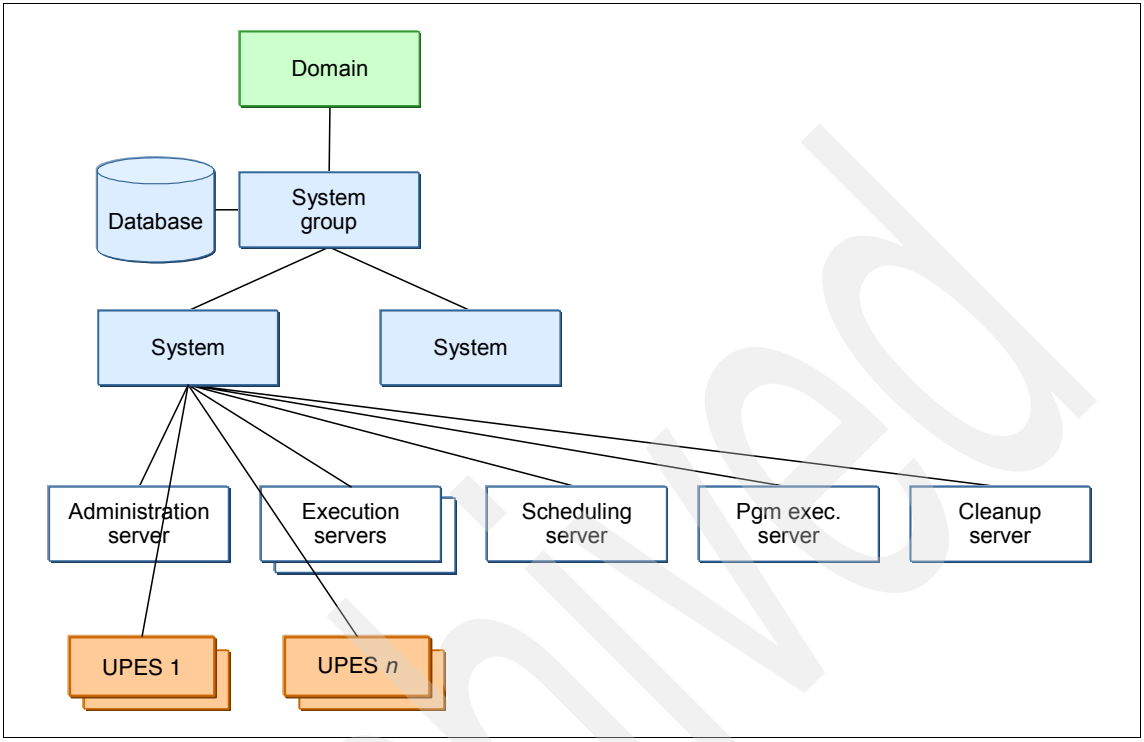


Figure 7-1 Example of a WebSphere MQ Workflow system hierarchy

The top-level hierarchy is called *domain*, representing all or parts of your organization. There can only be one domain per setup. A domain is a set of system groups that share the same workflow definitions. Any properties defined at this level (such as staff, data structures, programs, and IT resources) are valid for all systems and are inherited by all lower levels. If different settings are required at a lower level, they can be defined explicitly and are then valid for the level for which they were specified.

For example, a setting at system group level that defines that you do not want to keep audit information is valid for the system group and all systems in the system group in the domain. If, however, audit information for one of the systems in this system group is required, it can be specified for this particular system.

The next lower level is the *system group*. Within a system group, all systems share the same database. By installing more than one system in a system group, the workload for process execution can be distributed while still sharing the same data and the same workflow model. Databases cannot be shared by several system groups.

A *system* contains the client/server components that are needed to run the business processes. It hosts a set of special-purpose servers that use the same queue manager and that are commonly managed.

The actual installation can vary depending on the size and requirements of the organization and operating systems involved:

- It can be set up using more than one workflow system. WebSphere MQ Workflow is a client/server system with a hierarchical structure.
- It can reside on one or more physical machines. The system components that are installed on one physical machine are called a *node*.

The components of a WebSphere MQ Workflow system are designed for a multi-tier structure. The scope of each tier is clearly defined to exploit the available computing resources. The major components and their respective tiers are shown in Figure 7-2.

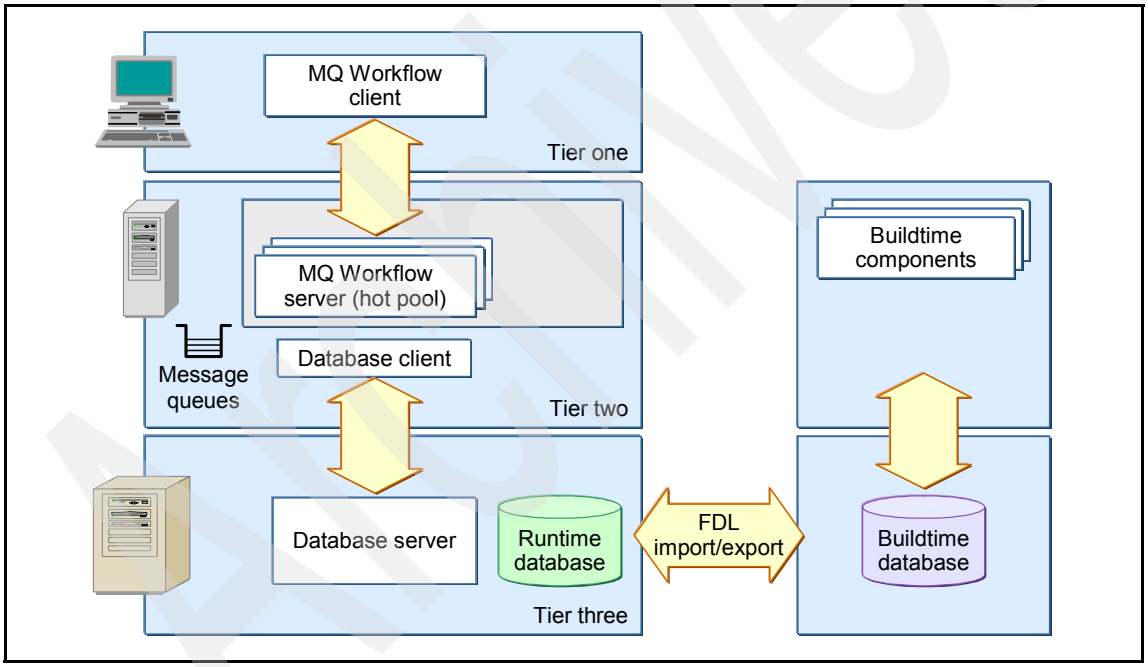


Figure 7-2 Three-tier architecture

The tiers are:

- Tier one: Client components

  Tier one represents the client APIs of WebSphere MQ Workflow and the clients that use these APIs. Clients are responsible for executing the program activities that interact with users. They give users access to the workflow management system (that is, the work items and running processes) and enable them to monitor the processes. Clients communicate with the servers through WebSphere MQ, using either the client or server message layer of WebSphere MQ Workflow, or via a Web-based client.

- Tier two: Server components and Buildtime

  Tier two represents the server components and Buildtime of WebSphere MQ Workflow. The server components manage the execution of processes at runtime. These components can be distributed across several machines to achieve workload balancing. For communication between the server components, WebSphere MQ message queuing is used.

- Tier three: Databases for Buildtime and Runtime

  Tier three represents the database tier. The database holds workflow-relevant data for a system group of WebSphere MQ Workflow. This includes status and setup information. For communication between the database server and its client, the transport protocols supported by the database product are used.

Note: Depending on the size of the organization and the size of the workflow model, the database can also reside on the same machine as all other server components. The system then consists of only two tiers, as shown Figure 7-3.

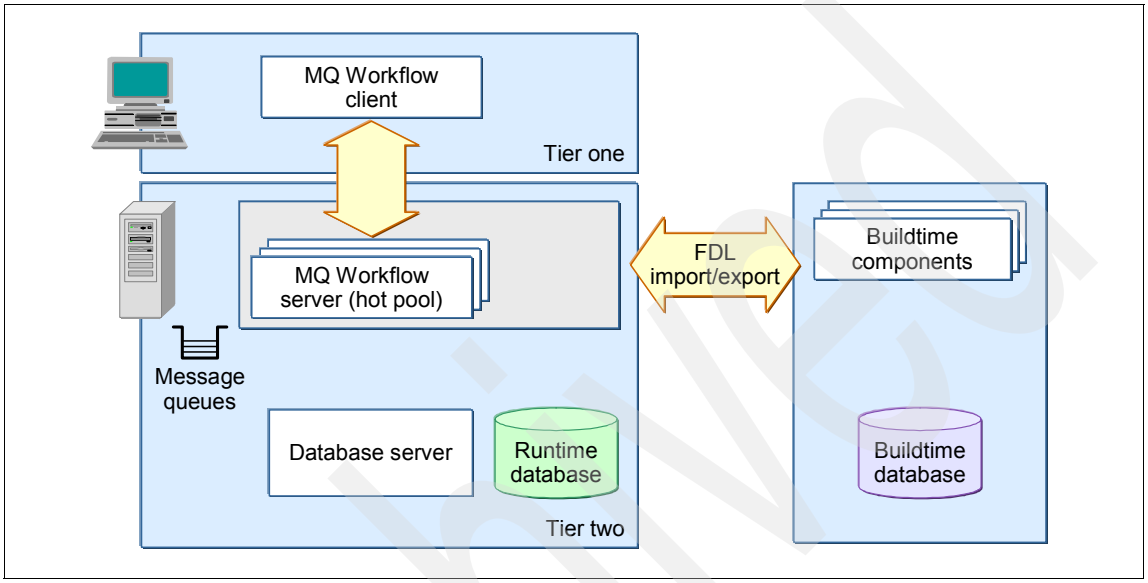


Figure 7-3 Two-tier architecture

In a two-tier structure, the server components, the message queuing components of WebSphere MQ, and the database management system reside on the second tier.

7.1.2 The server components

The server components coordinate and manage an MQ Workflow system and its clients. Server components also keep track of processes and administer process execution. Figure 7-4 shows the server components that make up a WebSphere MQ Workflow system.

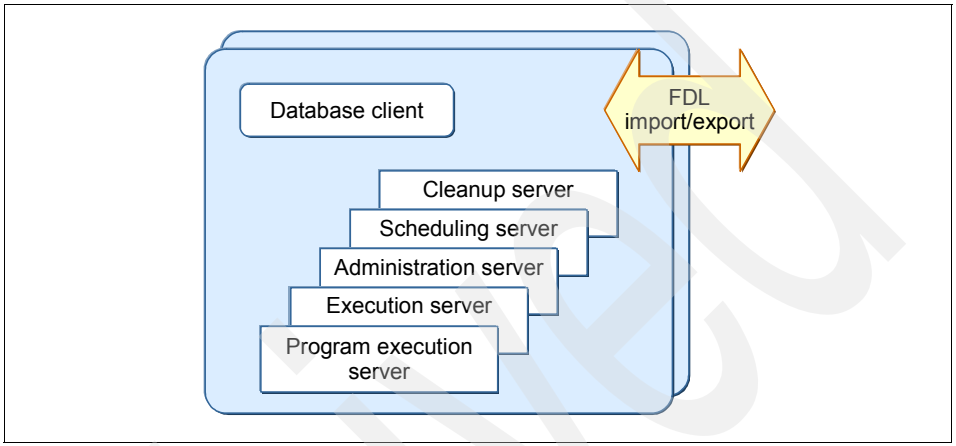


Figure 7-4 Server components of WebSphere MQ Workflow

The server components that make up a WebSphere MQ Workflow system are:

- Execution server

  The execution server is responsible for execution of processes. If a process contains activities for automatic execution, the execution server performs the correct activities at the correct time without human intervention. If staff is involved in a process, the correct work item is moved to the correct person at the correct time. It maintains the information about the states of all activities in the Runtime database. It acts as a database client to communicate with the database server.

  Note: For scaling purposes (that is, to handle larger workloads) more than one execution server can be set up and active per system.

- Administration server

  The administration server manages a WebSphere MQ Workflow system. It communicates with all other components in a system or system group. It is the working center of the administration component. It is responsible for the availability, operation, and error recovery of all server components. A command interface is used to send requests to the server's administration utility.

- Scheduling server

  The scheduling server controls and manages notification for activities that must be performed within a certain time frame. For example, if activities or work items are overdue for a process, the scheduling server changes the state of the activity to *expired*. For work items, the scheduling server sends notifications to the worklists of the relevant persons.

  Note: Only one scheduling server can be defined per system group.

- Cleanup server

  The cleanup server is responsible for physically deleting process instances that are finished. Depending on the definitions set for the system, finished processes are deleted immediately or later in the day when the system is idle.

  Note: Only one cleanup server can be defined per system group.

- Program execution server

  The program execution server is available on z/OS only and supports the invocation of IMS and CICS transactions. It is not covered in this edition of the book.

7.1.3 The client architecture and components

With WebSphere MQ Workflow, the following client options are available. For a transition project, you must assess which options are in place and which functionality they offer so that the most appropriate target technology can be put into place.

- WebSphere MQ Workflow client

  With a WebSphere MQ Workflow client you can start the execution of processes, use worklists to manage work items, and monitor your processes. WebSphere MQ Workflow offers or supports the following clients:

  - *Standard client* based on WebSphere MQ Workflow ActiveX API (Windows platform only).
  - *Custom client* based on APIs. Customers may have designed their own interface to perform worklist or monitoring tasks with a custom client using the WebSphere MQ Workflow APIs.
  - *Web client* based on the Java API. This client is a Java servlet that provides a Web interface for WebSphere MQ Workflow. It offers full process and worklist control as well as process monitoring functions.

- *Customized Web-based client* based on the Java APIs. Customers may have designed their own interface to perform worklist or monitoring tasks with a custom client using the WebSphere MQ Workflow Java APIs.
- *Portal-based client* that provides access to your WebSphere MQ Workflow system from within IBM WebSphere Portal. The functionality is similar to the IBM WebSphere MQ Workflow Web client. As with the WebSphere MQ Workflow Web client, the look and feel of the portal client can be customized, and its functionality can be extended.
- *Custom application* allows you to address other specific business needs, for example, an automated batch application to start business processes.

- Administration utility

  An administration utility is the administrator's user interface to request services from the administration server. Using this utility, you can start and stop a WebSphere MQ Workflow system and list the current state of any server. You can also access error logs to check whether any problems exist within the system.

- Staff Administration API

  In some cases, customers have written a specialized client based on the Staff Administration API. This API can be used to retrieve and change information related to the people defined in the MQ Workflow Runtime database. For more information see the *ÌBM WebSphere MQ Workflow: Programming Guide,* SH12-6291.

For more information about assessing which client options were implemented in WebSphere MQ Workflow and how to design a corresponding client in the WebSphere Process Server architecture, refer to Chapter 6, "Planning for clients based on application programming interfaces" on page 115.

7.1.4 The architecture of integrating applications

WebSphere MQ Workflow supports the integration of applications. The components used for program execution and the various options for integrating applications are described in this section.

Components for program execution

Figure 7-5 shows the components involved for program execution in a workflow model.

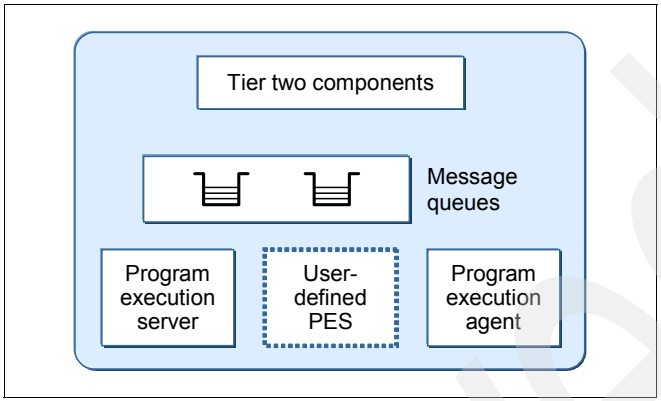


Figure 7-5 Components for program execution

The *program execution agent* is used to invoke application programs and tools that you defined in the workflow model. These programs can run on a different operating system from the one used for the WebSphere MQ Workflow servers. The program execution agent is used to start attended programs from the client machines, as well as programs that run in unattended mode on all non-z/OS platforms.

It is not considered best practice to use the program execution agent when planning to move to a WebSphere Process Server environment later or when planning on a browser-based environment for applications.

The program execution server also invokes application programs defined in the workflow model. However, these are programs run in a subsystem environment of z/OS, for example, CICS or IMS. In addition to program execution in IMS or CICS, you can plug in programs, using the program execution server invocation exit. These exits use invocation protocols for applications that are running in other environments. The program execution server is responsible for the mapping of parameters to the format that is expected by the invoked application. This applies if an application does not use the container API to access its input or output data, for example, a data structure that is passed in a CICS COMMAREA. For backend applications that run in unattended mode, no user involvement is needed. The program execution server is designed to incorporate security checking, such as z/OS Security Server (RACF). For details on how to incorporate application programs, see the *IBM WebSphere MQ Workflow: Programming Guide*, SH12-6291.

The *user-defined program execution server* (UPES) is implemented as a custom program using the documented message formats that apply to it. Using it you can integrate any type of application and use these applications to extend the management capabilities of WebSphere MQ Workflow. These applications can run on any type of operating system that is supported by WebSphere MQ. For example, it can be used to implement event-based applications using the Publish/Subscribe features of WebSphere MQ Integrator.

Integration using API support

WebSphere MQ Workflow offers APIs to support the interaction between the server and client components. In addition, the APIs can be used to invoke applications for workflow tasks. Using the client APIs, custom clients may also have been built, for example, for users to manage their work items.

Integration using message-based interfaces using XML

WebSphere MQ Workflow also offers a message-based interface. This interface supports the interaction with applications that use XML as message format. Instead of using APIs or the standard client interface, the message-based interface can be used, for example, to start a process instance using an XML message. This message can be created by a customer-written application or by any other application that can deal with XML messages (for example, WebSphere MQ Integrator).

Using WebSphere MQ Integrator publish/subscribe to manage events

WebSphere MQ Integrator offers a publish/subscribe programming model where so-called publisher applications send messages and so-called subscriber applications receive them. Publishers are not interested in where their publications are going and subscribers are not concerned with who delivers messages to them. WebSphere MQ Integrator supports an arbitrarily complex publish/subscribe network where publishers and subscribers can be connected to different brokers.

This publish/subscribe function can be used together with MQ Workflow to manage a program activity, which is designed to wait for an external event. The result of the external event then influences any further processing actions.

Running a WebSphere MQ Workflow process as a Web service

If a Web service is implemented as a process, at runtime a service requester submits a Web service request to the Web service provider. The Web service provider is a generic implementation delivered with an MQ Workflow SupportPac, which in turn intercepts and transforms the request into an XML format, which can be interpreted by WebSphere MQ Workflow. The implementation of the Web

service provider acts as a generic service adapter. It handles the transformation from the service request to the XML message, which is interpreted by WebSphere MQ Workflow Runtime to execute a process instance, and from the Workflow XML response message to the service result. For more information refer to the SupportPac WA07: WebSphere MQ Workflow - Web Services Process Management Toolkit available at:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000662&loc=en_US&cs=utf-8&lang=en

Integration using the UPES

The UPES is a fundamental part of the WebSphere MQ Workflow architecture. It helps WebSphere MQ Workflow connect to virtually any technology either to invoke a component implemented in that technology or to enable a component implemented in that technology to invoke MQ Workflow.

For more information about integration using the UPES and the corresponding target implementation options refer to Chapter 11, “Integrating back-end applications” on page 251.

Implementing a WebSphere MQ Workflow activity via a Web service

A Web service can provide the implementation for an activity within the context of a WebSphere MQ Workflow process. A process then acts as a service requester. A UPES is the component to handle the invocation of a Web service as the implementation of an activity. When the activity implementation is due to be invoked as part of the process navigation, WebSphere MQ Workflow sends an XML message to the UPES. This message carries all data that is needed for the service invocation. The content of the message identifies the Web service and specifies its parameters. Acting as a client to the Web service, the UPES calls the specified Web service via a proxy. Optionally, the UPES can return the result of the Web service as an XML response message to WebSphere MQ Workflow. This message signals the completion of the activity implementation.

7.1.5 Scalability, high availability, and workload management

The concepts of scalability, high availability, and workload management of WebSphere MQ Workflow and WebSphere Process Server differ considerably. This section covers the concepts available for WebSphere MQ Workflow at a high level.

Scalability

Scalability in WebSphere MQ Workflow can be achieved by hardware or by software. Scaling by hardware is done by increasing the processor capacity or using multiple processors. Scaling by software can be done in various ways.

Scaling by hardware:

- Processor capacity
- Multiple processors

Scaling by software:

- Two-tier system: Server and database on the same machine
  - Single execution server.
  - Multiple execution servers (hotpool). Each server has its own database connection. This helps to improve performance of the overall system.
- Three-tier system: Multiple servers per database
  - Distributed processing: WebSphere MQ queue manager and Runtime database located on different machines, exploiting the capacity of additional CPU.
  - Defining additional systems in the system group that access the same database or by connecting additional clients, or both.
- Thin clients for the Web

  By configuring multiple Web server agents as workflow clients.

High availability

WebSphere MQ Workflow is designed to support continuous operation. There are different influencing factors for availability, such as the system architecture supporting hotpooling and cluster support with shared queues.

To make a workflow management system highly available, a multi-tier configuration can be set up. In this type of setup, the database resides on a different tier from the WebSphere MQ Workflow servers. The clients reside on a tier of their own, as do the Web-based clients. Each tier is defined to exploit the available computing resources.

Workload management

The WebSphere MQ Workflow architecture allows you to manage your workload dynamically, depending on the setup that you choose.

Workload management within a system (hotpooling)

You can define more than one instance of the execution server. In this case, the load is shared between these instances for better performance. Each one of these instances has its own connection to the database server. This allows you to distribute the workload throughout the workflow system. The WebSphere MQ messaging and queuing functions allow multiple server instances to read from the same input queue. Each server instance executes in its own operating system process. All server instances that belong to the same WebSphere MQ Workflow system group use the same database. You can also define additional systems within a system group to balance the workload for your workflow system.

Workload management with WebSphere MQ clusters

WebSphere MQ queue manager clusters distribute the workload throughout all the systems of a system group. Within WebSphere MQ Workflow, a system group represents a WebSphere MQ cluster and the queue managers of the individual systems all belong to that cluster. Therefore, the execution servers of all the systems within an MQ Workflow system group represent a logical execution server and can be addressed through a single logical queue. Balancing the load between the queues within the system group is done automatically by the underlying WebSphere MQ cluster support.

7.1.6 WebSphere MQ Workflow security

Overall security in WebSphere MQ Workflow consists of these layers:

- Hardware/software infrastructure and networking.
- Database: General principles for securing data and database are used.
- Messaging layer: Principles for securing WebSphere MQ are used.
- WebSphere MQ Workflow: Performed via an administration component that communicates with an administration server. The administration server is responsible for session management in the WebSphere MQ Workflow system and handles all logon requests, and checks user identification, password, and authorization for a requested session.

7.1.7 Operational aspects of authorization and staff resolution

In WebSphere MQ Workflow, information about people is held in the Runtime database in a fixed, built-in staff repository. The available attributes of people, organizations, and roles are built-in and cannot be changed by the user.

An LDAP Bridge is provided to access an external LDAP staff repository to update the WebSphere MQ Workflow staff repository. The intention is to use this

tool (for example, each night) to propagate any changes made in the LDAP directory into the WebSphere MQ Workflow staff repository.

From an operational point of view, authorization and staff resolution is ensured as long as the staff data in the Runtime database is available. Updates of the staff repository with an LDAP database are not runtime-critical in WebSphere MQ Workflow.

7.1.8 Cleanup of processes and tasks

Both WebSphere MQ Workflow and WebSphere Process Server use databases to store process instance data as well as work items for human tasks. For long-running business processes, these databases enable the system to store the process state, so that even in the case of a server being stopped, the process can resume where it left off, rather than having to start again. A process template is stored in the database. When a process instance is started, this template is cloned into a process instance table. When the process has completed or is terminated, the process instance data is no longer needed in the database. However, immediately removing all of the process instance data and all of the work item data takes time. Rather than always performing these operations immediately, performance can be improved by deferring the work until a less busy time. For this reason, the concept of cleanup was introduced.

Cleanup in WebSphere MQ Workflow

WebSphere MQ Workflow includes a component called the cleanup server, which is responsible for deleting process and work item data from the database. Execution servers can be configured with the setting "NO IMMEDIATE CLEANUP", which causes the execution server to end a process faster because items were only *logically deleted*. At a later time, the cleanup server can perform the physical delete, perhaps running off-hours. Secondly, the times for *keep finished work items* and *keep finished processes* is set to eight hours, or some amount of time that would defer the clean-up to off-shift.

When the cleanup server starts, it first looks for work items to be deleted, then for process instances. When cleaning up a given process instance, the cleanup server checks whether work items associated with this process instance must be deleted. If the work items were previously deleted, then the cleanup server can proceed with cleaning up the process instance.

Modeling cleanup in WebSphere MQ Workflow

Cleanup policies are modeled on a process level or inherited. They are related to processes and work items.

To check these settings in WebSphere MQ Workflow Buildtime, open the process properties and select the **Control** tab. This tab contains selection options for keep finished work items and keep finished processes.

- Keep finished work items
  - Inherited

    If selected, the definition is taken from a previous setting in the inheritance hierarchy.
  - Never

    If selected, finished work items are not kept. Note that this is the default selection.
  - Forever

    If selected, finished work items are kept until you delete them from the Runtime database.
  - For

    If selected, you can specify for how long finished work items are kept until they are deleted.
- Keep finished processes
  - Inherited

    If selected, the definition is taken from a previous setting in the inheritance hierarchy.
  - Never

    If selected, finished process instances are not kept. This also deletes all the work items associated with the finished process instance. This is the default selection.
  - Forever

    If selected, finished process instances are kept until you delete them from the Runtime database.
  - For

    If selected, you can specify for how long finished process instances are kept until they are deleted.

    Note: If a process is deleted, all associated work items are also deleted.

The cleanup server has several settings. The check interval setting is used by the administration server, which regularly checks the cleanup server after it is

started. This parameter determines how long to wait between checks. The server start/stop time can be set to automatic, manual, or as specified by a start and stop time. In addition, a cleanup busy time can be specified that determines how long the server is allowed to run. When the time limit is reached, the server is stopped. If it completes before the specified time, it stops automatically. A cleanup idle time can be specified as well, to specify how long to wait between runs for the cleanup server. There is only one cleanup server for each system group.

7.2 Planning the WebSphere Process Server topology

Once the WebSphere MQ Workflow environment has been assessed, the next challenge is to plan for a topology in WebSphere Process Server that is able to host a comparable environment.

This section provides information for understanding the aspects involved in planning for a WebSphere Process Server topology:

- General planning considerations
- Types of servers: The main WebSphere Process Server topology patterns, from stand-alone server to clustered environments
- Workload balancing, availability, and scalability
- Client types: A brief summary of available client types
- Database setup
- Authorization and staff resolution
- Cleanup of processes and tasks: How and when
- Integration of back-end applications (This topic is covered in Chapter 11, “Integrating back-end applications” on page 251.)
- Decision factors for choosing a topology

When you define the target topology, most likely it is not just a one-on-one match of the old environment. In many cases, new requirements require you to define a new environment that caters for these as well.

Section 7.4, “WebSphere MQ Workflow to WebSphere Process Server transition recommendations” on page 168, describes how the most frequently implemented topologies in WebSphere MQ Workflow can be transitioned to a corresponding WebSphere Process Server topology.

Note: This chapter focusses on the *planning* aspects of a transition. For detailed implementation information, refer to Chapter 13, “Implementing operational aspects” on page 357.

Appendix A, “Transition planning worksheets” on page 411, provides worksheets that may prove helpful during the transition planning phase.

7.2.1 Planning considerations

Planning for a WebSphere Process Server environment involves determining the number of physical computer systems that you will use, their performance and capacity, the performance and capacity of network connections, and how these elements will be associated, whether in a single cell, multiple cells, or whether they are standalone systems. Decisions must be made as to which systems will host the process servers and whether more than one server process will reside on a single system. You must determine how many messaging engines and which buses to set up. A database configuration also must be determined:

- The number of databases in which the required tables will reside
- Which systems will host the databases
- What Relational Database Management System (RDBMS) to use

In many cases, trade-offs must be made between the available hardware and physical connections, the complexity of management and configuration, and nonfunctional requirements such as performance, availability, scalability, isolation, security, and stability.

Application requirements are also a factor. Different applications have different needs, determined by such factors as the application modules' export types, component types, interactions between components, import types, resources needed such as databases or JMS resources, the need for business events, and their transmission mechanism.

When planning the target topology, the intended business use of the WebSphere Process Server environment needs to be considered. If it is used for testing applications (sometimes called unit testing), then a standalone server might be sufficient to meet these needs. If a networked, clustered environment will meet the business needs (perhaps for integration testing) then a multiple-node, clustered topology may be required, but one that contains fewer physical systems than the planned production environment. And for a pre-production test environment, we recommend creating a copy of the full environment that will meet the production needs.

The following sections cover the topics at the level of detail required to make the major planning decisions.

7.2.2 Types of servers

You can install the server on server hardware as individual stand-alone servers or as a managed group of servers.

- A *stand-alone server profile* has its own administrative console and all sample applications (if you performed a complete installation or selected to install the sample applications gallery feature during a custom installation). Each stand-alone server is fully operational and is managed independently from all other servers.

  There are two main types of stand-alone server installations:

  - One stand-alone server on one system
  - Multiple stand-alone servers on a single installation on one system.

- A managed group of servers, also known as a *network* deployment environment, uses a deployment manager for centralized administration tasks.

A deployment manager is a server that manages operations for a logical group, or cell, of other servers. The deployment manager is the central location for administering the servers and clusters. It manages the configuration for all of the managed nodes in its cell and the deployment of applications to selected managed nodes in the cell. All profiles in the cell share their configuration.

The main reasons to use managed nodes in a network deployment environment rather than using the same number of stand-alone servers are the centralized administration that the deployment manager provides, and that this type of setup is required for scalability, high availability, and workload balancing.

If you want to balance workload, such as service requests, over a set of servers, you can create a server cluster, then add servers as members of that cluster. You can also create a backup cluster to provide failover support for the server cluster to which it is assigned.

There are two main types of server cell installations:

- Deployment manager and managed nodes (cell) on a single installation on one system
- Deployment manager and managed nodes on a single installation on multiple systems

One stand-alone server on one system

This procedure gives you a stand-alone server profile named default and a server named server1. This profile is a separate data partition with files that define the stand-alone server environment, as shown in Figure 7-6.

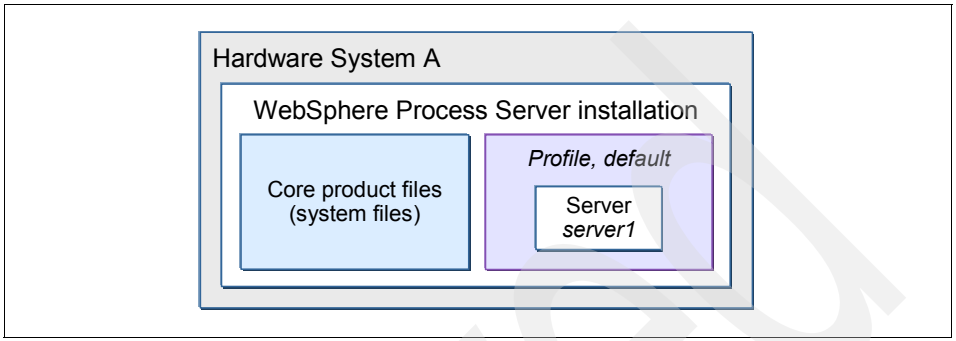


Figure 7-6 One stand-alone server profile on a single server

Multiple stand-alone servers on a single installation on one system

You can create several stand-alone server profiles in the same WebSphere Process Server installation.

Each profile has unique modules and applications, configuration settings, data, and log files. You can use multiple profiles to create separate server environments that you dedicate to different purposes. For example, each stand-alone server profile can be a separate test environment that you assign to a programmer or a development team.

Deployment manager and managed nodes (cell) on a single installation on one system

You can create a network deployment cell, a group of managed servers, on a single WebSphere Process Server installation. Use the Profile wizard and the deployment manager to create one or more custom nodes.

The *deployment manager* provides administration for all managed nodes in its cell. In a cell, modules and applications are run by the managed nodes, not by the deployment manager. Each managed node has a daemon process, called the node agent, used by the deployment manager to manage application servers on that node. You must start the node agent before you can start the servers. Figure 7-7 shows this type of setup.

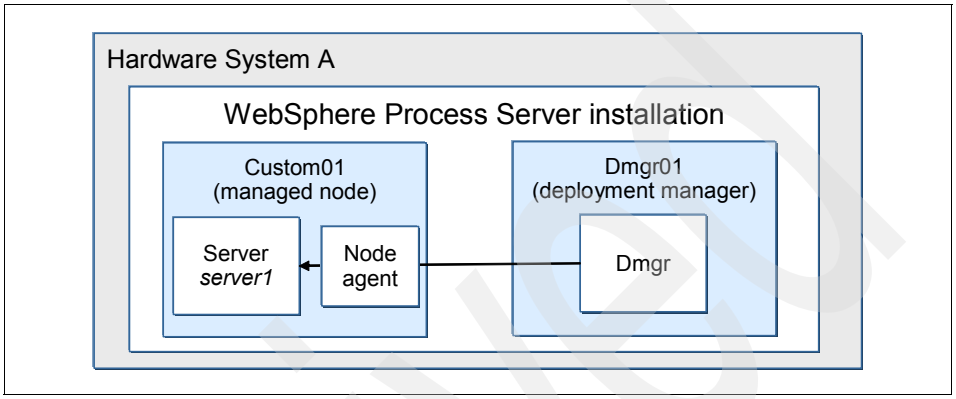


Figure 7-7 A managed node in a deployment manager cell

Periodically, the configuration and application files on a managed node are refreshed from the master copy of the files hosted on the deployment manager. This process is called node synchronization.

Deployment manager and managed nodes on multiple systems, with a server cluster

In a a network deployment cell, you can add managed nodes that are located on another system.

The primary advantage of a cell over a stand-alone server is single-point manageability. From one cell you can set up application interactions and also enable WebSphere Application Server clustering. Access is streamlined. For example, you can have a deployment manager located on one system, a managed node installed on another system, and a third system holding a managed node with a server cluster, as portrayed in Figure 7-8. Clusters are sets of managed servers that provide high availability and workload balancing for applications.

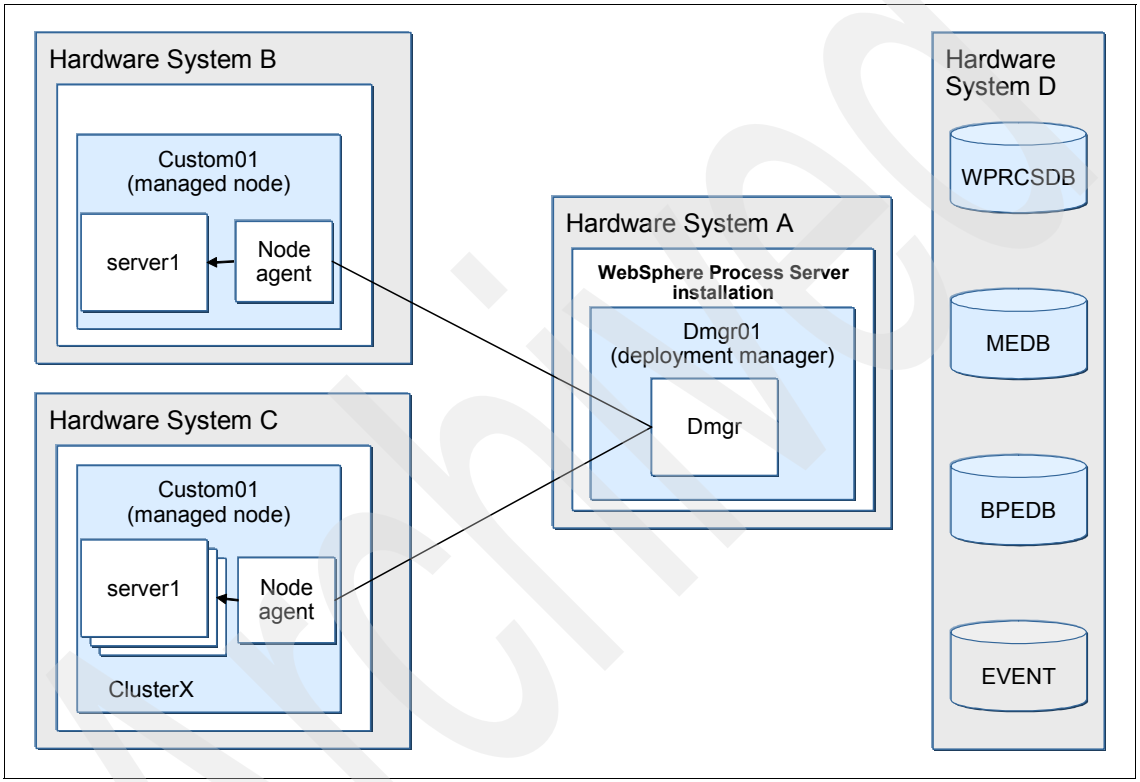


Figure 7-8 Several managed nodes in a multiple-server deployment manager cell

This type of setup with multiple hardware involved can scale in various ways. You can scale vertically by creating additional managed nodes, servers, and clusters on the same hardware. You can also scale horizontally by creating nodes, servers, and clusters on different hardware. Members of a cluster can be servers located on different hardware or servers located on the same hardware.

7.2.3 Workload balancing, availability, scalability, and failover

Clusters give your applications more capacity and availability than a single server. A clustered environment provides the following benefits:

- Workload balancing

  By running application images on multiple servers, a cluster balances an application workload across the servers in the cluster.

- Processing power for the application

  You can add processing power to your application by configuring servers as cluster members supporting the application on additional hardware.

- Application availability

  Availability of applications covers both high availability and continuous availability:

  - *Continuous availability* covers considerations for planned events such as application upgrades, infrastructure upgrades, hardware replacement, hardware relocation, reallocation of IPs, creating backups.
  - *High availability* covers unplanned events such as when a server fails, the application continues to process work on the other servers in the cluster, thereby allowing recovery efforts to proceed without affecting the application users. Clustering increases the system's availability by providing redundant Java Virtual Machine (JVM™) processes or hardware components that can ensure some level of continuity of service in case of failures.

- Scalability

  By adding servers to the cluster, you can easily scale the environment. Additional servers can be either on the same hardware, exploiting the existing processing power, or on different hardware, adding processing power to the cluster.

- Component failover

  Hardware and software components are doubled, which means that there are at least two instances of the same component in the system. In case of failure of one, the second takes over the load. In WebSphere MQ Workflow this is realized with WebSphere MQ clustering technology. In WebSphere Process Server, failover is ensured by setting up a clustered environment.

- System failover

  A standby or backup system is used to take over for the primary system if the primary system fails. This is realized with a clustered setup to cover the following aspects:

  - Automatic failover

    Monitoring the health of the system and automatic failover of the load to a secondary system.

  - Disaster recovery

    Similar to system failover, but the primary system and secondary system are at different geographical locations (primary site and backup site). The secondary system takes over the complete load in case of primary site loss. This implies duplicating every component. Disaster recovery complements, but does not imply, *high* availability. Time to activate the backup site is critical here.

The concepts of failover and scalability are largely independent. You may find that a topology that ensures scalability may not be very good at ensuring availability and vice-versa.

With WebSphere Process Server, there are many different ways to use clustering techniques to address availability and scalability. We strongly recommend that you become familiar with these techniques before setting up a production environment. Once you have decided on a particular cluster topology, some decisions made are irreversible. Going from one cluster topology to another is, in most cases, difficult or not possible. Section 13.3, “High availability, workload management, and scalability in WebSphere Process Server” on page 387, describes the options available to ensure high availability with WebSphere Process Server.

7.2.4 Client types

WebSphere Process Server offers the following clients:

- *Business Process Choreographer Explorer,* A Web-based client based on the Java API.
- *Business Process Choreographer Observer*, A Web-based light-weight event monitoring client based on the Java Common Event Infrastructure (CEI).
- A *Web-based business user client generator* that can be used to create clients. This generator is part of the WebSphere Integration Developer and can create JavaServer Faces (JSF) based clients, IBM Lotus Forms-based clients, or portlet-based client for WebSphere Portal

- *Business Process Integration Extension for IBM WebSphere Portlet Factory* for the development of Web-based or portal-based user interfaces for human tasks in business process management solutions.
- Customized Business Process Choreographer Explorer.
- Custom clients.
- Custom applications (for example, batch).

Details of the available client types and transition considerations for clients are discussed in detail in Chapter 6, “Planning for clients based on application programming interfaces” on page 115.

7.2.5 Database setup

A WebSphere Process Server environment makes use of databases. Several sets of database tables are used to manage WebSphere Process Server activities.

A WebSphere Process Server topology requires several types of database table sets, corresponding to the following:

- *Common* database tables
- *Event* database tables
- *Messaging* database tables
- Enterprise Service Bus Logging Mediation database tables
- Business Process Choreographer database tables
- Observer tables.

The common database tables are used by several WebSphere Process Server components and are a required part of any WebSphere Process Server environment. These tables manage the behaviors of some application components (such as mediations, relationships, or business rules) or management processes (such as failed events).

The event database tables are used by the Common Event Infrastructure server application. When monitoring WebSphere Process Server events you can capture event information. The event data is stored in the event database tables. A set of these tables is needed for each instance of the CEI server application that is included in the environment.

The messaging engine database tables are used to store messages that are associated with the asynchronous interactions of the applications and server components. These tables are a part of every WebSphere Process Server environment. They are used when the components and modules interact

asynchronously or when BPEL macro flows (long-running business processes) are running.

The Message Logger primitive records messages as they pass through a mediation flow. A mediation primitive can run in a new transaction so that even if the flow is rolled back a record is kept of the attempt. You can use a second Message Logger primitive inside the main flow of the transaction to keep a complete record of every message and whether it was rolled back. In order to use the Message Logger primitive you must have the appropriate database resources available.

Business Process Choreographer applications require database tables to manage the templates and instances of business processes and human tasks.

Business Process Choreographer Observer creates reports on processes that have been completed. You can also use it to view the status of running processes. The event collector application of the Observer reads event information from the CEI bus and stores it in the event collector table in the Business Process Choreographer Observer database.

Table 7-1 shows the database tables, the WebSphere Process Server environment components with which they are associated, and how many are required.

Table 7-1 WebSphere Process Server databases

Databases or tables	Default name	Usage
Common	WPRCSDB	► One per cell or standalone server ► Used for relationships, failed events, business rules ► Accessible by dmgr, all application deployment targets, business rules manager
Event	EVENT, CEIDB	► One per CEI server installation ► Used for storage of business events (may grow large quickly) ► Accessible by dmgr, CEI server
Enterprise Service Bus Logging Mediation Primitives	EsbLogMedDB	► One per cell ► Used for storing WebSphere Enterprise Service Bus mediation logging primitives ► Accessible by all deployment targets of the mediation module

Databases or tables	Default name	Usage
Messaging	MEDB	► One per messaging engine (active) ► Used for storage of messages ► Accessible by all potential messaging engines (active and standby)
Business Process Choreographer	BPEDB	► One per application deployment target ► Used for macroflows and business processes and human tasks ► Accessible by the local single server or the members of the local cluster.
Business Process Choreographer Observer	BPEDB	► For production, use a separate database ► Same deployment target as CEI server

The various sets of database tables are sometimes referred to as complete databases. This is not technically required. It is a planning decision as to whether to use a single database for all sets of tables (with separate schemas for the sets of tables) or use separate databases for each set of tables.

These decisions should be based on criteria such as the tuning capabilities of the database, how easy or difficult it is to configure the database, performance considerations, and scalability of the overall setup.

7.2.6 Authorization and staff resolution

The functional concepts of authorization and staff resolution are described in Chapter 4, “Planning for human interaction in business processes” on page 57. However, when planning the WebSphere Process Server topology, authorization and staff resolution may need to be planned from a topology point of view as well.

In WebSphere Process Server, a staff repository (also known as an enterprise, staff, or people directory) is the data store that contains the user and group information for human tasks. The most popular staff repository is the LDAP directory, which is based on the standardized Lightweight Directory Access Protocol.

Users interact either with the Business Process Choreographer Explorer or with another client based on the Business Process Choreographer APIs. These client applications send their requests to either the Business Flow Manager Process API or the Human Task API. Authentication is provided by the WebSphere Application Server security component. The instance-based and rule-based authorization is provided by Human Task Manager using work items.

When authorization management invokes the staff support service to resolve a staff query, it selects the suitable plug-in to which it delegates the staff resolution task. Staff resolution plug-ins are then responsible for retrieving the staff information from the external staff repository. Various types of staff repositories can be supported in this way.

Things to consider regarding authorization and staff resolution when planning the WebSphere Process Server architecture:

- You should use the same staff repository for Business Process Choreographer authorization that you use for WebSphere Application Server security and authentication (the one configured for the user registry). Otherwise, the two staff repositories might not always be synchronized, possibly resulting in authenticated users not being authorized to access business objects, or vice versa.
- If the overall WebSphere Process Server environment needs to be made highly available, make sure to plan for high availability of the database holding the staff information.

7.2.7 Cleanup of processes and tasks

WebSphere Process Server uses databases to store process instance data, as well as work items for human tasks. For long-running business processes, these databases enable the system to store the process state so that even in the case of a server being stopped, the process can resume where it left off rather than having to start again. A process template is stored in the database. When a process instance is started, this template is cloned into a process instance table. During execution, the process instance refers to the template. Only instance objects that are needed are stored in the database. Activity instances, for example, that are only needed inside one transaction are not persisted into the database.

When a process has completed or is terminated, the process instance data is no longer needed in the database. However, immediately removing all of the process instance data and all of the work item data takes time. As with WebSphere MQ Workflow, a heavily used system might want to defer this work to a time when it is less busy, such as off-shift or overnight. For this reason, the concept of cleanup was introduced.

WebSphere Process Server does not have a cleanup server as WebSphere MQ Workflow did. However, cleanup policies for WebSphere Process Server can be modeled in WebSphere Integration Developer on a process level and a task level:

- Process level: Define whether to automatically delete the process after completion. Business processes also offer the *only when completed successfully* option.
- Task level: Define the duration until the task is deleted (immediately or never), as well as the auto deletion mode (when completed, only when completed successfully).

Note: The cleanup of stand-alone task instances is independent of the cleanup of process instances. Maintain both to ensure that both are cleaned up.

The WebSphere Process Server Information Center provides information about scripts to administratively clean up processes and tasks:

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/tadmin_scripts.html`

The following maintenance topics are discussed there:

- Deleting audit log entries using administrative commands: Use the administrative commands to delete some or all audit log entries for the Business Flow Manager.
- Deleting process templates that are not valid: Administrative commands to delete, from the Business Process Choreographer database, process templates that are no longer valid.
- Deleting human task templates that are not valid: Administrative commands to delete, from the Business Process Choreographer database, human task templates that are no longer valid.
- Deleting completed process instances: Administrative command to selectively delete from the Business Process Choreographer database any top-level process instances that have reached an end state of finished, terminated, or failed.
- Deleting data from the observer database: Administrative command to selectively delete from the Business Process Choreographer Observer database, all of the data for process instances that match specified conditions. Deleting unnecessary data can improve the performance-generating reports.

- Removing unused people query results using administrative command: Administrative commands to remove unused people query results from the database.

For a more detailed discussion on cleanup functions in WebSphere Process Server refer to 13.5, “Cleanup of processes and tasks” on page 395.

7.3 Capacity planning

Capacity planning is the process of collecting details about what there is today and what is planned for the future. The result of the capacity planning process is a rough estimate of the amount of resources needed in the future to satisfy the expected processing demand.

Items to collect for an existing WebSphere MQ Workflow installation:

- System parameters:
  - CPU utilization
  - Memory usage
  - Disk busy utilization
  - A typical day's load profile
- Business parameters and business demand:
  - The number of process instances per day/hour in averages and peaks
  - The number of queries for process templates, process instances, work items per day/hour
  - Complexities of filters, staff assignments, authorizations
  - The number of concurrent users working with the system

The most basic rule for capacity planning is that if the business demand increases, the system resource utilization will increase accordingly.

7.3.1 Guidance for some scenarios

The following sections provide information about how to plan the capacity requirements for various transition scenarios.

Transition with almost no rework

If the goal is to keep the amount of planned rework as small as possible, then this means that you will need to continue to use message-based invocation of program activities. For WebSphere Process Server, this means using JMS or MQ

bindings and asynchronous interactions with the to-be-invoked programs and services.

For this scenario, more processing capacity will be needed, primarily due to additional processing within the JMS and MQ layers inside WebSphere. It is difficult to quantify how much additional capacity will be needed, and no direct comparisons can be provided here. It can be as low as 10–15% on top of what was needed before, but depending on the complexity of the solution, it might be more than that.

Transition with some rework

If the transition allows for changes to the model and possibly to the implementation of the invoked activities and services, the required processing capacity for WebSphere Process Server can be equal to or even a bit smaller than that needed for the WebSphere MQ Workflow solution.

Small changes to the model could be, for example, the conversion of asynchronous invocations of Web services or EJBs implemented using a UPES in WebSphere MQ Workflow, to synchronous invocations in WebSphere Process Server. Further optimizations could be achieved by, for example, adjusting the transaction boundaries in the BPEL model so that more than just one invocation is included within one BPEL transaction.

Transition with more rework

The best opportunity to reduce the processing capacity required for the solution is when parts of a process model or entire process models can be transformed to microflows (that is, short-running flows). In this case the processing is reduced to a single all-encompassing transaction, and intermediate process state updates are not persisted to the database. This further reduces processing cycles and time-consuming I/O.

The drawback to this solution is, however, that asynchronous processing within microflows cannot be done under the microflow's transaction control. Additionally, some remodeling must be done if the flow included human tasks for processing of exceptional situations in the business logic.

7.3.2 Infrastructure-related factors

In addition to the process models and the services implementations, there are other factors that can influence the required processing capacity.

Temporarily needed additional capacity

If the solution's nature allows for a transition from WebSphere MQ Workflow to WebSphere Process Server within a few hours, additional capacity is most likely

not needed. But when long-running WebSphere MQ Workflow flows must be *dried out* over a period of days and weeks while WebSphere Process Server is phased into production, additional processing resources may be needed to cover the capacity required to run WebSphere MQ Workflow and WebSphere Process Server in parallel during the phase-in/phase-out period.

Scaling and high availability

High availability in WebSphere Process Server can be achieved using a clustered setup in combination with horizontal and vertical scaling. Horizontal scaling is done across multiple boxes. Vertical scaling is done using multiple WebSphere Process Server nodes within the same (large) system. Depending on the machine resources and business requirements, horizontal and vertical scaling can be used in conjunction.

However, the machine that hosts the database should not be forgotten when planning for high availability of the overall environment. When one of the machines hosting the nodes of a WebSphere Process Server cluster becomes unavailable, the service as such is still available on the other nodes. This is not the case for the database. The machine that hosts the database must be backed up with high-availability means of the chosen platform. For a pSeries® machine, this could, for example, be achieved using High Availability Cluster Multi-Processing (HACMP™).

Choice of processors

Due to the nature of Java and J2EE-based processing it is advantageous to choose processors with large second-level caches. The cache sizes can have a considerably larger influence on obtainable throughput and response times than the clock speed of the CPU itself.

7.3.3 Considerations for database size

For practical purposes, questions about the database size to expect are more or less irrelevant. This may sound illogical, but is due to the fact that in order to get good I/O performance for the typical WebSphere Process Server I/O load, many dedicated physical spindles are needed. The reason is not the space they offer, but because they provide the number of physical read/write arms required to distribute I/O activity and reduce I/O contention that would occur otherwise. This also applies to SAN storage devices.

WebSphere Application Server transaction log, messaging engines' database tables, messaging engines' database logs, Business Process Choreographer's database tables, Business Process Choreographer's database logs, WebSphere Application Server correlation-related datastore—all these can become I/O hotspots depending on the processed volumes and the interaction scenarios. To

avoid these hotspots, in extreme cases, at least one dedicated physical drive must be considered for each of these entities. In some cases, groups of dedicated drives would even be required, combined via, for example, a striped file system or other means to effectively create a logical RAID-0 setup.

For development purposes such as unit test or functional test, a single disk setup is feasible. However, as soon as maximum throughput of an environment is required (for example, for stress tests or for high volume production) you will most likely need many dedicated disks to avoid I/O contention. The total space that they offer can easily amount to terabytes. However, only a small portion of it will probably be used for persistence purposes related to WebSphere Process Server.

Today's disks have at least 40 GB, 80 GB, or more space. Heavy-duty production-level disks are even larger. Typical large-scale databases for WebSphere MQ Workflow or WebSphere Process Server have sizes from 100 GB up to 250 GB. 250 GB would easily fit on a currently available typical single drive. Tests have shown that two million process instances use around 100 GB. This assumes that the amount and sizes of business objects (BOs) used as variables in a process are not excessively large.

The typical database sizes for Business Process Choreographer production databases, in terms of occupied space on disk, are a small fraction of the space, in terms of disk drives and number of disk drives needed for performance. Actually, up to 80% of the available disk space is *wasted* to get production-level performance.

All of the above applies to solutions where WebSphere Process Server can become I/O bound. If your processes use only in-memory processing using microflows, for example, then you will not run into size problems for the WebSphere Process Server database either, since the database for persisting states will not be large.

In summary, when thinking about database capacity, you do not need to worry about how large a production database for WebSphere Process Server has to be. It will definitely fit on the disk array that you must provide to ensure the required performance.

7.4 WebSphere MQ Workflow to WebSphere Process Server transition recommendations

Once you have assessed the environment in place in WebSphere MQ Workflow and are familiar with the functionality offered in WebSphere Process Server, the

next step is to plan for the target WebSphere Process Server environment, and how to get from one to the other.

Transitioning an existing WebSphere MQ Workflow environment to WebSphere Process Server can mean:

- Ensuring that you can transition the WebSphere MQ Workflow functionality to similar functionality in WebSphere Process Server.
- Designing a new environment in WebSphere Process Server that takes the old environment into consideration where necessary but is also designed to exploit new functionality and new requirements.

When designing the new environment, you must make sure that it not only caters to the current requirements, but that it can be scaled to meet future ones. With WebSphere Process Server, there are many different ways to use clustering techniques to address availability and scalability. We strongly recommend that you become familiar with them before setting up a production environment. Once you have decided on a particular cluster topology, some decisions made are irreversible. Going from one cluster topology to another is, in most cases, difficult or not possible.

Table 7-2 discusses the transition considerations for the most common WebSphere MQ Workflow server environments and describes what you must consider for production-scale functionality of the infrastructure.

Table 7-2 Transition scenarios for the server architecture

WebSphere MQ Workflow environment	WebSphere Process Server options	Considerations for transition
Single system	Stand-alone server A stand-alone server provides an environment for deploying Service Component Architecture (SCA) modules in one server process. This server process is the deployment target for your custom applications and includes, but is not limited to, an administrative console, the messaging support, the business rules manager, and the Common Event Infrastructure.	A stand-alone server is defined by a stand-alone server profile. You can deploy your own solutions to a stand-alone server, but a stand-alone server cannot provide the capacity, scalability, or robustness that is generally required of a production environment. For your production environment, it is better to use a network deployment environment of interconnected servers. It is possible to start off with a stand-alone server and later include it in a network deployment environment by federating it to a deployment manager cell, provided that no other nodes have been federated to that cell. It is not possible to federate multiple stand-alone servers into one cell. **Note:** A stand-alone server runs the messaging engine in the same JVM as the server process. This setup cannot be upgraded to a *golden topology*, where an application server cluster uses a remote messaging cluster.

WebSphere MQ Workflow environment	WebSphere Process Server options	Considerations for transition
Single system	Managed server A managed server is a server that is configured in a managed node. It provides a resource within the deployment environment that runs your applications.	To provide a robust, production-scale process server, configure a deployment environment of interconnected servers. In addition to the performance, availability, scalability, isolation, security, and stability characteristics that cannot be provided by a stand-alone server, a deployment environment of interconnected servers or clusters has the advantage that you can manage all the servers or clusters from a centralized deployment manager. If you do not need the full cluster functionality to begin with, create the clusters with one managed server. To scale the environment, you can add members to the cluster when needed. **Note:** A managed server runs the messaging engine in the same JVM as the server process. This setup cannot be upgraded to a *golden topology*, where an application server cluster uses a remote messaging cluster.

WebSphere MQ Workflow environment	WebSphere Process Server options	Considerations for transition
System group, all systems on the same hardware	Network deployment environment with server clusters. A cluster is a set of managed servers that provide high availability and workload balancing for applications.	Members of a cluster can be servers located on the same host (the same node). A clustered environment provides the following benefits: ► Workload balancing: By running application images on multiple servers, a cluster balances an application workload across the servers in the cluster. ► Processing power for the application: You can add processing power to your application by configuring server hardware as cluster members to support the application. ► Application availability: When a server fails, the application continues to process work on other servers in the cluster. This allows recovery efforts to proceed without affecting the application users. ► Maintainability: You can stop a server for planned maintenance without stopping application processing. ► Flexibility: You can add or remove capacity as needed by using the administrative console of the deployment manager. **Note:** This type of setup does not cater for hardware failure situations.
System group, systems on multiple hardware	Network deployment environment with server clusters.	To best achieve high availability and workload balancing, distribute the cluster members onto different host machines.
Two-tier architecture (database on same tier as Workflow servers)	Database on same tier as WebSphere Process Server servers or clusters, or both.	This type of setup is not recommended for production setups that must achieve high availability or performance. For these requirements, place the database on separate hardware.
Three-tier architecture (database on separate hardware)	Database on separate hardware.	We recommend this for production setups that must cater for high availability, flexibility, and performance.

Recommended reading

It is beyond the scope of this book to provide the detail of information required to set up the appropriate topology for the target environment. For this purpose, a number of in-depth publications are available:

- WebSphere Process Server Version 6.1 information center: Planning your deployment environment

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.wps.610.doc/doc/tins_inst_topologies.html

- WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern

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

- WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 2: My first WebSphere Process Server cluster

  http://www.ibm.com/developerworks/websphere/library/techarticles/0701_carlson-neumann/0701_carlson-neumann.html

- IBM Redbook Publication: *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413

  http://www.redbooks.ibm.com/abstracts/sg247413.html

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

  http://www.redbooks.ibm.com/abstracts/sg246688.html

8

Transition planning

This chapter covers the most important planning aspects and considerations of a WebSphere MQ Workflow to WebSphere Process Server transition project. It discusses the following:

- Transition project considerations
- Transition assessment
- Gap analysis
- Preparing for transition
- Transition
- Testing and deployment
- Pilot phase and rollout to production
- Best practices for a transition project
- Skill requirements and reference materials
- Links to education classes and further reading

8.1 Transition project considerations

This section summarizes the aspects to consider for an overall transition project. Such a project is complex and will most likely differ from customer to customer. This section therefore cannot provide an exact project plan or detailed information about sizing and topology. Rather, it provides a set of hints, considerations, and functional areas to assess when planning the transition of your environment.

Figure 8-1 shows the main steps involved in a transition project.

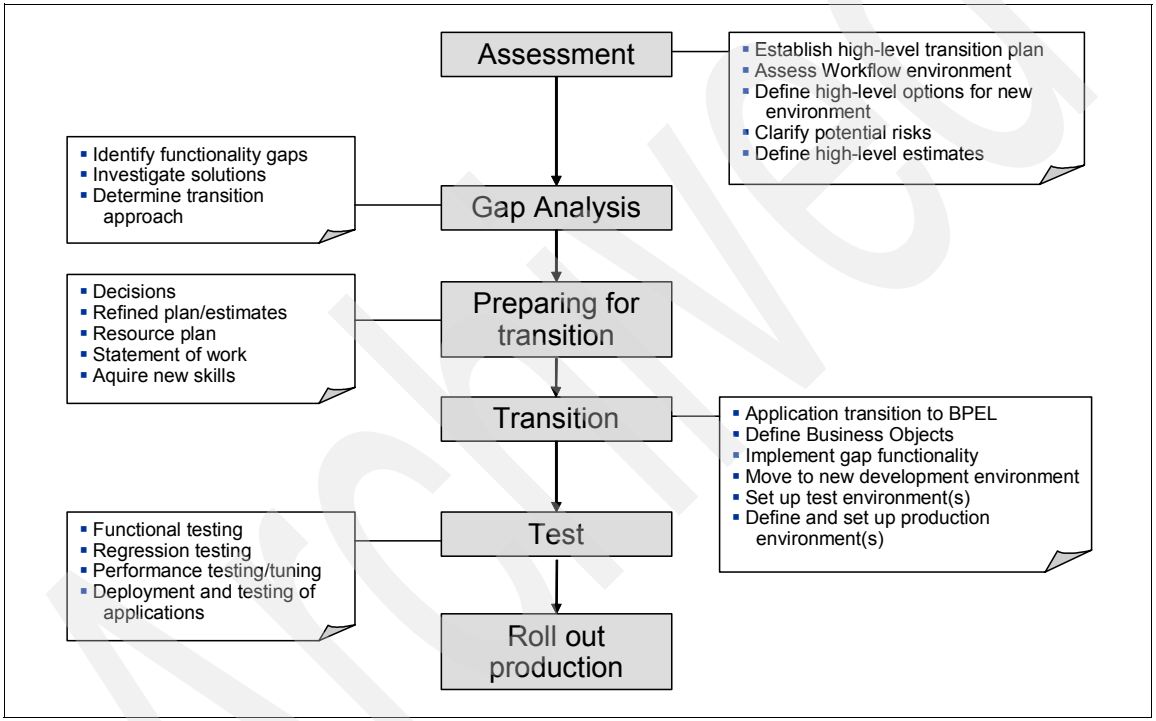


Figure 8-1 Transition project phases

Appendix A, “Transition planning worksheets” on page 411, provides a series of worksheets that help assess the WebSphere MQ Workflow environment, define the business requirements, and define the target business requirements, as well as the hardware and software environment.

Before introducing new software into your enterprise information system make sure that the system that you implement meets your needs. Detailed information about how and what to plan can be found in the WebSphere Process Server information center:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/doc/cpln_plancons.html

The information center covers the following topics to help you design a deployment environment to meet your needs:

- What are your business objectives and how can software help you achieve those objectives?
- What type of applications must you integrate?
- Do you want to eliminate duplicate information?
- What are the requirements for system response time and availability?
- What financial, hardware, software, and human resources are available for you to complete the installation?
- Do you require the services of other departments?
- Which tasks must be executed? Who will be responsible?
- Which hardware (existing or new) do you need for the installation?
- Can you use existing databases or do you require new databases?
- Can existing user IDs be used by WebSphere Process Server components or do you require new IDs? What authorization do new IDs require?
- Are there financial considerations that limit the number of product licenses you can purchase?
- How is your system going to evolve? For example, will it need to handle increased load or handle more concurrent users in the future? Will you need to add additional resources in the future to meet additional demands?
- Will your system have to dynamically add or remove resources to handle daily fluctuations on demand?
- Does your system have to support fluctuations in load or number of concurrent users on a periodic basis?

Also, think about your current goals.: Are you planning a test or production environment? Is it small-scale or large-scale? Do you want to set up quickly using default values or do you want to customize your environment?

The information center also provides planning suggestions for several different scenarios, depending on what you are trying to achieve.

8.2 Transition assessment

During the transition assessment phase, a series of reviews takes place to assess the areas of transition. At the end of this phase, a detailed understanding of the WebSphere MQ Workflow environment specifics has been documented, the target environment has been defined, and a transition plan has been established. In parallel, skill build-up is started.

The following reviews are carried out during this phase:

- Business and functional review:
  - Overview of business requirements
  - Should the existing processes be enhanced/changed
  - High-level description of integration scenarios
  - Potential scaling of the new environment
  - Availability requirements
  - Business impact

  Result: Document describing the usage profile.
- Architectural review:
  - Installation environment:
    - Hardware
    - Operating system
    - Software
    - Versions
  - External systems interaction: ERP systems, CRM systems, HR systems, SCP systems, Web-based applications, legacy systems.
  - Service Level Agreement (SLA): Discuss whether the same SLA is to be provided by the new environment.
  - Topology, availability, scalability, and workload management of the old environment.
  - Performance: Throughput, utilization, response times, capacity assessment, and others.
  - Security: Security zones, users, roles and groups, access rights.
  - Human task identification.
  - Workflow worklists, work item assignment, and staffing model.
  - Front end application (custom client, Web client) and API usage.

- Listing of interfaces and relation to business requirements/scenarios.
- Classification of interfaces by type (sync/async, real-time/batch, and so on).
- Process type analysis (duration: long versus short, complexity, functionality used).

Result:

- System context diagram showing the WebSphere MQ Workflow interfaces
- Preliminary transition risk/watch areas list

- Interface configuration and code review
  - Perform technical review of existing interfaces (human tasks, user interface application, flow control, relationships, long-running business processes, collaboration groups, server access interface, error handling, auditing, and so on).
  - Assess process complexity (collaboration templates, maps).
  - Custom code volume review (classes, packages, lines, and so on) for collaboration templates/maps, utility classes, custom data handlers.
  - Spot check code quality/maintainability.

  Result:

  - Transition plan.
  - Recommended transition approach.
  - Identify capability gaps between WebSphere MQ Workflow usage and WebSphere Process Server.
  - Component transition: Automated with no rework, automated with some rework, manual with redesign/re-implementation.
  - Risk area assessment and risk mitigation plan.
  - Identifies all issues found with their appropriate strategy for resolution.
  - Document identifying any issues that must be resolved or require design changes to the application.
  - Transition planning to compensate for long-running, in-flight WebSphere MQ Workflow processes.
  - Transition effort estimate (in effort days).
  - High-level project plan identifying tasks and time lines.

8.3 Gap analysis

The task that must be performed during the gap analysis is requirements gathering and analysis:

- Identify functional gaps and areas where there is no direct functional mapping from the old to the new environment.
- Investigate and determine possible solution alternatives in the WebSphere Process Server environment.
- Gather additional requirements from information technology (IT) and business departments and consolidate them with old environment assessment results. These additional requirements include changes in SLA, functionality changes that can be connected with the transition, technological changes, and others.

8.4 Preparing for transition

The following sections describe what to consider during the preparation phase.

8.4.1 Choosing the appropriate transition approach

A major decision to make is which transition approach or which combination of transition approaches can be used to move from WebSphere MQ Workflow to WebSphere Process Server in the specific project context.

There are several approaches to handle a transition of a WebSphere MQ Workflow system:

- Coexistence approach

  Coexistence is the means of running both environments, WebSphere MQ Workflow and WebSphere Process Server, in parallel for a defined period of time.

- Move-at-once approach

  This is an approach for handling long-running process instances by moving them at a certain point in time by stopping the work of the instances in the WebSphere MQ Workflow environment and starting respective instances in the new WebSphere Process Server environment.

For a detailed description of the concepts around model transition, process instance transition, coexistence of both environments, as well as a detailed mapping of features and functions, refer to Chapter 3, “Transition concepts” on page 29.

8.4.2 Environment planning

Within this phase your new environment is planned. Requirements obtained in the previous phase must be incorporated in the planning. Based on the results of this phase the new environment is prepared.

The following steps must be performed to achieve the expected results:

1. Document hardware and software topology of your new environment including hardware and middleware software components. The most important considerations to be made are:
   - Performance considerations
   - Availability and scalability considerations
   - Failover and disaster recovery considerations
   - Security considerations.
2. Prepare a list of required hardware and software upgrades.
3. Define the transition path, decide which components are transitioned one to one, and what needs to be customized and newly developed. Completely new modules may be also considered.

8.5 Transition

In the transition phase the environment is prepared and, depending on the transition approach, the transition is performed (in the development or test environment). All the necessary development and configuration work is performed in this phase.

Perform the following tasks in this phase:

- Environment preparation
  - Prepare the new environment (hardware infrastructure, data, middleware, user accounts, and others).
  - Set up the new tooling environment.
  - Move to the new Runtime environment.

- Application transition
  - Process modeling
  - Transition of applications from WebSphere MQ Workflow to WebSphere Process Server
  - Configuration work
  - Customization and new development
  - Preparation of additional transition scripts

8.6 Testing and deployment

In the following sections the different tasks that must be performed during the testing and deployment phase are described in detail.

8.6.1 Testing

Testing is performed in iterations together with transition until acceptance criteria are met. Testing ensures that the transition approach is correct and that applications work correctly in the new environment.

Perform the following tasks in this phase:

- Design test scenarios.
- Design and implement automatic tests.
- Perform tests.

Types of tests to be performed are:

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

8.6.2 Deployment

Deployment is probably the most complex phase of the transition. It is necessary to ensure business continuity, no data loss, and to ensure that it is possible to roll back in case of significant problems.

Perform the following tasks in this phase:

1. Environment backup

   Make a full backup of the production environment. This allows rollbacks in case of problems and ensures having a running production system.

2. Application transition

   Transition of production applications and deployment to the new (pre-production environment). Both the transition and deployment must be thoroughly tested and automated to a maximum degree. This automation helps to minimize time needed for transition and iterative testing.

3. Smoke testing

   Quick retest that the applications have been transitioned correctly.

8.7 Pilot phase and rollout to production

The last phase of the transition is the stepwise rollout to production. Depending on the size and complexity of the transition project and the overall environment, this phase may be split up into subphases.

8.7.1 Piloting

In the piloting phase, applications are running in the new environment for testing. In parallel, applications are still running in the old environment. Users are only working with the applications in the old environment.

Selected test users are using the new environment. This approach ensures the following:

- A non-functional pilot environment does not stop production. Your old environment can still be used.
- Problems in the pilot environment (bug fixes, outages) impact only test users.
- There is enough time to find most of the problems before going to production.

8.7.2 Rollout to production

At the end of the pilot phase, before going to actual rollout to production, the following criteria are met:

- Functional testing of the applications was completed.
- Testing was done on an infrastructure that matches the rollout infrastructure.

- Performance tests (peak load, limit-boundary, workload management, high availability, and so on) were completed and matched the defined criteria.
- A backup strategy is in place and has been tested.
- Users have been introduced to or trained on the new environment.

Some very important points to keep in mind for the production rollout are:

- Factor in time to drain the processes. Similar to other application transitions, parallel infrastructure may be required for a period of time. Process durations drive unique situations and duration of parallel infrastructure.
- Leverage extended license entitlement for transition. Installing base licenses with active subscriptions can activate entitlement to WebSphere Process Server and receive support for old and new product during transition.

8.8 Best practices for a transition project

Moving from a WebSphere MQ Workflow environment to a WebSphere Process Server environment can be a complex undertaking. To keep the complexity of a transition at a manageable level and within a manageable time frame, the following recommendations will help:

- Start building WebSphere Process Server (and WebSphere Application Server) skills now:
  - Pick a project to begin acquiring and building skills on the SOA platform
  - A transition from WebSphere MQ Workflow to WebSphere Process Server should not be the first project with exposure to new tooling, runtime, BPEL, and so on
- Start planning your transition now.
- It is most cost-effective to combine a transition with the next application or infrastructure change:
  - Next time you must open up and change your applications
  - Next time you upgrade the infrastructure.
- Identify the business objectives and value of the modified solution.

Enhance the application with capabilities not possible before. Leverage new functions:

- Calculate and add the time to drain the processes:
  - Similar to other application transitions, parallel infrastructure is required for a certain period of time.
  - The duration in which two parallel infrastructures must be maintained is affected by the duration of the processes.
- Leverage extended license entitlement for transition.

  Install base licenses with an active subscription can activate entitlement to WebSphere Process Server and receive support for old and new products during transition.

8.9 Skill requirements and reference materials

For a successful transition from WebSphere MQ Workflow to WebSphere Process Server, the following skills are assumed:

- You are familiar with WebSphere MQ Workflow V3.6 and the related products with the latest fix packs:
  - WebSphere MQ V6.0.
  - DB2 Universal Database V8.2
  - Oracle Database 10.1.0
- You have an understanding of the Eclipse and Java/J2EE platforms.
- You have a working knowledge of WebSphere Application Server V6.
- You have a ready environment with WebSphere Integration Developer and WebSphere Process Server V6.1 installed and configured.

For detailed information about these products, refer to the following documentation:

- WebSphere MQ Workflow V3.6
  - *IBM WebSphere MQ Workflow Administration Guide*, SH12-6289
  - *IBM WebSphere MQ Workflow Installation Guide*, SH12-6288
  - *IBM WebSphere MQ Workflow Getting Started with Runtime*, SH12-6287
  - *IBM WebSphere MQ Workflow Getting Started with Buildtime*, SH12-6286
  - *ÌBM WebSphere MQ Workflow Programming Guide,* SH12-6291

  The following URL provides these documents for download:

  `http://www.ibm.com/software/integration/wmqwf/library/index.html`

- WebSphere MQ V6.0

  WebSphere MQ System Administration Guide, SC34-6584

  http://www.ibm.com/software/integration/wmq/library/library6x.html

- DB2 Universal Database V8.2

  http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp

- WebSphere Process Server V6

  http://www.ibm.com/software/integration/wps/library/

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm

8.10 Links to education classes and further reading

For WebSphere Process Server, you can find a list of courses and self-guided training materials here:

http://www.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_list&subChapter=2079&subChapterInd=S&subChapterName=WebSphere+Process+Server

In addition, the following site provides self-study information about *in the spotlight* documentation, planned maintenance, education, and general self-help resources:

http://www.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&dc=DA400&uid=swg27008332&loc=en_US&cs=UTF-8&lang=en&rss=ct2307websphere

The IBM Education Assistant training on WebSphere Process Server and WebSphere Integration Developer can be found here:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wpi_v6/wpi6_coverpage.html

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wpi_v6/wpswid/6.0.2/Migration.html

A wide range of reference and education materials on WebSphere Process Server and Business Process Choreographer can be found on developerWorks here:

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

9

Business Process conversion

This chapter describes various approaches for converting the business process to the WebSphere Process Server environment. Examples are used to illustrate the value and limitations for each of the these approaches:

- Business model conversion
- Rewriting
- Using the FDL2BPEL Conversion tool
- Combining rewriting and using FDL2BPEL Conversion

In addition, a combination of these approaches is discussed.

9.1 Business model conversion

MQ Workflow Definition Language (FDL) is an execution language much like WS-BPEL. This can make it difficult to understand the business intent of a process because of the technical nature of an execution language. In some cases, the business process is initially expressed by creating a business model using a tool such as WebSphere Business Modeler. This allows a non-technical business analyst to specify what the process is, after which IT can add technical details and generate useful artifacts. If the process was first defined using a business model, the model itself can be used to create the new WS-BPEL implementation.

In this section, a simple credit request process is used to illustrate the different approaches to process conversion.

9.1.1 The initial business model

The initial business modeling was performed using Websphere Business Integration Workbench V4.2.4. Using this tool, a business analyst can create a process model using a non-technical basic mode. An IT resource can use integration mode to add technical details to the process, as shown in Figure 9-1.

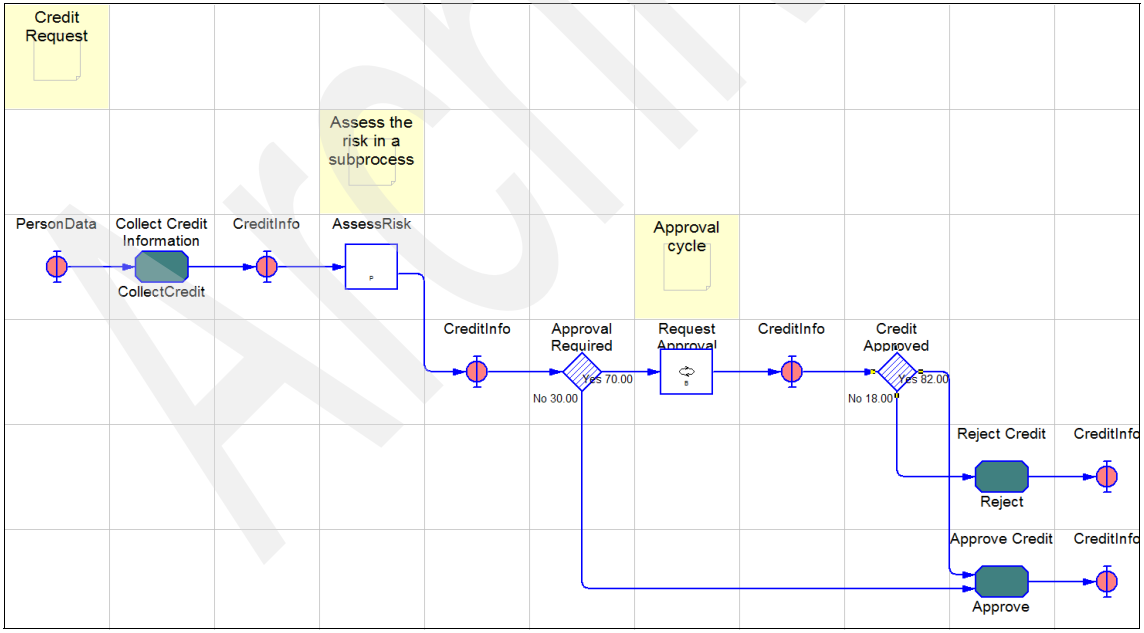


Figure 9-1 Initial business model

The Credit Request process contains many important process constructs including:

- Human tasks: Collect credit information, reject credit, and approve credit
- Subprocess: AssessRisk
- Block: Request approval
- Decisions: Approval required and credit approved

In addition, the AssessRisk subprocess contains an automatic step defined as a user-defined program execution server (UPES). These are representative of typical parts of a model that must be transitioned to the new environment. The model named CreditRequest.org is available from Appendix B, “Additional material” on page 433.

The business process model can be edited in integration mode, which allows additional technical details to be added into the model. For example, Websphere Business Integration Workbench V4.2.4 can model information about servers such as the number of execution servers to start, information about the WebSphere MQ queue manager and queue to be used by a UPES, and software version information. When the process is exported to WebSphere MQ Workflow, it is all contained in the generated FDL file named CreditRequest.FDL. See Appendix B, “Additional material” on page 433.

9.1.2 The initial MQWorkflow Definition Language implementation

The FDL file from Websphere Business Integration Workbench V4.2.4 can be deployed to a WebSphere MQ Workflow server, as shown in Figure 9-2. The file can be visualized by importing it into WebSphere MQ Workflow Buildtime.

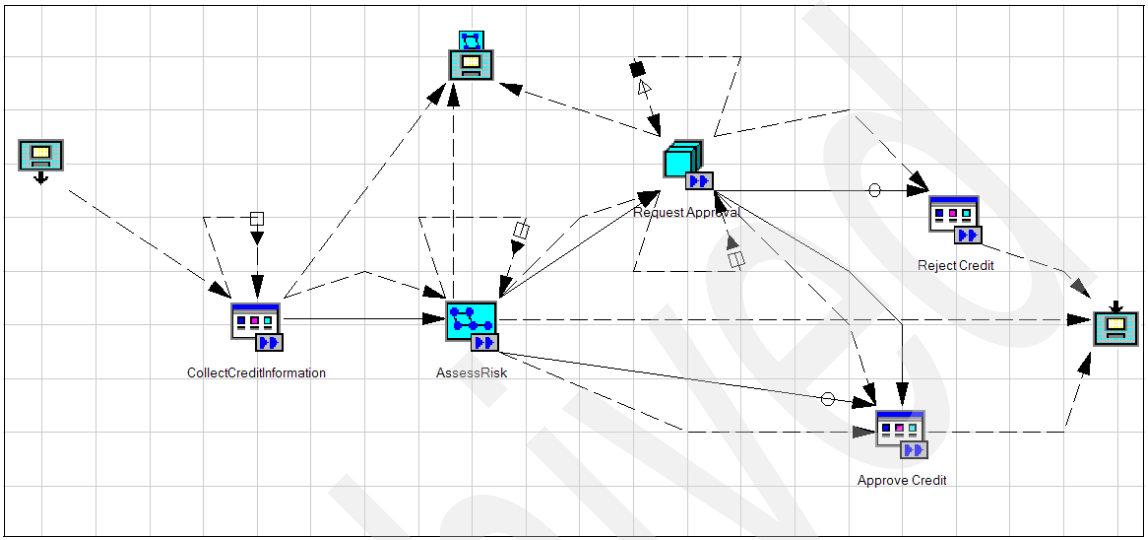


Figure 9-2 Credit request process in WebSphere MQ Workflow Buildtime

The process contains all of the technical information from the business model. However, business metrics such as how frequently each path of a decision is navigated is not included in the technical model, since an execution language such as FDL does not support such constructs.

In Buildtime, the flow of data is indicated with dashed lines, whereas the flow of control is shown with solid lines. Flow connectors with circles are called default connectors, and are used if none of the other flow connectors from an activity evaluate to true. There are special data flow connectors to show the mapping of an activity's input container to its output container, as well as for looping back data for repeating activities such as the block. Connections from some activities to the Global Data Container for the process are also visible in Buildtime. The business model only shows the flow of control, which is what a business analyst would be more concerned with. The technical model shows far more detail, which can obscure the business intent of the process.

If business process modeling had not been used, this is the technical model that would have been created. The different options for process conversion use these models as their inputs.

9.1.3 Transitioning the business model

WebSphere Business Modeler V6.1 is the successor to Websphere Business Integration Workbench V4.2.4. WebSphere Business Modeler can be used by non-technical analysts to specify the business process. A resource from IT can then add technical details into the model to prepare it for exporting. WBI Workbench could export a process into MQ Workflow Definition Language for WebSphere MQ Workflow, while WebSphere Business Modeler can also export processes to WebSphere Integration Developer, the tooling for WebSphere Process Server.

A model from Websphere Business Integration Workbench V4.2.4 can be imported into WebSphere Business Modeler V6.1 from the import menu. For more information about how to do this, refer to:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.btools.help.modeler602.doc/doc/tasks/migrating/importingadf.html

When performing the import, there is an option for mapping preference. It is important to select **non-MQ Workflow** for this option. If you do select MQ Workflow, the imported model is optimized for *exporting* to WebSphere MQ Workflow. However, the goal is to export into WebSphere Process Server, which means that non-MQ Workflow must be selected.

The model before clean-up

WebSphere Business Modeler V6.1 is a different product, with different modeling notation and constructs from WBI Workbench. An import of a .org file into WebSphere Business Modeler therefore requires some rework, as shown in Figure 9-3.

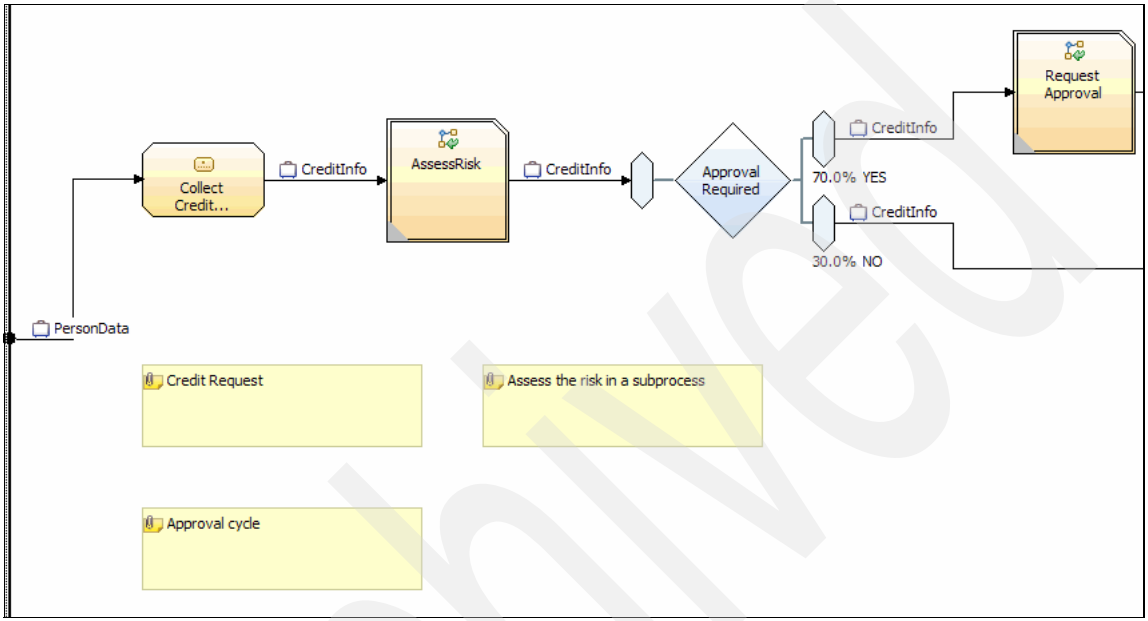


Figure 9-3 Imported process model: Part 1

Constructs from the WBI Workbench model have been mapped as closely as possible to constructs in WebSphere Business Modeler. For example:

- Phis: Phis, which represent the flow of data between tasks, are transformed into business items. Data structures for the phis are converted to attributes in the business items.
- Tasks: Tasks in the initial model remain as tasks in the imported model.
- Decisions: Decisions are transformed into multiple-choice decisions.

Limitations

Any tool that performs an automatic conversion may have limitations due to environmental differences. It is not unreasonable that an imported model requires some manual rework. For this scenario, several such limitations were discovered:

- A subprocess in WBI Workbench that is marked as a block is normally used for looping. A while loop would be a closer match, but the import creates a global process.
- Bulk resources were added to each of the tasks, with the name of the program defined in the original model.
- Role requirements were not set with a resource definition of *Staff*.
- Some of the tasks did not retain their role information from the original process.
- Expression® logic from the decisions does not flow into the new model. It needs to be recreated by hand.
- Many of the objects had the minimum number of inputs and outputs set to 0 rather than 1 as required.
- Two paths flow into *approve credit,* rather than using a merge before this task. When exported, a task with multiple inputs generates a WSDL with multiple operations and requires complex logic in the business model.

Cleaning up the model

The business model should show the business intent of the process, so the model should be corrected before it is exported to WebSphere Integration Developer. Otherwise, the IT staff would have to try and understand what the process is supposed to do at a business level. This responsibility falls on the business analyst, not the technical resource.

However, the model created by the import process required extensive modifications before it could be exported to WebSphere Process Server V6. These modifications required knowing some of the most complex techniques in WebSphere Business Modeler and could only be accomplished by a technical resource.

In order to clean up the business model:

- Tasks were resized to make their names readable.
- The bulk resources were deleted, since they were not needed.
- Role requirements were added to tasks where required, and all were set to a resource requirement of *staff*.

- *Request approval* was replaced by a while loop. The task inside was moved into the while loop via copy and paste. The empty request approval process was then deleted.
- The while loop required additional process modifications in order to be able to both simulate and to export properly. For more information about this, refer to the publication *Using Loops in WebSphere Business Modeler v6 to improve simulations and export to BPEL* available at:

  http://www.ibm.com/developerworks/websphere/library/techarticles/0703_fasbinder/0703_fasbinder.html

  In addition, a map node was required in order to flow data into the repository. Without the map, the *assess risk* subprocess would have to be modified, which could impact reusability.
- *Assess risk* and *credit request* were edited to set the minimum number of inputs and outputs to 1, as well as to remove the second input criteria from the Input Logic tab.
- The required number of inputs and outputs was set to 1 for other objects in the process.
- For each decision on the input tab, the minimum number of inputs required was set to 1, and the expression logic was defined.
- A merge activity was added before the approve credit step. Approve credit was modified to have one single input. This reduced the complexity of the task and results in a generated WSDL with one single operation.

Every object in the model needed to be extensively edited in order to be useful. It would often be easier to replace an object, rather than to clean it up to the point where it was useful. For this reason, the team does not recommend using this approach.

9.1.4 Migrating an MQ Workflow Definition Language (FDL) model

WebSphere Business Modeler V6.1 has the capability to import FDL files. This enables users to import processes, even if the original model was created in WebSphere MQ Workflow Buildtime.

The importer only works with FDL files that do not contain information about topologies. If the FDL file includes server information, the entire file is ignored. Make sure to deselect server information when exporting from WebSphere MQ Workflow Buildtime.

The model before clean-up

The process model that is created in WebSphere Business Modeler V6.1 requires some work before it can be used, as shown in Figure 9-4.

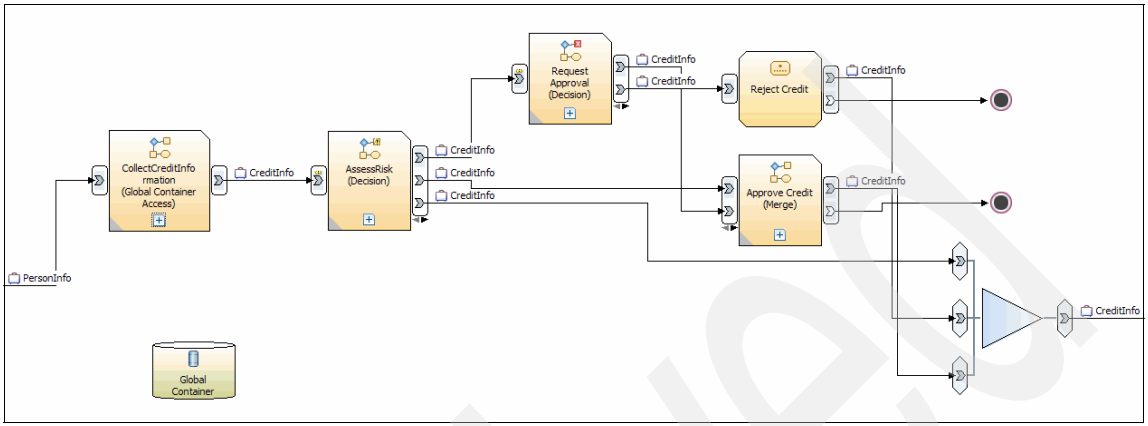


Figure 9-4 Initial process model

Each of the objects in the original FDL process is represented in the business model. The importer tries to closely match the semantics of the steps, resulting in a process that is more complex than it would be if it was modeled from scratch.

For example, examine the first step, CollectCreditInformation. A local subprocess is created named *CollectCreditInformation (Global Container Access)*. This name contains useful information about the technical model, which is not relevant to a business analyst. This local subprocess contains another local subprocess named *CollectCreditInformation (Default Data Connector)* as well as a map activity. See Figure 9-5.

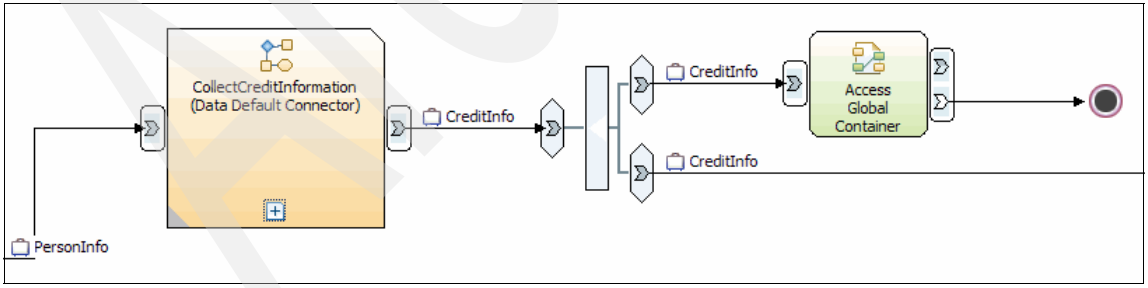


Figure 9-5 CollectCreditInformation (Global Container Access) local subprocess

The fork to a parallel path shows that the flow goes along the bottom connector, while there is a separate flow of data to the global container. These are the

outputs of the local subprocess CollectCreditInformation (Default Data Connector). This local subprocess can be expanded, as shown Figure 9-6.

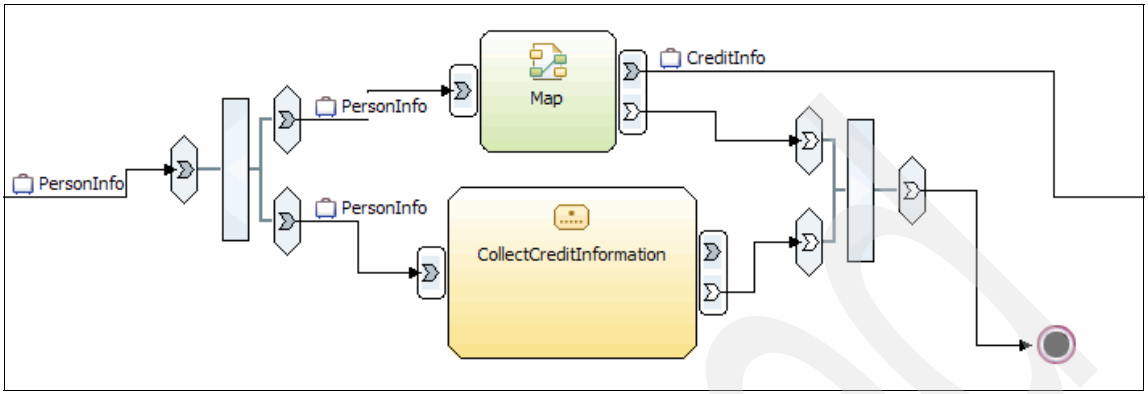


Figure 9-6 CollectCreditInformation (Default Data Connector) local subprocess

In this second level local subprocess is the actual task itself. The task is in parallel with a map. The default data connector moves information from the input container to the output container, which the task can then overwrite. Using a map in parallel with the activity attempts to preserve this functionality.

The CollectCreditInformation task is properly set up with the role requirement specified in the FDL, with a resource type of staff. The task could be useful since it preserves information from the original process. However, each local subprocess is transformed into a WS-BPEL scope activity, while maps are transformed into Java snippets. If this process is exported to WebSphere Process Server V6, the resulting WS-BPEL would be overly complex and would require extensive cleanup before it could be run.

The next section of the process (Figure 9-7) is a local subprocess named *AssessRisk (decision)*. This name is somewhat misleading, since AssessRisk was a subprocess in the original process, followed by a flow connector with a transition condition and a default connector. By having the decision within this local subprocess, the high-level process flows exactly like the original process in FDL. However, it makes the business logic more obscure, since you must drill down to discover it. The expressions for the decision outputs are automatically set based on the original model.

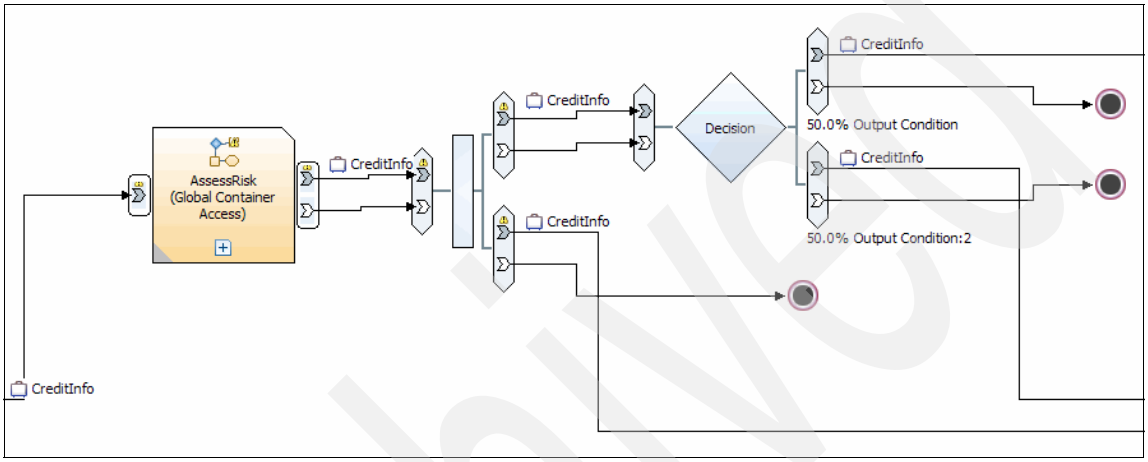


Figure 9-7 AssessRisk local subprocess

As with the first step, AssessRisk (Global Connector Access) contains a local subprocess to represent the default data connector. Within that local subprocess is a link to the global process AssessRisk. A parallel connector is included even though no map is generated. See Figure 9-8.

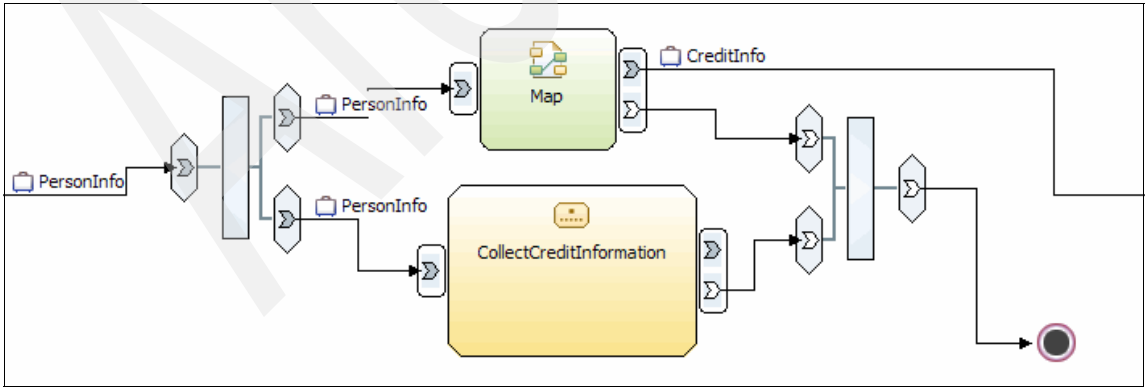


Figure 9-8 AssessRisk (Default Data Connector) local subprocess

The AssessRisk process contains a local subprocess for the default data connector. Within that local subprocess is a task representing the automatic user-defined program execution server (UPES) in the process. The task contains no implementation information in the technical attributes.

The FDL block activity for request approval is mapped into a loop in the model. However, a do-while loop is used, which is not supported in WebSphere Process Server V6, because WS-BPEL only specifies a while loop. In addition, many levels of nesting were created:

- The RequestApproval task is contained within a local subprocess named RequestApproval.
- This local subprocess is contained in another local subprocess named Request Approval (Data Default Connector).
- This local subprocess along with two map activities is contained within the do-while loop named Request Approval (Exit Condition Loop).
- The do-while loop, along with a local repository and a map is contained within another local subprocess named Request Approval (Exit Condition Structure).
- This subprocess along with a map is contained in a local subprocess named Request Approval (Global Container Access).
- This subprocess along with a decision is contained within a local subprocess named Request Approval (Decision).

For the human tasks in the process, a warning is issued, recommending converting them from tasks to human tasks so that further details can be defined.

Limitations

There are several limitations to the FDL import:

- Some constructs were created with the minimum number of inputs set to 0 instead of 1. These needed to be corrected for the model to be usable.
- Objects are named with their previous names and their object type or data connector type in parenthesis. Most likely, these names must be changed in order to be meaningful to a non-technical user of WebSphere Business Modeler.
- Steps and decisions are combined into local subprocesses together, obscuring the business intent of the process.
- One level of local subprocess is created for each data connector type used.
- User-defined program execution server (UPES) activities generate tasks with no implementation information.
- Extra fork activities are added even when not needed.

- Blocks resulted in a task nested six levels deep within subprocesses and a do-while loop, which is not supported in WS-BPEL. The loop did not contain an exit condition.
- Decisions map default connectors as a condition of true. There is no concept of *else* for a multiple-choice decision in WebSphere Business Modeler. An expression must be defined for branches set to true.

Cleaning up the model

Most of the subprocesses and other constructs created by the import are not necessary for a process to be exported to WS-BPEL. A new process can be constructed using tasks and decisions salvaged from the import. Performing this work requires a user with a good level of technical skills. See Figure 9-9.

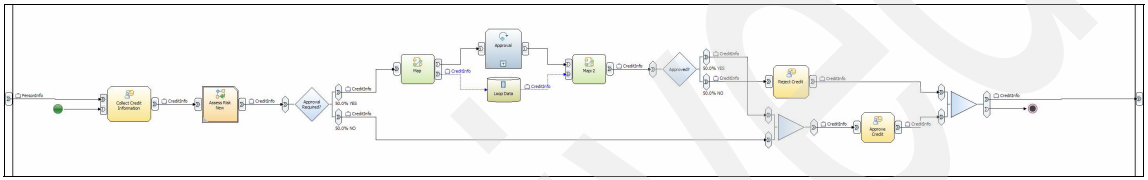


Figure 9-9 The business model after clean-up

To create the new process:

- Two new processes were defined:
  - Credit Request New
  - Assess Risk New
- Tasks and decisions were cut and pasted from the imported process to a new version of the processes. Manual tasks were converted to local human tasks.
- Extra outputs were deleted from the tasks and decisions.
- A while loop was added in place of the generated do-while loop. The loop condition was created and a repository was required to flow data into the loop. For more information about this refer to the publication *Using Loops in WebSphere Business Modeler v6 to improve simulations and export to BPEL* available at:

  http://www.ibm.com/developerworks/websphere/library/techarticles/0703_fasbinder/0703_fasbinder.html

A map was needed before the loop because the data must flow into the repository from the global process *Assess Risk New*. Adding a second connector to the global process could limit its reusability, so a map was used. Likewise, a map was used after the loop so that there could be a single connector flowing into the decision.

- Expressions were defined for the decision paths corresponding with the default connectors in WebSphere MQ Workflow.

The cleaned-up model named *Credit Request 61 Initial.mar* can be accessed in Appendix B, “Additional material” on page 433.

The model was exported to WebSphere Integration Developer as a project interchange. For the purposes of illustration, all artifacts were placed in a single module. The project interchange named ProjectInterchange_1193674296515.zip can be accessed in Appendix B, “Additional material” on page 433.

For information about the invocation of the user-defined program execution server see Chapter 11, “Integrating back-end applications” on page 251.

9.1.5 Optimizing the generated WS-BPEL

The process can be exported to WebSphere Integration Developer V6 from WebSphere Business Modeler V6.1. The business process is transformed into WS-BPEL along with other artifacts such as XSDs for the data and WSDLs for the interfaces for each task. See Figure 9-10.

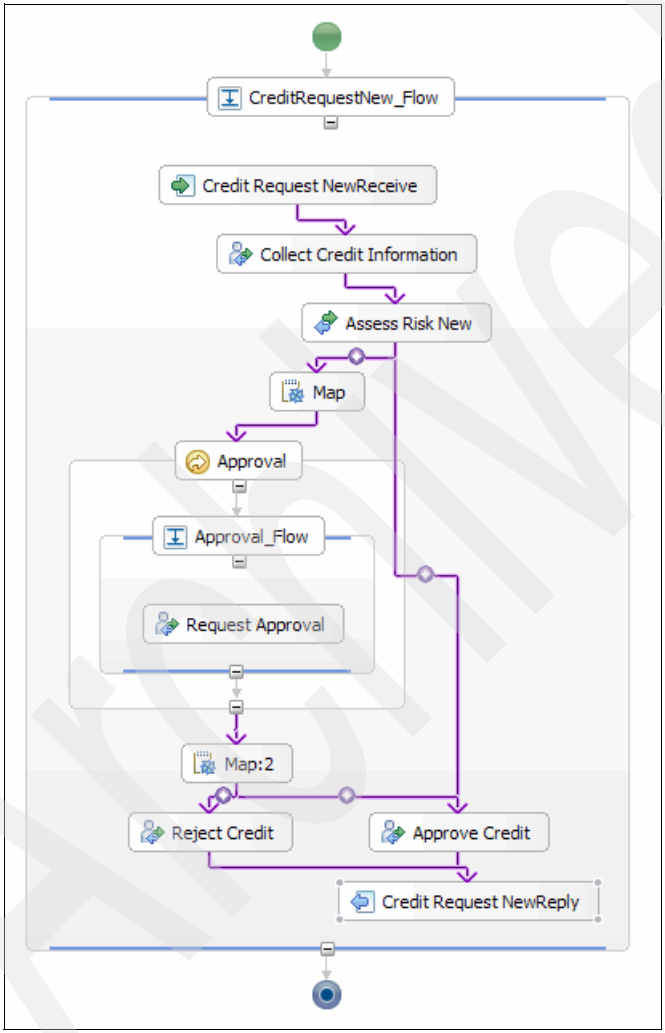


Figure 9-10 Credit Request New WS-BPEL process in WebSphere Integration Developer

The generated WS-BPEL for the process is close to being usable. The map activities are not needed in this case and could be deleted. Note that the

connectors flowing out from Map:2 contain branching logic. In order to preserve the logic:

1. Delete the connector from the approval loop to Map:2.
2. Move each of the connections flowing out from Map:2 so that they flow out from approval instead.
3. Once the connections are moved, you can delete Map:2.

If you just delete Map:2, the connectors flowing out from it would also be deleted. Using this procedure preserves the flow logic. See Figure 9-11.

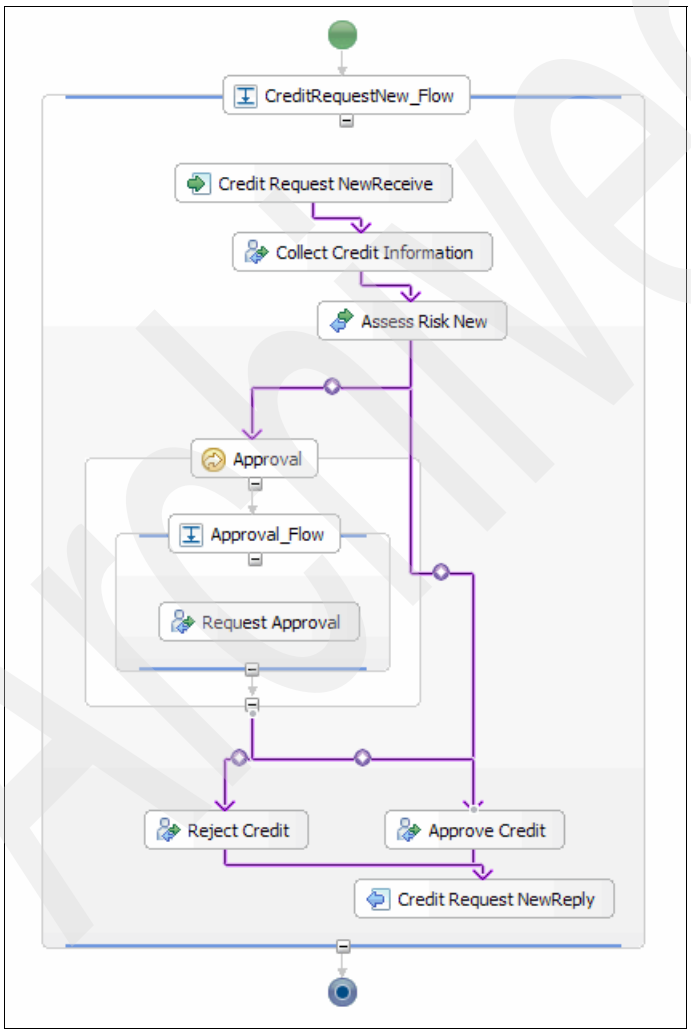


Figure 9-11 Credit Request New process after clean-up

For each of the activities in the process, there is an important setting to consider called *Continue On Error*. For each activity in the process, the generated WS-BPEL has the Continue On Error box checked. This is the default when an activity is created. See Figure 9-12.

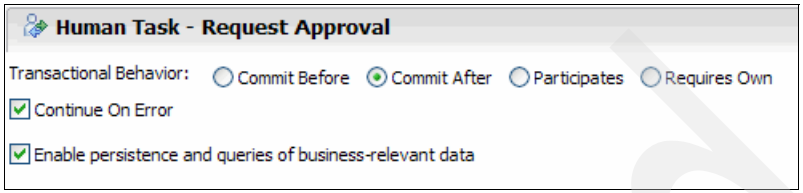


Figure 9-12 Request Approval server settings

When Continue On Error is checked, any faults that occur are sent to the fault handler of the scope enclosing the activity. If that fault handler cannot deal with the fault, it is thrown back to the next higher level. If the fault reaches the outermost scope and it is still not handled, the process terminates. This behavior is different from WebSphere MQ Workflow.

If Continue On Error is unchecked, the behavior is much closer to WebSphere MQ Workflow. If a fault is thrown and there is no fault handler for the enclosing scope, then the process transitions to a *stopped* state and a work item is created for the process administrator. For details see:

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/tasks/cconerr.html`

At this point, a critical decision must be made. Should the new process follow the behavior of WebSphere MQ Workflow? Or should new functionality in WebSphere Process Server be leveraged, such as fault handlers? Depending on the answer, the process may need to be updated. See 3.2.2, “Determining WebSphere Process Server enhanced functionality” on page 49, for more information.

A Service Component Architecture assembly diagram was also generated for the module. The Credit Request New process invokes the Asses Risk Factor process, as expected. In the business model, no implementation information was added for the Assess Risk UPES step, so a component with no implementation type was generated with the proper interface. See Figure 9-13.

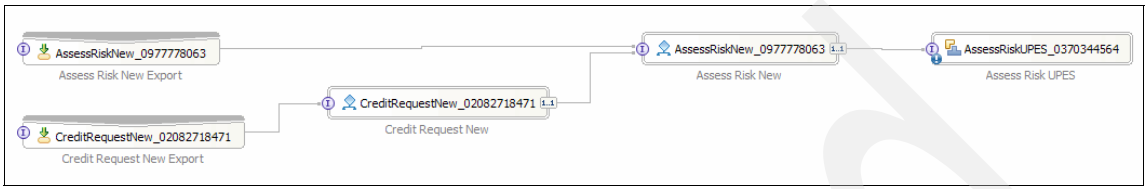


Figure 9-13 Generated Service Component Architecture assembly diagram

This technique resulted in a usable business process. There were some advantages:

- The results included both a business model and an execution model.
- Information from the original model was preserved, such as data types, staff assignments, and decision logic.
- A correct Service Component Architecture assembly diagram was generated.

There were some disadvantages:

- The MQWorkflow Definition Language file could not contain topology information. A new export is required, or hand-editing to remove the non-supported constructs.
- Cleanup of the model was required at the business modeling level, as well as at the execution model level.
- The expressions for some of the conditions were more complex than necessary. Since the original process used default connectors, the optimal expression would be to use *simple* as the expression language and select **Otherwise**. This had to be done manually.
- The useful parts of the imported process in WebSphere Business Modeler were often nested several levels deep.
- More WS-BPEL variables were created than were needed. If an overly large number of variables are used, the process is not as efficient as possible.

9.2 Rewriting

Instead of migrating an existing model, one valid option is rewriting the process from scratch. This could be done at the business modeling level or at the execution model level depending on business needs.

In order to perform this technique, both WebSphere MQ Workflow Buildtime and WebSphere Integration Developer (or WebSphere Business Modeler) must be open concurrently so that constructs from the existing tool can be recreated in the new tool.

For the Credit Request process, the following steps were performed to rewrite the process in WebSphere Integration Developer:

- A data type was defined in WebSphere Integration Developer to match the predefined data structure from Flow Definition Language.
- For each data structure in WebSphere MQ Workflow Buildtime, a data type was created in WebSphere Integration Developer. These new data types included a reference to the predefined data type as well.
- For each task in the original process, an interface was defined in WebSphere Integration Developer referencing the data types.
- A process was created for Assess Risk, along with an interface for the process. The process was created as being long-running and autonomy of peer.
  - An interface partner and a reference partner were created using the appropriate interfaces.
  - A WS-BPEL variable was created based on the CreditInfo data type.
  - An invoke WS-BPEL activity was added to the process and its properties set.
  - The process was saved then dropped onto the Service Component Architecture assembly diagram.
  - A component with no implementation type was added to the Service Component Architecture assembly diagram, then linked with the reference from the Assess Risk process.

- A process was created for Credit Request. Steps similar to the ones above were used, as well as:
  - An additional variable was required because the first step takes in data of one type, then outputs data of a second type.
  - Expressions were added to the links as necessary, based on the expressions in the transition conditions from the original Flow Definition Language model.
  - A WS-BPEL while loop was added, along with a loop condition based on the exit condition from the Flow Definition Language block.
  - Human tasks were added into the process. Their parameters were set based on the information from the staff activities in the original Flow Definition Language process.
  - The process was saved, then added to the Service Component Architecture assembly diagram. The reference from Credit Request was linked to the interface of the Assess Risk process.
  - An export was created for the Credit Request process using Service Component Architecture binding and linked to the interface of the process.
- The completed processes could then be tested.

There were some advantages to this technique:

- Only one tool was required.
- The resulting model did not require a clean-up effort.
- Fewer variables were created than with the imported version of the process.
- Logic for the expressions was not as cumbersome as the imported version.

There were some disadvantages to this technique as well:

- Everything needed to be entered by hand.
  - This poses the danger of introducing errors, such as selecting an incorrect data type or missing a field of a large data structure.
  - For a larger process it could become rather tedious to recreate every single artifact by hand.
  - The person recreating the process needs deep knowledge in both WebSphere MQ Workflow and WebSphere Process Server.
- No business model was created, only a technical execution model.

9.3 Using the FDL2BPEL Conversion tool

One last option for model migration is to use the FDL2BPEL Conversion tool to convert the technical execution language from Flow Definition Language to WS-BPEL. The FDL2BPELConversion tool is built into WebSphere Integration Developer. The documentation is available as a SupportPac at:

http://www.ibm.com/support/docview.wss?rs=795&uid=swg24008362

Before you use this tool, you should spend some time reading through the documentation so that you understand different considerations in the conversion process. For example, when first importing your FDL file, there are two important options that you must consider:

- Treat name conflicts as errors. FDL did not restrict you from having multiple constructs with the same name. For example, the Assess Risk UPES uses a program name AssessRisk, which is the same as the name of the process. If you select the option to treat name conflicts as errors, you are warned of the conflict so you can update the FDL, then import again. If you leave this option unselected, FDL2BPEL handles the name conflict by appending a number to the generated name, such as AssessRisk01.
- Create predefined data members. Along with the data that you define in your process, FDL includes predefined data members such as _PROCESS_MODEL and _ACTIVITY. If you do not need the predefined data members, you can deselect this option. The WS-BPEL created is more efficient, since there are no variables for the predefined data members.

9.3.1 The value of FDL2BPEL

Using the FDL2BPEL tool, there are several advantages that can be gained:

- As discussed in 9.2, “Rewriting” on page 205, if you recreate the process by hand, there is a risk that you can miss a detail or enter something incorrectly. FDL2BPEL ensures that your data types are defined correctly.
- Using FDL2BPEL is much faster than redefining all of the objects by hand or converting the business model, enhancing, then exporting.
- FDL2BPEL is built into WebSphere Integration Developer, enabling you to do all of the work on one single tool.
- Of all the techniques for model conversion, FDL2BPEL results in WS-BPEL that most closely matches the process functionality in WebSphere MQ Workflow and MQWorkflow Definition Language.

- FDL2BPEL automatically generates code for the data bindings needed to serialize and deserialize JMS messages. No user programming is required. This enables you to use your existing UPES implementations unchanged.
- FDL2BPEL does not require deep knowledge of WebSphere MQ Workflow to use. It also does not have any restrictions on the MQWorkflow Definition Language, as the import from WebSphere Business Modeler has.

9.3.2 Artifacts created by FDL2BPEL

The file CreditRequest.FDL from 9.1.1, “The initial business model” on page 188, was imported into WebSphere Integration Developer. The tool created several artifacts. A project interchange for WebSphere Integration Developer named FDL_Import_Initial.zip is available in Appendix B, “Additional material” on page 433.

Data types

A number of data types were created by the FDL2BPEL tool, as shown in Figure 9-14.

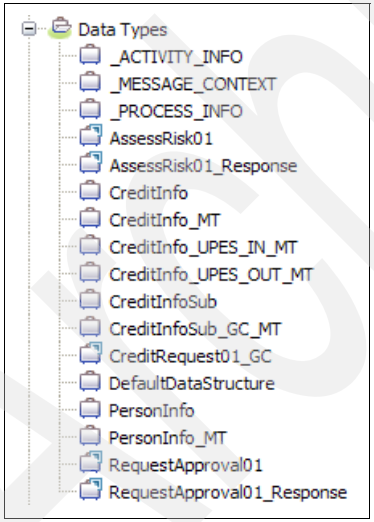


Figure 9-14 Data types

For each of the Flow Definition Language data structures, a corresponding data type is defined, such as PersonInfo. If you do not deselect the Create predefined data members option, a second data type is defined with the structure of an WebSphere MQ Workflow data container, such as PersonInfo_MT. The PersonInfo_MT data type is more complex than PersonInfo because it also contains the predefined data members, but it could be very useful in that

WebSphere MQ Workflow constructs such as exit conditions often refer to variables such as _RC. However, this additional information could add overhead if it is not used. See Figure 9-15.

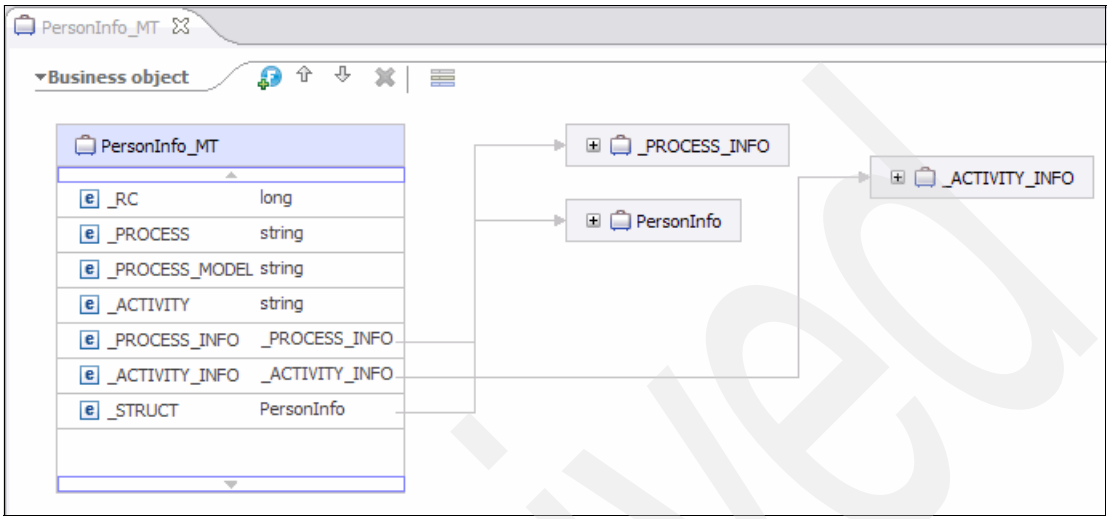


Figure 9-15 PersonInfo data type

The Credit Request process did not use the predefined data members in the FDL, so the option to create them was not selected, resulting in fewer data types, fewer variables, and a more efficient process.

For the user-defined program execution server (UPES) step in the process, another data type named _MESSAGE_CONTEXT was created. This data type includes all of the information in a user-defined program execution server message. This data type along with the automatically generated data binding code enables you to use your existing UPES implementation without having to rewrite it.

Figure 9-16 shows the _MESSAGE_CONTEXT data type.

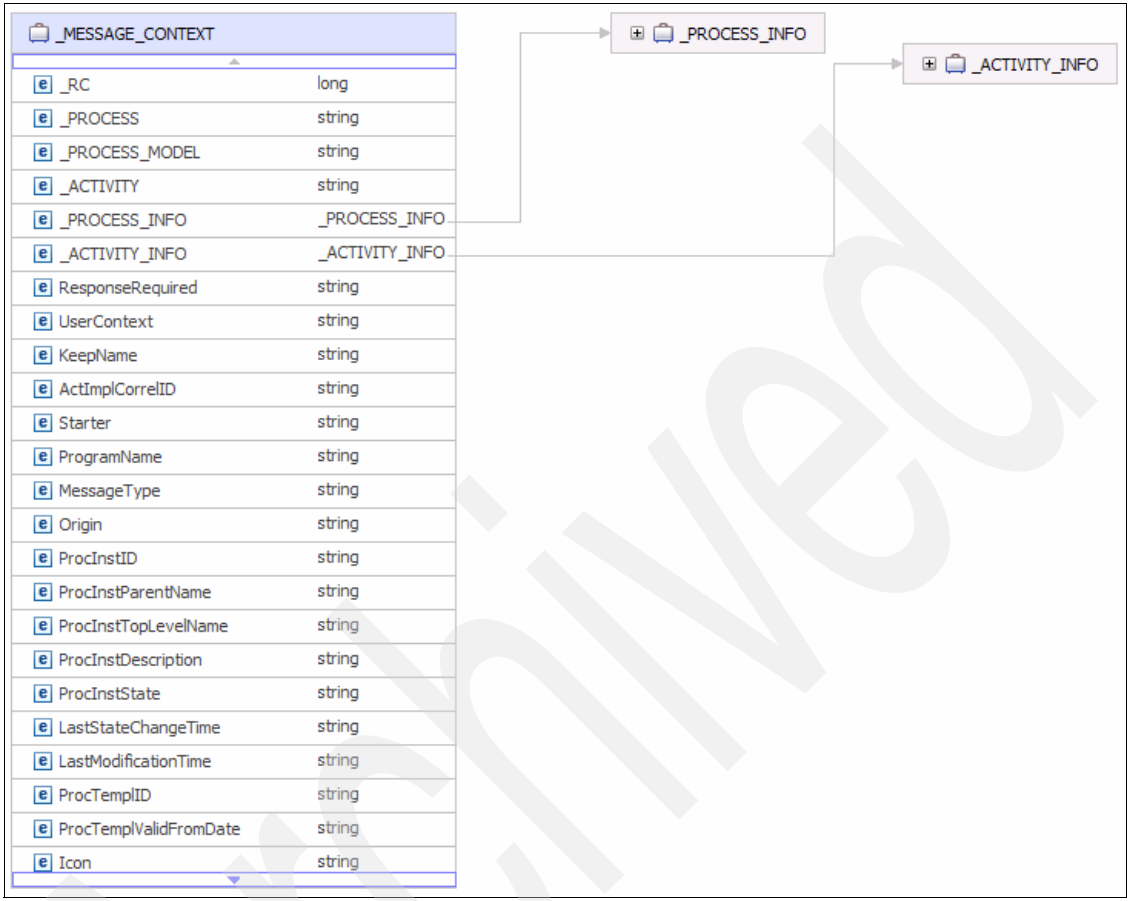


Figure 9-16 MESSAGE_CONTEXT data type

Interfaces

Interfaces were automatically generated for activities in the process, as well as for the process itself. If your import includes the predefined data members, these interfaces use the larger variables created with the _MT suffix. Since this option was not selected, the interfaces created have smaller, more efficient input and output messages.

Figure 9-17 shows the interfaces.

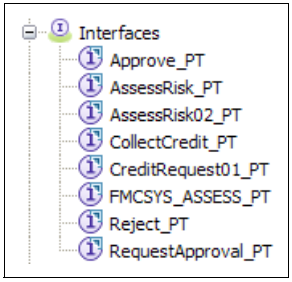


Figure 9-17 Interfaces

Two interfaces were created for Assess Risk. AssessRisk_PT is not used by any of the other artifacts, while AssessRisk02_PT is used by the generated processes. The interfaces are all simple, with just one operation along with an input and an output message. The FMCSYS_ASSESS_PT interface used for the user-defined program execution server step includes a fault message.

Business processes

The FDL2BPEL Conversion tool also generated WS-BPEL business processes based on the original Flow Definition Language. CreditRequest in the MQWorkflow Definition Language process was mapped to a WS-BPEL process named CreditRequest01.

The generated process included an interface partner using the CreditRequest01_PL interface to define how the process can be invoked. A reference partner was defined using the AssessRisk02_PL interface, for the step where the second process is invoked.

WS-BPEL variables were also generated using the various data types. These variables can be broken down into several categories:

- GLOBAL_CONTAINER is a variable to emulate the global data container in WebSphere MQ Workflow. The output from various activities could be mapped to this variable, which would enable programs and clients that relied on the global data container to be able to work with the new process.
- For each activity in the process a variable is created for the input and the output, such as RequestApproval01_IN and RequestApproval01_OUT. Activities in WebSphere MQ Workflow have input and output data containers, whereas in WS-BPEL all variables are scoped at the process level. By creating input and output variables for each activity, the behaviors from WebSphere MQ Workflow can be emulated more closely. The trade-off is that the resulting process may not be as readable or as efficient as possible. Figure 9-18 shows the WS-BPEL variables.

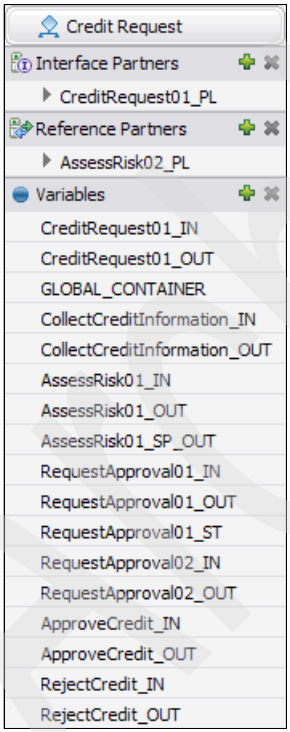


Figure 9-18 WS-BPEL variables

- In WebSphere MQ Workflow, processes have a source and a sink for data flowing in and out. A pair of WS-BPEL variables was generated to emulate the source and sink:
  - CreditRequest01_IN
  - CreditRequest01_OUT
- RequestApproval is a block in the original process. A variable RequestApproval01_ST was generated, which gets used so that the WS-BPEL can mimic the behavior of FDL. An FDL block is like a do-while loop, always performing at least one iteration. WS-BPEL only supports while loops, which only iterate if the loop condition is true. This extra variable is used to ensure that the loop always runs the first time, to match the behavior of an FDL block.
- In the original process, RequestApproval had both a default data connector and a data loop connector, moving data between input and output containers. Since WS-BPEL does not have data containers, a pair of extra variables was generated for this step in order to emulate the MQWorkflow Definition Language constructs:
  - RequestApproval02_IN
  - RequestApproval02_OUT
- Since the AssessRisk step is a subprocess, a variable AssessRisk01_SP_OUT was generated. Data from this variable is then mapped into the normal output variable. This is done so that any default values set can be merged with the output data from the subprocess.

The WS-BPEL process generated closely matches the structure of the original MQWorkflow Definition Language process, as shown in Figure 9-19.

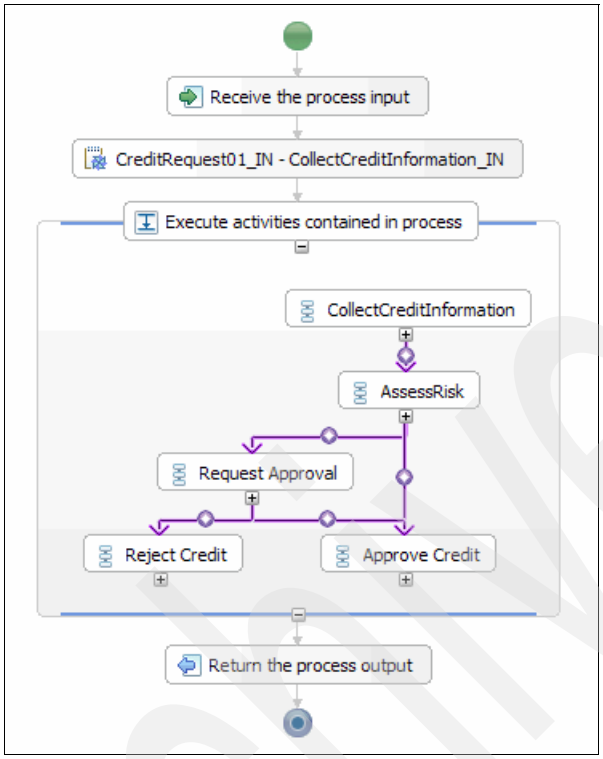


Figure 9-19 WS-BPEL Credit Request process with sequences collapsed

The original process had a data connector from the process input container to the input container of the CollectCreditInformation activity. Since WS-BPEL does not have the concept of a data connector, the process begins with a Java snippet to access the CreditRequest01_IN variable, which the receive activity writes into, then set up the CollectCreditInformation_IN variable, which the following activity reads from.

This snippet is shown in Example 9-1.

Example 9-1 Java snippet

```
CollectCreditInformation_IN =
com.ibm.bpe.interop.WMQWFHelper.merge(CreditRequest01_IN, "/_STRUCT",
CollectCreditInformation_IN, "/_STRUCT",
getVariableType("CollectCreditInformation_IN"));
if (CollectCreditInformation_IN == null) {
```

```
    com.ibm.websphere.bo.BOFactory boFactory = (com.ibm.websphere.bo.BOFactory)
com.ibm.websphere.sca.ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOF
actory");
    CollectCreditInformation_IN =
boFactory.createByType(getVariableType("CollectCreditInformation_IN"));
}
```

Each step in the original process is represented as a WS-BPEL sequence, which is a structured activity, meaning that it contains a collection of other activities. As a default, they are all collapsed when the process is first edited, as shown in Figure 9-20.

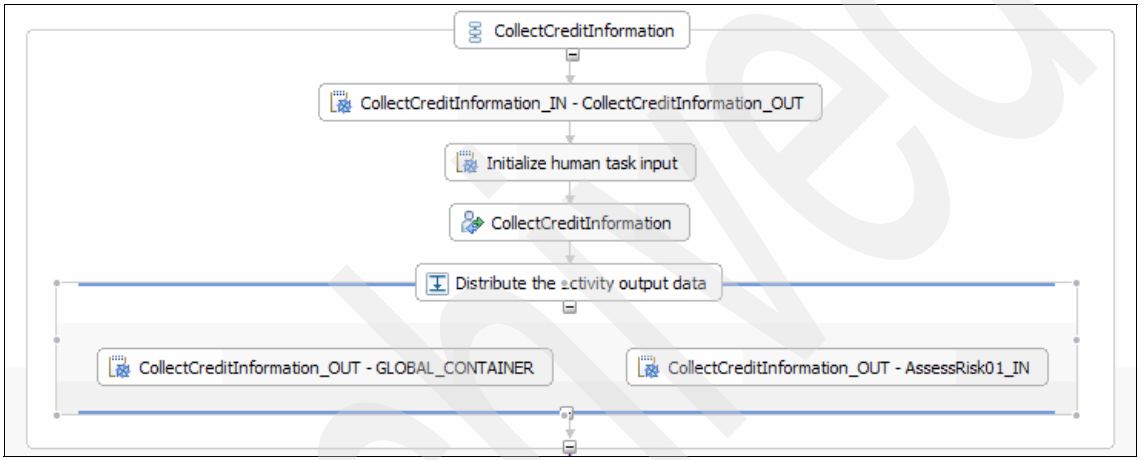


Figure 9-20 CollectCreditInformation sequence expanded

When expanded, the CollectCreditInformation sequence shows that it is made up of a number of activities.

- *CollectCreditInformation_IN - CollectCreditInformation_OUT* is a Java snippet used to initialize the output variable with the input variable. This mimics the functionality of the default data connector in WebSphere MQ Workflow. The data mappings in the original data connector are preserved. For each data member mapped, an instruction in the Java snippet moves the data from the input to the output.
- *Initialize human task input* is a Java snippet that checks whether the input variable for the next step is null. If so, it builds a business object.
- *CollectCreditInformation* is the human task itself. The staffing information from the original process is preserved.

- *Distribute the activity output data* is an activity of the WS-BPEL type parallel activities. It contains two more Java snippets to be run in parallel:
  - *CollectCreditInformation_OUT - GLOBAL_CONTAINER* mimics the data connector from CollectCreditInformation to the global data container. The mappings from this connector are used to generate Java, which updates the WS-BPEL variable GLOBAL_CONTAINER.
  - *CollectCreditInformation_OUT - AssessRisk01_IN* writes from the output variable of the CollectCreditInformation step to the input for the next task. This mimics the data flow connector between tasks in the original process.

If consecutive tasks in a process use the same data types in a WS-BPEL process, it would be more efficient to use the same variable rather than performing this mapping at each step.

The RequestApproval sequence is slightly more complex, since it represents the WS-BPEL version of a Flow Definition Language block activity.

Figure 9-21 shows the RequestApproval sequence with all nodes expanded.

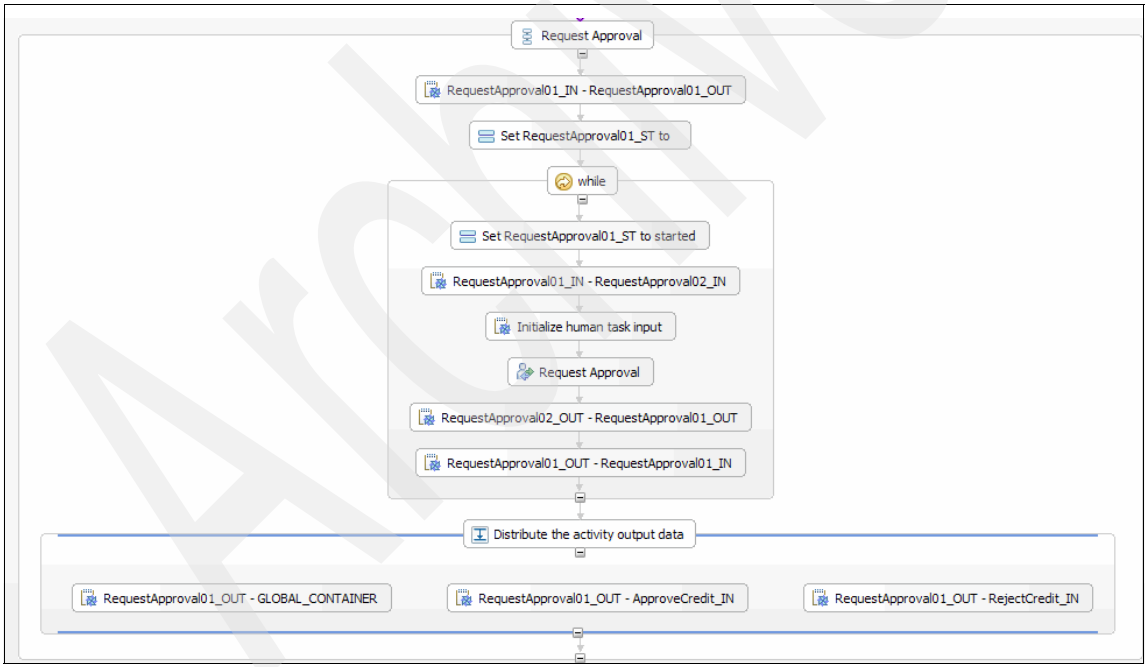


Figure 9-21 RequestApproval sequence with all nodes expanded

This sequence of WS-BPEL activities mimics the behavior of a Flow Definition Language block:

- *RequestApproval01_IN - RequestApproval01_OUT* mimics the behavior of the default data connector by initializing the output variable.
- *Reset RequestApproval01_ST* is an assign activity that initializes the variable RequestApproval01_ST to null.
- A while loop starts next, containing several activities:
  - *SetRequestApproval01_ST to started* is an assign activity that sets the string variable that was just initialized to a value of *started*. This is used to emulate the behavior of a block in an FDL process, ensuring that at least one iteration is performed.
  - *RequestApproval01_IN: RequestApproval02_IN* is a Java snippet used to perform data mapping. Inside the block in the original process, a data connector maps the source for the block to the input container of *request approval*. This Java snippet emulates the data connector.
  - *Initialize human task input* is a Java snippet that initializes the input variable if it is *null*.
  - *Request Approval* is the human task itself. Staffing information from the original MQWorkflow Definition Language model is preserved.
  - *RequestApproval02_OUT: RequestApproval01_OUT* is a Java snippet that sets a variable used to emulate the data mapping from *request approval* to the sink for the block.
  - *RequestApproval01_OUT: RequestApproval01_IN* is a Java snippet that takes the output and sets the input variable to emulate the data loop connector in MQWorkflow Definition Language.
- The while loop is followed by a WS-BPEL parallel activities container named *distribute the activity output data*, with three activities:
  - *RequestApproval01_OUT: GLOBAL_CONTAINER* is a Java snippet that maps the subset of the output data to the variable that emulates the global container in WebSphere MQ Workflow.
  - *RequestApproval01_OUT: ApproveCredit_IN* is a Java snippet that emulates the data connector to the *approve credit* step.
  - *RequestApproval01_OUT: RejectCredit_IN* is a Java snippet that emulates the data connector to the *reject credit* step.
- The sequence activity is now completed.

The sequence itself looks rather complicated, with so many extra steps besides the task itself. However, each of the Java snippets and map activities is included to emulate constructs in MQWorkflow Definition Language that do not have a

direct correlation to WS-BPEL. The means that the new process very closely emulates the behavior of the original process. However, each of these steps adds overhead. For a large process, this could lead to inefficiency. For these reasons, while the generated process could be run, in most cases some amount of optimization could be performed. Using any tool, a generated process is never as efficient as one written by hand.

For each activity in the generated business process, the checkbox for Continue On Error is unchecked. As discussed in 9.1.5, “Optimizing the generated WS-BPEL” on page 201, leaving the box unchecked more closely emulates the behavior of WebSphere MQ Workflow.

Figure 9-22 shows the server settings for the request approval task.

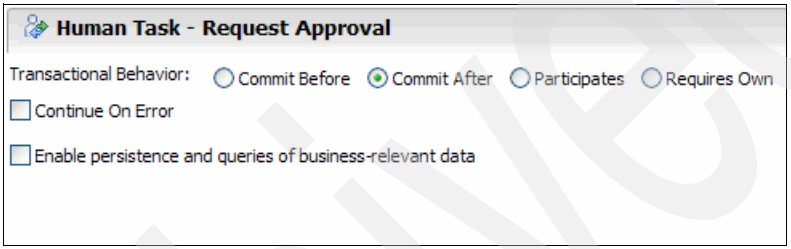


Figure 9-22 Server settings for request approval task

A second WS-BPEL business process was generated: AssessRisk02. This process consisted of a single step in the original MQWorkflow Definition Language, along with data connectors to the source and sink, as well as a default data connector. The generated process is somewhat more complex looking.

Figure 9-23 shows the AssessRisk02 process in WS-BPEL.

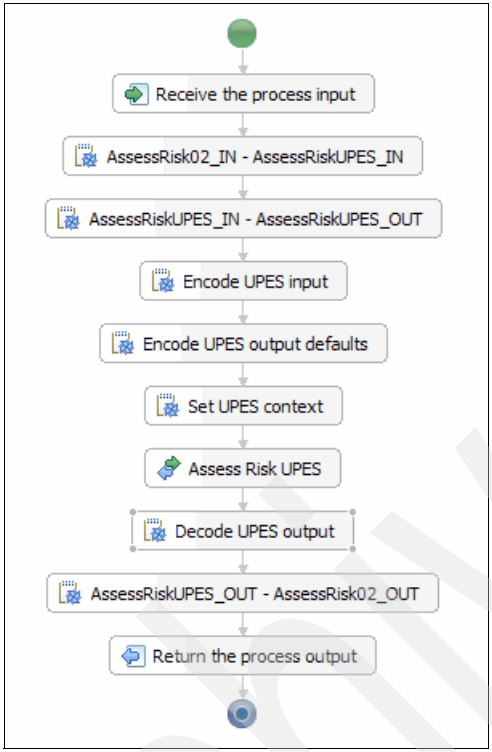


Figure 9-23 AssessRisk02 process in WS-BPEL

As with the CreditRequest01 process, the AssessRisk02 process has several variables and Java snippets that emulate the behavior of the constructs in WebSphere MQ Workflow.

- AssessRisk02_IN: AssessRiskUPES_IN is a Java snippet that initializes the AssessRiskUPES_IN variable, emulating the incoming data connector.
- AssessRiskUPES_IN: AssessRiskUPES_OUT is a Java snippet to initialize the output variable, as the default data connector did in MQWorkflow Definition Language.
- Encode UPES input is a Java snippet that copies values into the AssessRiskUPES_UPES_IN variable (data type CreditInfo_UPES_IN_MT, which includes the message context and the message container) from AssessRiskUPES_IN (data type CreditInfo_MT).
- Encode UPES defaults is a Java snippet that sets up the message default container values in AssessRiskUPES_UPES_IN.

- Set UPES context is a Java snippet that populates the message context data in the AssessRiskUPES_UPES_IN variable.
- Assess Risk UPES is a WS-BPEL invoke activity to invoke the back-end user-defined program execution server.
- Decode UPES output is a Java snippet that moves data from the AssessRiskUPES_UPES_OUT variable to AssessRiskUPES_OUT.
- AssessRiskUPES_OUT: AssessRisk02_OUT emulates the data connector to the sink by mapping the output data from the Assess Risk UPES step to the AssessRisk02_OUT variable.

The version of the process that was rewritten used one variable for this process. The generated version used six. The rewritten version had one step (other than the receive and reply activities), while the generated version had eight. In an attempt to closely match the behavior of WebSphere MQ Workflow, FDL2BPEL created a process that was not as efficient as possible. If you intend to keep your current UPES implementation, you must have a variable with a data type that matches the current implementation. If you are moving to a new implementation of the UPES, these extra structures may not be needed.

Service Component Architecture assembly diagram

A Service Component Architecture assembly diagram was generated for the components in the module, as shown in Figure 9-24.

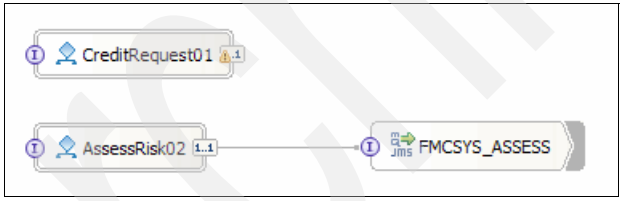


Figure 9-24 Service Component Architecture assembly diagram

The CreditRequest01 process is represented by a process component with an interface and one reference so that it can invoke the subprocess AssessRisk02. However, these components were not automatically connected by FDL2BPEL. The reference partner AssessRisk02_PL specifies a process template, which overrides any SCA wiring. This ensures that late binding is used, meaning that the current version of AssessRisk02 is invoked, rather than early binding, where the version used is always the one it was deployed with. This mimics the behavior of WebSphere MQ Workflow, which uses late binding.

The component representing the AssessRisk02 process also has an interface so that it can be invoked from the CreditRequest01 process. In addition, it has a reference that is linked to an export with MQ JMS binding. The name of the

WebSphere MQ queue manager from the WebSphere MQ Workflow user-defined program execution server is preserved. Since the JMS API is being used for this component, a JNDI name is needed for the end-point rather than the queue name used in the user-defined program execution server definition. The JNDI name used places the message into the proper queue, preserving the name from the original FDL.

When messages are written to a queue, the business object in WebSphere Process Server must be serialized. A data binding is used for this purpose. WebSphere Process Server includes a helper class WMQWFUpesDataBinding. Along with the MQ JMS export, com.ibm.workflow.sca.jms.data.AssessRiskUPES_Response_UPES_MSG_DataBindingImpl.java is generated, which extends WMQWFUpesDataBinding by setting the data (CreditInfo_UPES_OUT_MT in this case), as well as setting the name space. This means that you do not need to write any code to serialize and deserialize the user-defined program execution server messages. The FDL2BPEL tool generates it automatically.

9.3.3 Optimizing the FDL2BPEL output

As with any automated tool, the output of FDL2BPEL is not as efficient as the rewritten version. Before using the generated process, optimization should be performed. The amount of work done to the process varies depending on the requirements. The following are possible optimizations that can be considered, but may not apply to all cases:

- The generated process has an overly large number of WS-BPEL variables, but it is more efficient to reuse variables. Processes should use multiple variables when there are parallel paths of execution. When the process follows a single path, you can safely reuse a variable.
- Reducing the number of variables would also mean that the number of Java snippets used for mapping could be reduced or eliminated. You could eliminate the Java snippets and reuse a single variable in several situations:
  - Two consecutive activities in the FDL process are connected with "_STRUCT to _STRUCT" mapping in the data connector, with no joining or branching, and no references to predefined data members.
  - Two activities flow to one single activity, where "_STRUCT to _STRUCT" data mapping is performed, with no other data connectors used, and no default values set in the output of the two activities, and no references to the predefined data members.
  - One activity branches out to other activities, where "_STRUCT to _STRUCT" mapping is used, with no other data connectors and no default

values set for the inputs of the activities, and no references to the predefined data members.

  - Inbound data connectors from a data source.
  - Outbound data connectors to a data sink.
  - A loop data connector, with "_STRUCT to _STRUCT" data mapping, no default values used, and no references to predefined data members.
  - A default data connector with "_STRUCT to _STRUCT" data mapping, no default values used, and no references to predefined data members.

- If it is determined that the functionality from the data mapping in the Java snippets is required, they could be combined. In the AssessRisk02 process, for example, rather than having five snippets before the invoke, they could be combined into one single snippet. This would make the process look cleaner, and would make it more efficient since the process would only have to navigate through one activity instead of five. The snippets are all set with their transactional behavior set to *Participates*, so they do not require a database commit between activities, but the process engine must still navigate through each of them.
- The names of the generated artifacts could have been clearer if name conflicts were removed in the FDL before exporting.
- If you attempt to optimize your process by eliminating extra tasks and variables, be aware that you must carefully test the updated process to ensure that it still delivers the same results as the WebSphere MQ Workflow runtime.

MQWorkflow Definition Language constructs that cannot be transitioned

Some constructs in WebSphere MQ Workflow described by MQWorkflow Definition Language cannot be transitioned to WS-BPEL and WebSphere Process Server because of differences in the architecture. For example, WebSphere Process Server does not have a program execution agent (PEA) as WebSphere MQ Workflow did. If a process uses such a construct, you must determine how to handle the functionality in the new system. See Chapter 11, "Integrating back-end applications" on page 251, for a discussion.

Recommendations

After running FDL2BPEL, each of the generated WS-BPEL sequences should be examined in detail, and a determination made:

- Does the generated WS-BPEL match the business intent of the original MQWorkflow Definition Language process?

  The flow being migrated is a business process. After transitioning to the new system, the business intent of the original process must be preserved. A determination must be made to ensure that the new system accomplishes the business goals of the original process, even if the method of doing so has to change.

- Are all of the generated Java snippets required?

  As discussed above, FDL2BPEL attempts to map all of the MQWorkflow Definition Language constructs to WS-BPEL, and in doing so generates numerous Java snippets. As the number of Java snippets adds up in a large process, inefficiency can be introduced, as well as complexity of the execution model. A determination must be made as to which of the mappings are necessary in the new system. For example, if a default data connector was used in every activity in the original process, an extra Java snippet is generated along with special variables in some cases. If this is not required in the new system, efficiency can be gained by eliminating the unnecessary mappings.

- Is the generated process as efficient as possible?

  Especially for high-volume flows, a determination must be made for the overall process efficiency.

- Is there a better way to do it?

  As discussed in 11.2.2, “UPES-invoked applications” on page 260, there are new functions in WebSphere Process Server that did not exist in WebSphere MQ Workflow. As the model is being converted, it is important to determine whether a new function could be used, rather than attempting to exactly emulate the original process. For example, WS-BPEL supports constructs for dealing with errors such as fault handlers and compensation. If the original process has branches for when errors occur, with *undo* or other cleanup steps, it would be better to use a fault handler and a compensation handler, rather than to have a process model that looks more complex because of the cleanup logic.

- How should constructs be handled that cannot be transitioned?

  When constructs were used in the original process that cannot be directly migrated, a determination must be made as to how the new process can deliver the business requirements using the available functionality. As part of

the process model migration, cleanup, and optimization, these limitations must be dealt with.

9.3.4 Limitations

In migrating the *credit request* process, we noted several limitations:

- FDL2BPEL created a larger number of variables than needed. With a large process, this could be inefficient, since the use of variables involves reads and writes to the database.
- Whether needed or not, each activity in the original process was mapped as a sequence of activities in WS-BPEL, introducing more complexity than necessary in some cases. All of the additional steps could make the business intent of the process more difficult to understand.
- The names of the generated artifacts were not always optimal. You may want to refactor some of the names to improve readability.
- Using FDL2BPEL, only the technical execution model is created. There is no corresponding business model.
- FDL2BPEL did not connect all the components in the Service Component Architecture assembly diagram. While the specification of a process template in the reference partner overrides SCA wiring, having the wire makes it clear which processes are invoking each other.
- FDL2BPEL created one single module with all artifacts. The best practice for reusable components is to place items such as data types and interfaces into a library project so that they can be more easily reused. It is also a best practice to separate the business logic from the implementation details by placing components in a separate module from the business process. This allows you to modify the component without having to redeploy the process. In order to apply these best practices, you would have to manually move the artifacts, refactor, then rebuild the project.
- Human tasks were always created as inline human task activities. This enables the tasks to access the process context and perform staffing scenarios such as using the f*our eyes principle*, assigning work to a role but excluding a user who performed a previous step. This allows more mappings to WebSphere MQ Workflow staff assignments. However, some companies prefer to use external tasks because the WS-BPEL process can use a standard invoke activity rather than an IBM extension. Also, if the task was later automated, it is easier to make an update if an external task is used. FDL2BPEL does not offer a choice for which type to use.

- Artifacts such as processes and interfaces are built with names generated by FDL2BPEL. These names may not meet naming standards for a given organization and could require updating.
- FDL2BPEL cannot migrate all of the MQWorkflow Definition Language constructs.

For additional detail on limitations, see SupportPac WA73: FDL2BPEL Conversion Tool available at:

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en>>

9.4 Combining rewriting and using FDL2BPEL Conversion

When converting the existing process model to WS-BPEL, there are distinct advantages and disadvantages to each of the three approaches above. An additional option is to use a hybrid approach, combining the use of the FDL2BPEL Conversion tool and rewriting. This approach can be used to leverage the advantages offered by each method.

- FDL2BPEL can be used to generate a module with many useful artifacts that can be salvaged from the MQWorkflow Definition Language process model.
- In keeping with best practices, create a library project to hold reusable data types and interfaces. Copy and paste data types from the FDL2BPEL module into the library project. Only the required data types must be moved. However, when a data type is copied to the clipboard, all data types it references (or that reference it) are also copied. Similarly, cutting and pasting an interface brings along all referenced interfaces and data types.
- Copy the import module to a new module.
- Delete any constructs that are not needed.
- Create your process by reusing the artifacts generated by FDL2BPEL.

As mentioned in the limitations above, not every construct in MQWorkflow Definition Language has a direct correlation to WS-BPEL. For this reason, process migrations may require some degree of manual rework. Even if just for this reason alone, the technique of using FDL2BPEL along with rewriting is required in some cases.

Part 2

Transition techniques

This part discusses the technical aspects of a transition. It enables you to decide how to convert business processes and integrate existing back-end applications in the new environment, and provides the information required to transition existing clients and to set up an operational environment that meets you current and future needs.

This part contains the following chapters:

- Chapter 10, “Implementing human interaction in business processes” on page 229, discusses how to map staff repository content from WebSphere MQ Workflow to an LDAP directory used by WebSphere Process Server. A sample is provided that contains a configuration definition to use the WebSphere MQ Workflow LDAP Bridge, a Java program using APIs to map staff information, as well as a Java program to map substitution information. In addition, customization options for Human Task Manager staff resolution are discussed.
- Chapter 11, “Integrating back-end applications” on page 251, describes the different integration variants for existing user-defined program execution

server (UPES) environments and the transition options available in WebSphere Process Server.

- Chapter 12, “Implementing clients based on application programming interfaces” on page 275, provides an overview of the interfaces of WebSphere Process Server and WebSphere MQ Workflow and how to map them. It describes major elements of application programming interface-based application and gives mapping hints. The programming models of both products are shown in detail for a better understanding of an application transition.
- Chapter 13, “Implementing operational aspects” on page 357, provides the information required to assess the WebSphere MQ Workflow topology in place and define the desired target topology in a WebSphere Process Server environment. The aspects discussed are the server topology, high availability, workload management and scalability, as well as best practices and recommendations for planning the target topology.

10

Implementing human interaction in business processes

This chapter discusses how to map staff repository content from WebSphere MQ Workflow to an LDAP directory used by WebSphere Process Server. The topics discussed are:

- Using the LDAP Bridge to map staff repository content from WebSphere MQ Workflow to WebSphere Process Server
- Using APIs to map staff repository content from WebSphere MQ Workflow to WebSphere Process Server
- Mapping substitution information from WebSphere MQ Workflow to WebSphere Process Server
- Customization options for Human Task Manager staff resolution

In order to show concrete mapping definitions we assume that Tivoli Directory Server is used as an LDAP server together with an out-of-the box WebSphere Virtual Member Manager (VMM) configuration. This means that the LDAP object classes inetOrgPerson and groupOfNames are used to represent people and groups.

The sample contains three parts:

- A configuration definition to use the WebSphere MQ Workflow LDAP Bridge
- A Java program using APIs to map staff information
- A Java program to map substitution information

In addition, we discuss customization options for Human Task Manager staff resolution.

10.1 Using the LDAP Bridge to map staff repository content from WebSphere MQ Workflow to WebSphere Process Server

In order to keep this example simple, our task is to map organizations and people from the WebSphere MQ Workflow staff repository to an assumed LDAP directory used by WebSphere Process Server and mapping only some attributes. It should be easy for you to enhance the examples in such a way that all attributes relevant for you are also mapped.

The WebSphere MQ Workflow LDAP Bridge can be easily set up to map staff data from WebSphere MQ Workflow to LDAP. On the other hand, especially when mapping references between records, it has some limitations. In the example here, these limitations prevent us from using the LDAP Bridge to map:

- The manager relation

  In WebSphere MQ Workflow, the manager is an attribute of an organization, whereas in the assumed LDAP schema, it is an attribute of a person.

- The organization hierarchy relation

  In the given LDAP schema, the attribute member of a group contains information about both the people who are members of the group and the sub-groups of this group.

These relations are mapped in the next chapter with the help of appropriate APIs.

The LDAP Bridge configuration is done with two files:

- The configuration properties file
- The mapping definition on file

At first, we discuss the beginning of the LDAP Bridge configuration file. To better explain the content, each line is preceded by a line number, as shown in Example 10-1.

Example 10-1 Beginning of the LDAP Bridge configuration file

```
1  MasterDevice =          FDL
2  ReplicaDevice =
3  OutputDevices =         LDAP
4
5  FDL.Type =              FDLFile
6  FDL.InputFileBase =     Input
7
8  LDAP.Type =             LDAPClient
```

```
 9  LDAP.MappingFile =      LDAPSampleMapping.xml
10  LDAP.Server =           ServerName
11  LDAP.ServerPort =       389
12  LDAP.User =             cn=root
13  LDAP.Password =         ***
14  LDAP.OutputForwardRef = true
```

Remarks about the configuration file:

- Lines 1–3: These lines contain the definitions of the devices. As we simply want to map data from WebSphere MQ Workflow to LDAP, the replica device should be left empty. The master device is the data input.
- Lines 5–6: As the master device is FDL, all definitions with the prefix FDL are relevant for the definition of this device. For the sake of simplicity we assume here that the input is available as an FDL file. A direct access of the WebSphere MQ Workflow staff repository with the WebSphere MQ Workflow Staff Administration API would also be possible.
- Lines 8–14: For output device LDAP all relevant definitions can be found here. As the device type is LDAP Client, the updates are made directly in the LDAP directory. An alternative would be to write an LDIF file instead, which could be imported into the directory.

Example 10-2 discusses the mapping definition file. To better explain the content, each line is preceded by a line number.

Example 10-2 LDAP Bridge mapping definition file

```
 1  <?xml version="1.0" encoding="UTF-8" ?>
 2  <!DOCTYPE MappingSpecs SYSTEM "LDAPMapping.dtd">
 4  <MappingSpecs>
 5      <FDLObjectClass Name="PERSON">
 6          <LDAPObjectClass Name="inetOrgPerson" RdnName="cn"/>
 7          <LDAPObjectClass Name="top"/>
 8          <LDAPObjectClass Name="organizationalPerson"/>
 9          <LDAPObjectClass Name="person"/>
10          <LDAPQuery Suffix="ou=People,o=company"/>
11          <FDLAttribute Name="Name">
12              <DirectAttributeMapping>
13                  <LDAPAttribute Name="cn"/>
14              </DirectAttributeMapping>
15          </FDLAttribute>
16          <FDLAttribute Name="PHONE">
17              <DirectAttributeMapping>
18                  <LDAPAttribute Name="telephoneNumber"/>
```

```
            </DirectAttributeMapping>
        </FDLAttribute>
        <FDLAttribute Name="LAST_NAME">
            <DirectAttributeMapping>
                 <LDAPAttribute Name="sn"/>
            </DirectAttributeMapping>
        </FDLAttribute>
        <FDLAttribute Name="DESCRIPTION">
            <DirectAttributeMapping>
                <LDAPAttribute Name="description"/>
            </DirectAttributeMapping>
        </FDLAttribute>
    </FDLObjectClass>
    <FDLObjectClass Name="ORGANIZATION">
        <LDAPObjectClass Name="groupOfNames" RdnName="cn"/>
        <LDAPObjectClass Name="top"/>
        <LDAPQuery Suffix="ou=Organizations,o=company"/>
        <FDLAttribute Name="Name">
            <DirectAttributeMapping>
                <LDAPAttribute Name="cn"/>
            </DirectAttributeMapping>
        </FDLAttribute>
        <FDLAttribute Name="DESCRIPTION">
            <DirectAttributeMapping>
                <LDAPAttribute Name="description"/>
            </DirectAttributeMapping>
        </FDLAttribute>
        <FDLAttribute Name="RELATED_PERSON">
            <AttributeReferenceMapping>
                <LDAPAttribute Name="member"/>
            </AttributeReferenceMapping>
        </FDLAttribute>
    </FDLObjectClass>
</MappingSpecs>
```

Remarks about the mapping file:

- Lines 5–10: The mapping definition for PERSON records with general definitions for this record type.
- Lines 11–30: The mapping definitions for the following attributes of a PERSON record are defined here:
  - Name
  - Phone
  - Last name
  - Description
- Lines 32–35: The mapping definition for ORGANIZATION records with general definitions for this record type.
- Lines 36–50: The mapping definitions for the following attributes of an ORGANIZATION record are defined here:
  - Name
  - Description
  - Member of the organization

  Note that in the default LDAP schema, the LDAP attribute member is a mandatory attribute, which means that only organizations that have members can be mapped to the LDAP directory.

10.2 Using APIs to map staff repository content from WebSphere MQ Workflow to WebSphere Process Server

The following Java program shows how to map staff repository content from WebSphere MQ Workflow toWebSphere Process Server. It makes usage of the following APIs:

- The WebSphere MQ Workflow Staff Administration API is used to retrieve the information from the WebSphere MQ Workflow staff repository.
- The Java Naming and Directory Interface (JNDI) of the Java Toolkit is used to write the information into the LDAP directory. (Most corporate directories do *not* permit write operations. Instead, LDAP administration based on LDIF files is commonly used.)

Compared with the configuration of the LDAP Bridge, writing a Java program is a more complex task. But nevertheless, it could be easily used to not only map the missing data after the usage of the LDAP Bridge, as shown in Example 10-3, but it could also be used instead of the LDAP Bridge.

Example 10-3 API example

```
//*************************************************************************
//  Licensed Materials - Property of IBM
//  (C) Copyright IBM Corp. 2007
//  US Government Users Restricted Rights - Use, duplication or
//  disclosure restricted by GSA ADP Schedule Contract
//  with IBM Corp.
//
//  DISCLAIMER
//  ----------
//  This material contains programming source code for your consideration.
//  These examples have not been thoroughly tested under all conditions.
//  IBM, therefore, cannot guarantee or imply reliability, serviceability,
//  or function of these programs.
//  ALL PROGRAMS CONTAINED HEREIN ARE PROVIDED TO YOU "AS IS", WITHOUT ANY
//  WARRANTIES (EXPRESS OR IMPLIED) OR SUPPORT WHATSOEVER, INCLUDING BUT
//  NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
//  FOR A PARTICULAR PURPOSE.
//*************************************************************************

import java.util.Hashtable;

import javax.naming.Context;
import javax.naming.directory.Attribute;
import javax.naming.directory.Attributes;
import javax.naming.directory.BasicAttribute;
import javax.naming.directory.BasicAttributes;
import javax.naming.directory.DirContext;
import javax.naming.directory.InitialDirContext;

import com.ibm.workflow.api.FmcException;
import com.ibm.workflow.staff.StaffOrganization;
import com.ibm.workflow.staff.StaffPerson;
import com.ibm.workflow.staff.StaffSession;

public class LDAPUpdate {
    static DirContext ctx;
    static StaffSession session;
```

```
38
39      static String getOrgDN(StaffOrganization org) throws FmcException {
40          return "cn=" + org.name() + ",ou=Organizations,o=company";
41      }
42
43      static String getPersonDN(StaffPerson person) throws FmcException {
44          return "cn=" + person.name() + ",ou=People,o=company";
45      }
46
47      public static void main(String[] args) {
48
49          try {
50              // Login to LDAP
51              {
52                  String serverURL = "ldap://yourLDAPServer:389";
53                  String user = "cn=root";
54                  String password = "***";
55
56                  Hashtable env = new Hashtable(11);
57                  env.put(Context.INITIAL_CONTEXT_FACTORY,
58                          "com.sun.jndi.ldap.LdapCtxFactory");
59                  env.put(Context.PROVIDER_URL, serverURL);
60                  env.put(Context.SECURITY_PRINCIPAL, user);
61                  env.put(Context.SECURITY_CREDENTIALS, password);
62                  ctx = new InitialDirContext(env);
63              }
64
65              // Initialize the Staff Administration API session
66              {
67                  session = StaffSession.getInstance();
68                  session.logon("YourMQWFConfiguration", "ADMIN", "password",
true);
69              }
70
71              StaffOrganization[] org = session.queryAllOrganizations();
72              for (int i = 0; i < org.length; i++) {
73                  setMembersOfGroup(org[i]);
74                  StaffPerson member[] = org[i].relatedPersons();
75                  StaffPerson manager = org[i].manager();
76                  System.out.println("Organization: " + org[i].name()
77                          + " is managed by: " + manager.name());
78
79                  for (int j = 0; j < member.length; j++) {
80                          setManagerOfPerson(member[j], manager);
81                  }
```

```
            }
            session.commit();
        }

        catch (Throwable e) {
            e.printStackTrace();
        }

        try {
            session.logoff();
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }

    static void setMembersOfGroup(StaffOrganization group) {
        try {
            StaffPerson[] people = group.relatedPersons();
            StaffOrganization[] orgs = group.children();
            String groupDn = getOrgDN(group);
            System.out.println("Set members of group " + groupDn);

            Attributes attrs = new BasicAttributes(true);
            Attribute oneAttr = new BasicAttribute("member");
            for (int i = 0; i < people.length; i++) {
                oneAttr.add(getPersonDN(people[i]));
            }
            for (int i = 0; i < orgs.length; i++) {
                oneAttr.add(getOrgDN(orgs[i]));
            }
            attrs.put(oneAttr);
            ctx.modifyAttributes(groupDn, DirContext.REPLACE_ATTRIBUTE, attrs);
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }

    static void setManagerOfPerson(StaffPerson person, StaffPerson manager) {
        try {
            String personDn = getPersonDN(person);
            String managerDn = getPersonDN(manager);
            System.out.println("For person " + personDn
                    + " set manager attribute to " + managerDn);

            Attributes attrs = new BasicAttributes(true);
```

```
127             Attribute oneAttr = new BasicAttribute("manager");
128             oneAttr.add(managerDn);
129             attrs.put(oneAttr);
130             ctx.modifyAttributes(personDn, DirContext.REPLACE_ATTRIBUTE, attrs);
131         } catch (Throwable e) {
132             e.printStackTrace();
133         }
134     }
135 }
```

Remarks about the Java program:

- Lines 71–82: After establishing the connection to both WebSphere MQ Workflow and LDAP, these lines are the heart of the program. A loop iterates over the organizations found in the WebSphere MQ Workflow staff repository.
- Lines 79–81: For each member of a WebSphere MQ Workflow organization, the manager is set appropriately in LDAP.
- Lines 97–117: After retrieving the necessary information from the WebSphere MQ Workflow staff repository in lines 99 and 100, the necessary attributes are set to update the LDAP directory.
- Lines 119–134: Quite similar to the previous method, here the manager attribute of a person is set in the LDAP directory.

10.3 Mapping substitution information from WebSphere MQ Workflow to WebSphere Process Server

Substitution information can be used during staff resolution. Together with the absence indicator, this information can be used to assign work only to people who are available.

If you want to use the substitution feature of WebSphere Process Server, you must use WebSphere Virtual Member Manager (VMM) to access the staff repository. The substitution information is stored in a look-aside table. This means that in case you use LDAP as your sole people repository, you cannot access the substitution information with an LDAP interface. However, you can still configure VMM to use your LDAP repository.

Instead, the sample here shows how to set the substitution information with an API provided by WebSphere Process Server. Our sample uses the following two APIs:

- As in the previous example, the WebSphere MQ Workflow Staff Administration API is used to retrieve the information from the WebSphere MQ Workflow staff repository. In order to use this API, on your machine an installation of at least the WebSphere MQ Workflow Runtime database tools must exist.
- The WebSphere Process Server Human Task Manager EJB API is used to write the substitution information into the staff repository of WebSphere Process Server. To keep this sample simple, we access WebSphere Process Server in the test environment of WebSphere Integration Developer.

This sample shows how to build an application in WebSphere Integration Developer V6.1 to accomplish this task. Knowledge of how to write an EJB client application is helpful especially when adapting this sample to a WebSphere Process Server production system.

For more information about this see IBM Redbook Publication: *Experience J2EE! Using WebSphere Application Server V6.1*, SG24-7297.

Example 10-4 contains the following Java program.

Example 10-4 WebSphere Process Server substitution Java program example

```
//*****************************************************************************
//  Licensed Materials - Property of IBM
//  (C) Copyright IBM Corp. 2007
//  US Government Users Restricted Rights - Use, duplication or
//  disclosure restricted by GSA ADP Schedule Contract
//  with IBM Corp.
//
//  DISCLAIMER
//  ----------
//  This material contains programming source code for your consideration.
//  These examples have not been thoroughly tested under all conditions.
//  IBM, therefore, cannot guarantee or imply reliability, serviceability,
//  or function of these programs.
//  ALL PROGRAMS CONTAINED HEREIN ARE PROVIDED TO YOU "AS IS", WITHOUT ANY
//  WARRANTIES (EXPRESS OR IMPLIED) OR SUPPORT WHATSOEVER, INCLUDING BUT
//  NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
//  FOR A PARTICULAR PURPOSE.
//*****************************************************************************

import java.rmi.RemoteException;
import java.util.LinkedList;
```

```
import javax.ejb.CreateException;
import javax.naming.InitialContext;
import javax.naming.NamingException;

import com.ibm.task.api.HumanTaskManager;
import com.ibm.task.api.HumanTaskManagerHome;
import com.ibm.workflow.api.FmcException;
import com.ibm.workflow.staff.StaffPerson;
import com.ibm.workflow.staff.StaffSession;

public class MigSubst {
   private StaffSession mqwfSession;

   private HumanTaskManager aHTM;

   private MigSubst() throws FmcException, NamingException, CreateException,
         RemoteException {

      // Access the MQWF Staff Administration API
      {
         mqwfSession = StaffSession.getInstance();
         mqwfSession.logon("YourMQWFConfiguration", "ADMIN", "password",
               true);
      }

      // Access the WPS Human Task Manager EJB API
      {
         // Obtain the default initial JNDI context
         InitialContext initialContext = new InitialContext();

         // Lookup the remote home interface of the BusinessFlowManager
         // bean
         Object result = initialContext
               .lookup("java:comp/env/ejb/HumanTaskManagerHome");

         // Convert the lookup result to the proper type
         HumanTaskManagerHome aHTMHome = (HumanTaskManagerHome) javax.rmi.PortableRemoteObject
               .narrow(result, HumanTaskManagerHome.class);
         if (aHTMHome != null)
            aHTM = aHTMHome.create();
      }
   }

   public static void main(String[] args) {
      System.out.println("Start migration of substitutes");
      try {
         new MigSubst().migrateSubstitute();
      } catch (Throwable e) {
         e.printStackTrace();
```

```
        }
        System.out.println("End migration of substitutes");
    }

    private void migrateSubstitute() {
        try {
            // Read the the substitution information in MQWF
            StaffPerson[] person = mqwfSession.queryAllPersons();

            for (int i = 0; i < person.length; i++) {
                String substitute = (person[i].substitute() == null) ? null
                        : person[i].substitute().name();
                setSubstitutesInHTM(person[i].name(), substitute);
            }
            mqwfSession.commit();
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }

    private void setSubstitutesInHTM(String user, String substitute) {
        try {
            LinkedList<String> substituteList = new LinkedList<String>();
            if (substitute == null) {
                System.out.println("No substitute set for " + user + ".");
            } else {
                substituteList.add(substitute);
                System.out.println("Substitute of " + user + " set to "
                        + substitute + ".");
            }
            aHTM.setSubstitutes(user, substituteList);
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }
}
```

Together with the comments, the program should be easily understood. Note that error handling is only rudimentary. The following section explains how you can set up this sample in WebSphere Integration Developer.

10.3.1 Create the application client

In this section, we create the project containing the Java code of this sample:

1. In the drop-down menu of WebSphere Integration Developer select **File** → **New** → **Project...** → **Application Client Project** and click **Next**.
2. In the Application Client module pop-up window, complete the following fields:
   - Project Name: SubstClient
   - Target Runtime: WebSphere Process Server V6.1
   - Configurations: <custom>
   - Select Add project to an EAR
   - EAR Project Name: SubstClientEAR

   Click **Next**.
3. On the Project Facets pop-up, click **Next**.
4. On the Application Client module pop-up, deselect **Create a default Main class** and click **Finish**.

10.3.2 Create the EJB reference

In this section, we make the definitions necessary to access the Human Task Manager EJB API:

1. In the Project Explorer, double-click the project **SubstClient** and then open the **Deployment Descriptor**.
2. On the Overview tab of the Deployment Descriptor editor, in the Main class section, click the edit button to open the JAR Dependency Editor and insert MigSubst as the main class. Ignore the message `The class does not exist or is not visible in the project using the saved settings`. This is solved later. Save and close the JAR Dependency Editor.
3. In the Deployment Descriptor editor, switch to the **References** tab.
4. Add the Human Task Manager EJB API by clicking **Add** to open the Reference pop-up.
5. On the Reference pop-up, select **EJB reference** and click **Next**.
6. On the EJB Reference pop-up, perform the following actions:
   a. Set the name to **ejb/HumanTaskManagerHome**.
   b. Select the radio button **Enterprise Beans not in the workspace**.
   c. Set the type to **Session**.
   d. For the Home field, click **Browse** and select **HumanTaskManagerHome – com.ibm.task.api**.

e. For the Remote field, click **Browse** and select **HumanTaskManager – com.ibm.task.api**.

f. Click **Next** and then **Finish**.

7. After the wizard closed, in the WebSphere Bindings section of the References tab, enter the JNDI name `com/ibm/task/api/HumanTaskManagerHome`.

   The Deployment Descriptor now looks like Figure 10-1.

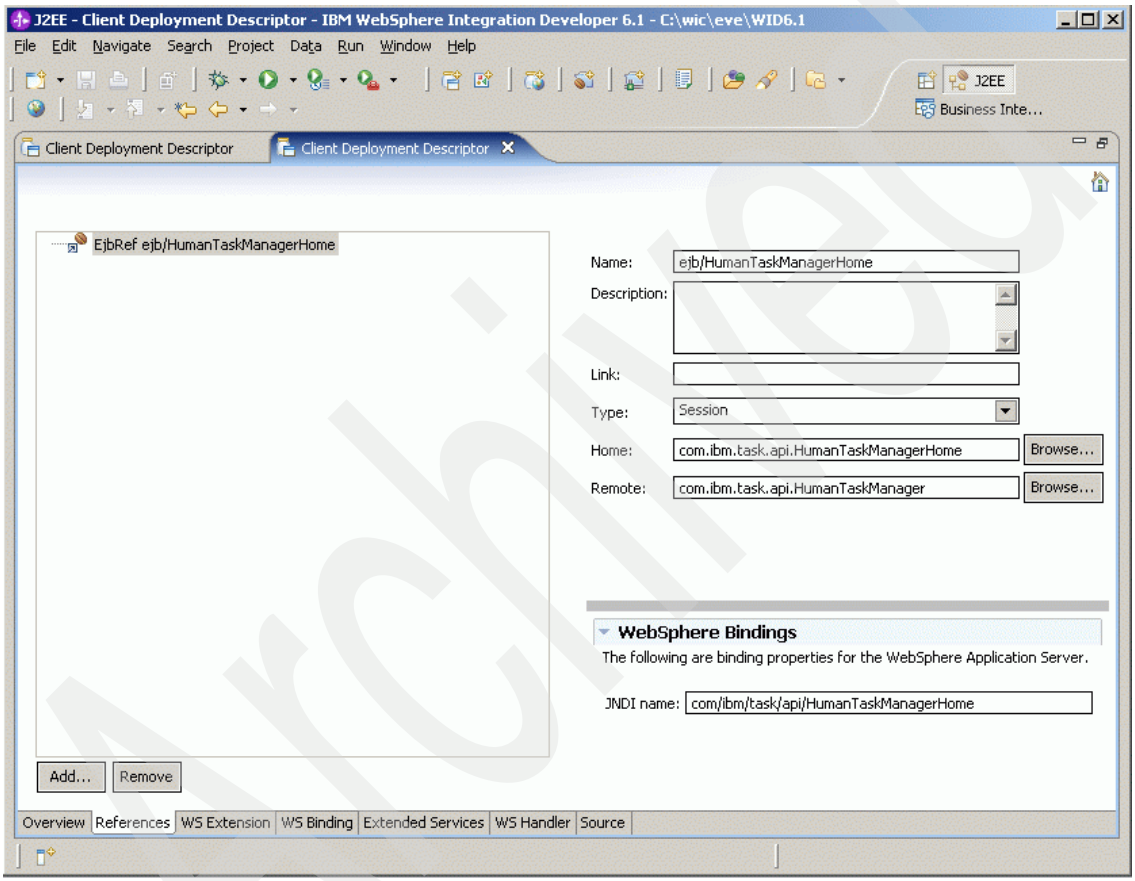


Figure 10-1 Deployment descriptor

8. Save the Deployment Descriptor.

10.3.3 Insert library references

The libraries necessary for the project must be defined. To do this:

1. In the Project Explorer window, right-click the project **SubstClient**, open the context menu, and select **Properties**.
2. In the Properties pop-up, select **Java Build Path** and the **Libraries** tab.
3. Click **Add External JARs** and in the file dialog box insert the location and name of the WebSphere MQ Workflow Staff Administration API fmcistf.jar. For reference, see Figure 10-2.

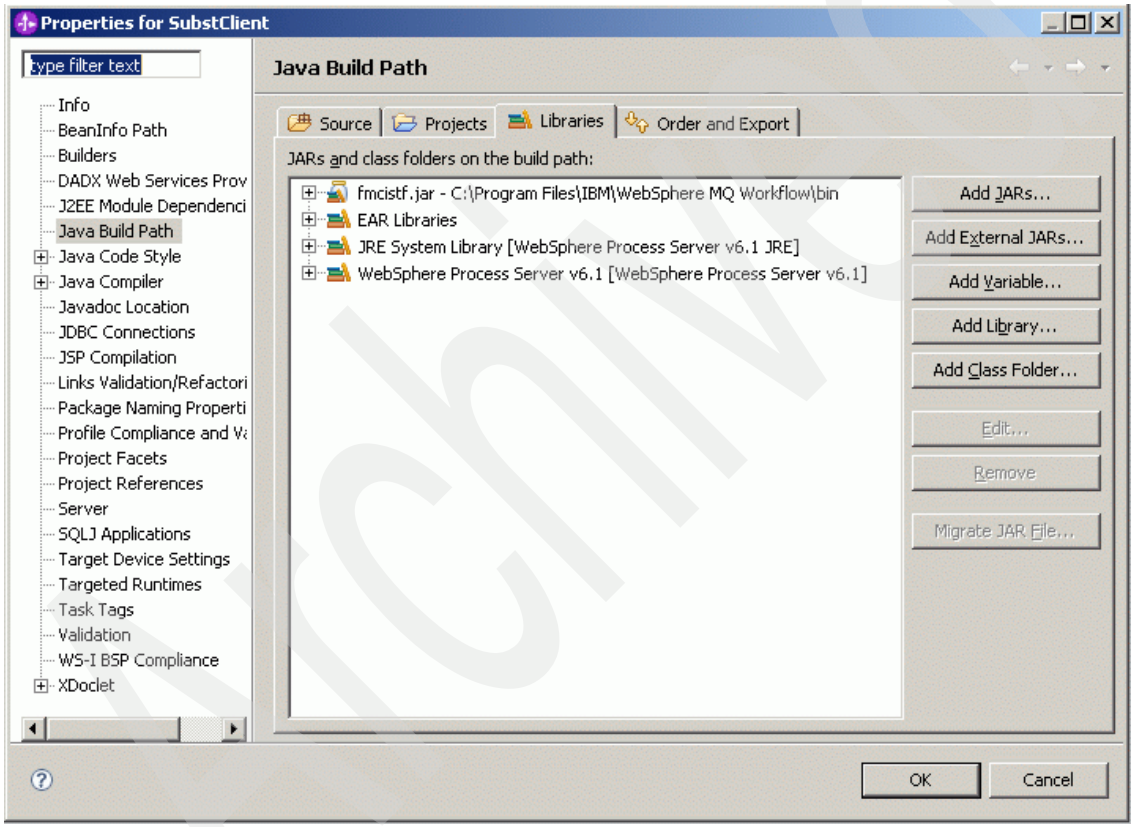


Figure 10-2 Java Build Path: Libraries

4. Click **OK** to save this definition.

10.3.4 Add the Java program to the application client project

To do this:

1. Check where your project SubstClient is located. In the Project Explorer window, right-click the project **SubstClient** and choose **Properties**. Note the value of the location, for example, with copy and paste.
2. Copy the Java program shown at the beginning of the sample (the file MigSubst.java) into the directory <SubstClient project location>/appClientModule/.
3. In the Project Explorer window, right-click the project **SubstClient** and choose **Refresh** to make this Java file known to the project.
4. In the Project Explorer, in the project SubstClient in the path appClientModule → appClientModule, open the file **MigSubst.java**.
5. Edit the following line:

   ```
   mqwfSession.logon("YourMQWFConfiguration", "ADMIN", "password",
   true);
   ```

   Insert the WebSphere MQ Workflow configuration name that you want to use and the user and password to connect to the WebSphere MQ Workflow Staff Administration API.
6. Save the file. When using the default settings of WebSphere Integration Developer, the projects are built automatically.

   The projects SubstClient and SubstClientEAR should now have no errors or warnings.

10.3.5 Start WebSphere Process Server

In the Servers tab of WebSphere Integration Developer, check the status of WebSphere Process Server V6.1. If the server status is not started, select **Start**. Wait until the server status has changed to *started*.

10.3.6 Run the program

To define how the sample program is started in WebSphere Integration Developer:

1. In the Project Explorer, select the project **SubstClientEAR**.
2. In the Run drop-down menu, select **Run**. This opens the Create, manage, and run configurations pop-up.
3. In this window, click the New launch configuration icon to define the settings of the sample program.

4. In the Application tab, set the name value to **SubstClient**.
5. Select **Enable application to connect to a server**. Check your settings with Figure 10-3.

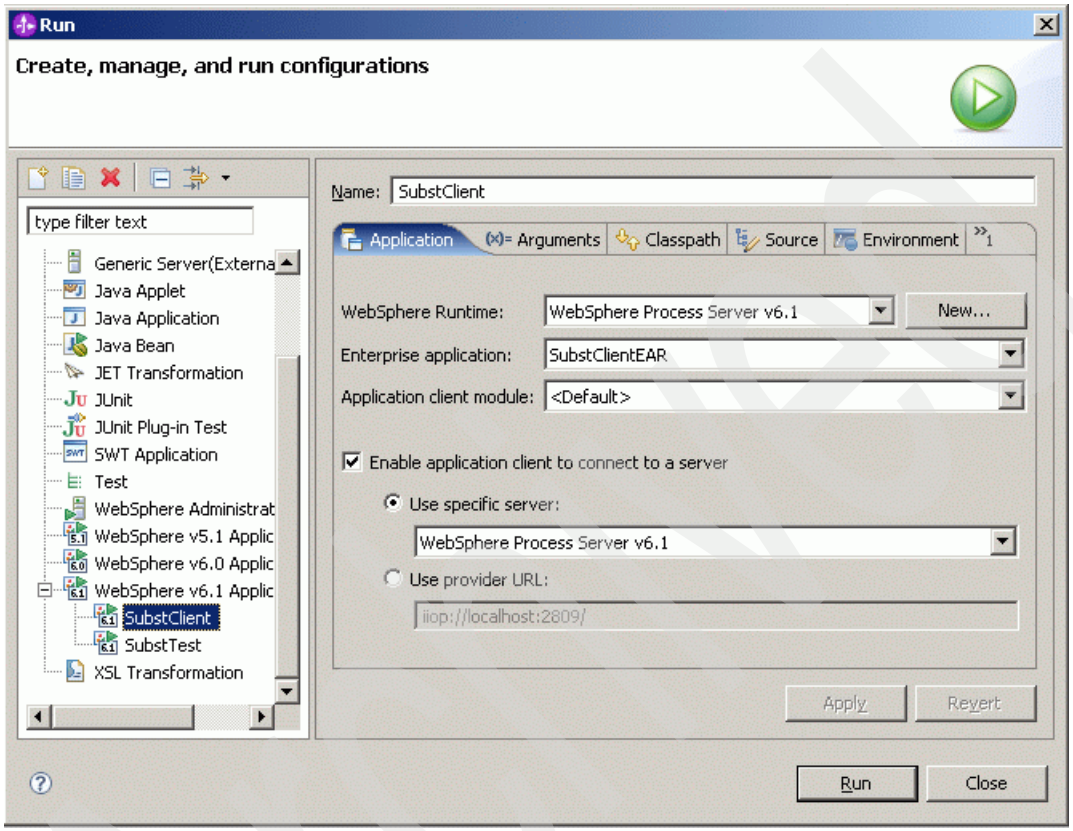


Figure 10-3 Run configuration: Application Settings

6. In the Arguments tab, in Program arguments, if you want to avoid having too many tracing messages on the console, you can remove the line -CCverbose=true.
7. In the Classpath tab, select the tree item **User Entries** and add the following external JAR files:
   - fmcistf.jar: The WebSphere MQ Workflow Staff Administration JAR. Depending on your database system and platform, add the JAR files for the appropriate JDBC driver. In our example this is db2jcc.jar and db2jcc_license_cu.jar.
   - To access the WebSphere Process Server Human Task Manager EJB API, insert taskapi.jar and task137650.jar.

This tab then looks like Figure 10-4.

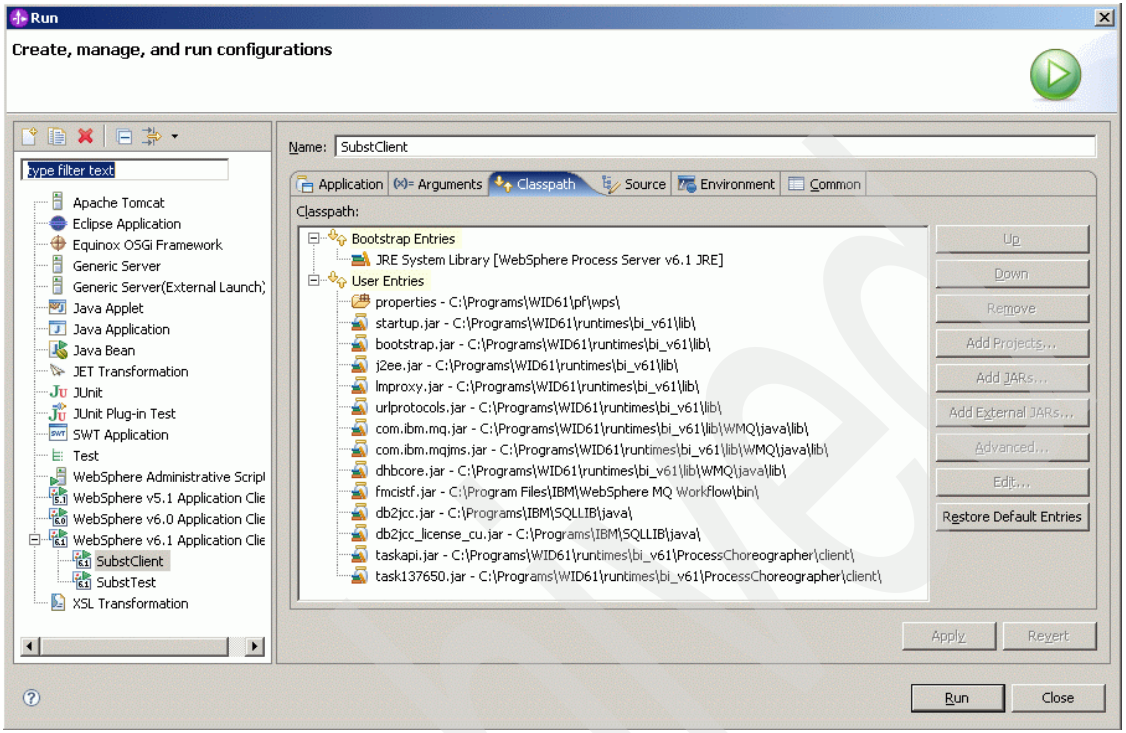


Figure 10-4 Run configuration: Classpath

8. Click **Run** to start the migration application. If security is switched on on your system, you are prompted for a login. The default values are user ID: admin and password: admin.

   In the Console window, you should get an output similar to Figure 10-5.

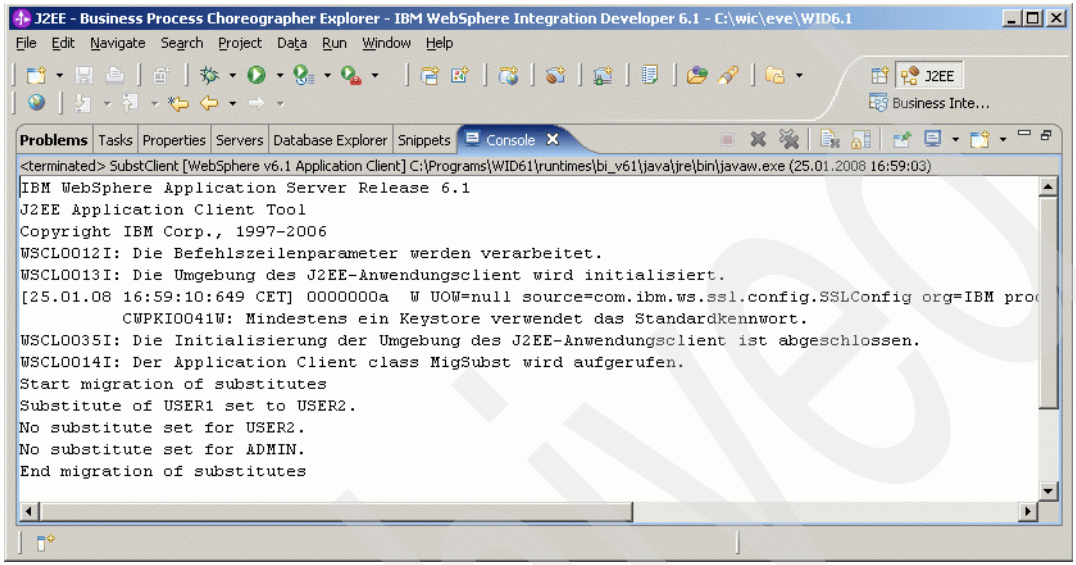


Figure 10-5 Console window

In this sample, three users are defined in WebSphere MQ Workflow and only for USER1. A substitute USER2 is defined.

9. Note that although the WebSphere user administration can be found in the WebSphere administration console, checking and setting substitution information can only be done with a Human Task Manager EJB client program such as BPC Explorer or this sample.

   To start the explorer, in the Servers tab, right-click the **WebSphere Process Server V6.1**, then click **Launch** → **Business Process Choreographer Explorer**.

   Now you are prompted for a login. Specify the same user ID and password as in the previous step.

   In the explorer select **Define Substitutes**, then type in a user name of interest (for example, USER1) and click **Lookup**.

A window similar to Figure 10-6 appears. Check that the substitute of USER1 is set as shown in Figure 10-6.

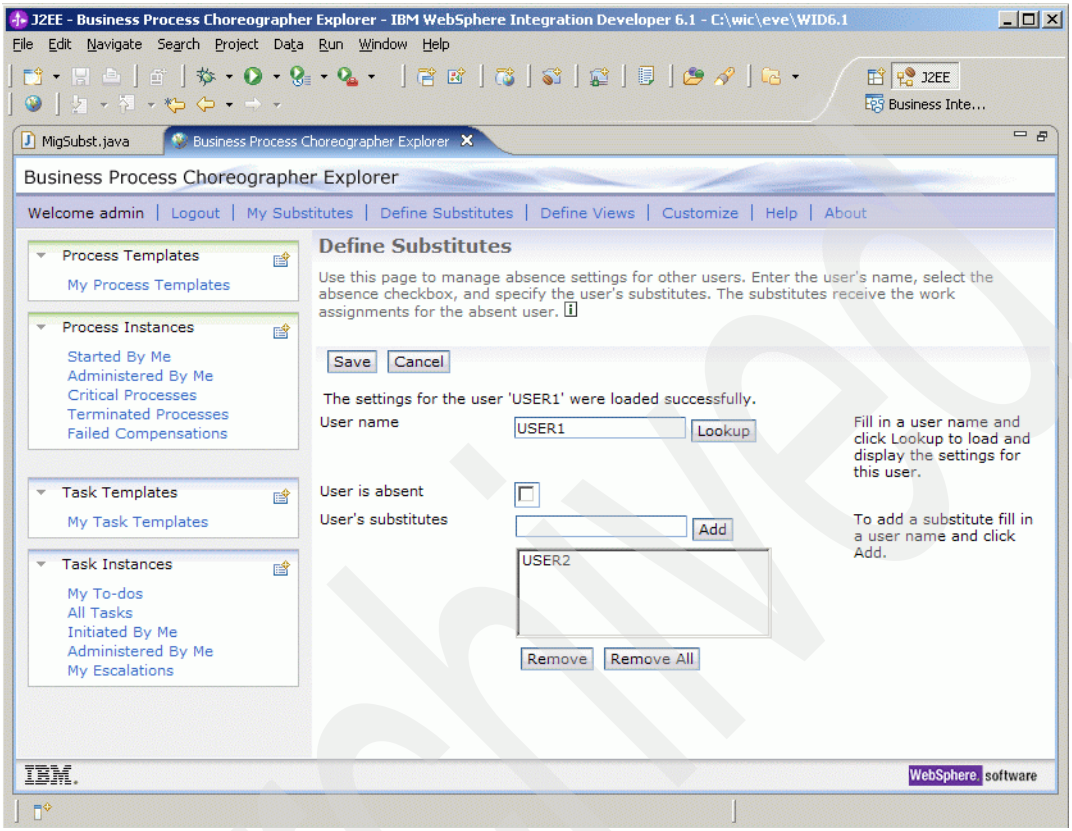


Figure 10-6 Business Process Choreographer Explorer: Substitutes

10.4 Customization options for Human Task Manager staff resolution

Since Human Task Manager people resolution is open for retrieving organizational information from various people directories, it provides various customization options. You can:

- Adapt existing people assignment criteria to your LDAP schema by creating a new people directory configuration.
- Create new people assignment criteria and support for them in the XSLT mapping file.

- Provide people resolution results via input messages or other process context data.
- Integrate a custom people directory with Human Task Manager by leveraging the support for custom repository adapters in WebSphere Virtual Member Manager (VMM).
- Perform workload balancing with Human Task Manager using the people assignment post-processor plug-in.

These customization options are comprehensively described in Authorization and staff resolution in Business Process Choreographer: Part 3: Customization options for staff resolution available at:

http://www.ibm.com/developerworks/websphere/techjournal/0712_lind/0712_lind.html

If you want to integrate a custom people directory you may want to leverage the support for custom repository adapters in VMM. This way you can integrate with both WebSphere Application Server security and Business Process Choreographer instance-based authorization by implementing a single interface, but with the powerful query functionality offered by VMM. For more details about configuring VMM and its RepositoryAdapter implementations, see the WebSphere Application Server Information Center.

11

Integrating back-end applications

This chapter describes the different integration variants for existing user-defined program execution server (UPES) environments and the available transition options. It provides the following information:

- User-defined program execution server implementations including:
  - Frameworks
  - Topologies
  - Dynamicity features
- UPES transition options and considerations including:
  - UPES programming model/architecture and transition pattern
  - Tools support
  - Dynamic UPES invocation implementation options and considerations

11.1 User-defined program execution server implementations

This chapter provides an overview over the different UPES integration variants to help analyze existing UPES environments and implementations for a transition to WebSphere Process Server.

11.1.1 UPES integration variants

The concept of a UPES allows for different integration possibilities. From WebSphere MQ Workflow Version 3.5 and later, a UPES framework is available as part of the product to provide a professional infrastructure for back-end integrations.

The UPES framework is a fully integrated and fully functional server framework. The framework is modular and expandable. Public interfaces are provided that can be used to plug in the functionality of a specific UPES. The UPES framework provides a server infrastructure for a UPES implementation. It can, for example, manage the queues and the incoming and outgoing messages, and can administer UPES instances. The UPES framework class library simplifies the development of a robust, scalable UPES. The framework includes a set of public interfaces that can be used to implement a UPES.

The UPES framework can be combined with additional layers, like a Java UPES and a Web services UPES. In addition to the UPES framework, other integration possibilities may have been used. The following list shows the most common UPES integration variants:

- Custom UPES implementation based on the WebSphere MQ Workflow UPES framework
- Custom UPES implementation based on the WebSphere MQ Workflow UPES framework

  On top of that a Java UPES is available as WebSphere MQ Workflow SupportPac WA05: *WebSphere MQ Workflow - Java UPES (toolkit):*

  `http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000664&loc=en_US&cs=utf-8&lang=en`

- Custom UPES implementation based on the WebSphere MQ Workflow UPES framework

  On top of that a Java UPES (see previous bullet) and a Web services UPES are available as WebSphere MQ Workflow SupportPac WA07: *WebSphere MQ Workflow - Web Services Process Management Toolkit:*

  http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000662&loc=en_US&cs=utf-8&lang=en
- Another WebSphere MQ product as UPES implementation, for example, WebSphere Message Broker (WMB)
- Custom UPES implementation based on none or on a custom UPES framework
- Custom J2EE UPES implementation

Note: A custom UPES implementation could implement a custom UPES framework.

Figure 11-1 illustrates the UPES implementation possibilities.

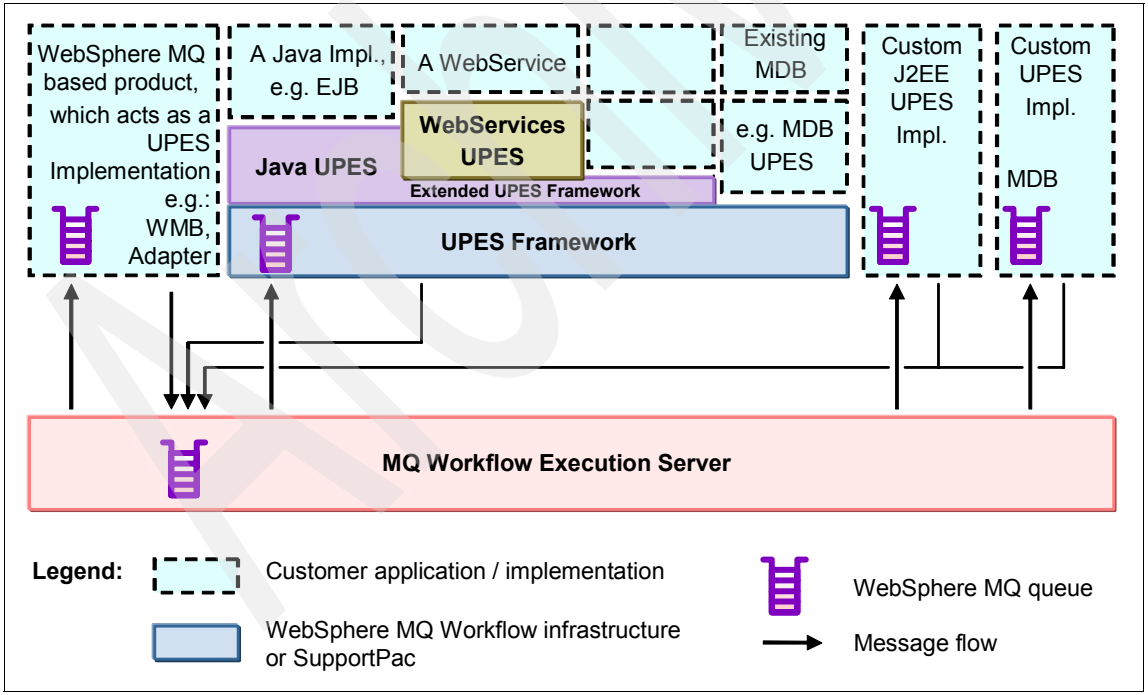


Figure 11-1 User-defined program execution implementation options

11.1.2 UPES implementation topologies

The WebSphere MQ Workflow architecture and the underlying WebSphere MQ infrastructure allow a wide range of scaling possibilities to invoke back-end applications via UPES. Some examples are:

- Single UPES with one instance based on one WebSphere MQ queue in one WebSphere MQ Workflow system
- Single UPES with multiple instances based on one WebSphere MQ queue in one WebSphere MQ Workflow system
- Multiple UPESs with multiple instances based on multiple WebSphere MQ queues in one WebSphere MQ Workflow system
- Multiple UPESs with multiple instances based on multiple WebSphere MQ queues in multiple WebSphere MQ Workflow systems
- UPES load balancing in a WebSphere MQ cluster between several WebSphere MQ Workflow systems

Figure 11-2 shows some UPES topology possibilities on a high-level.

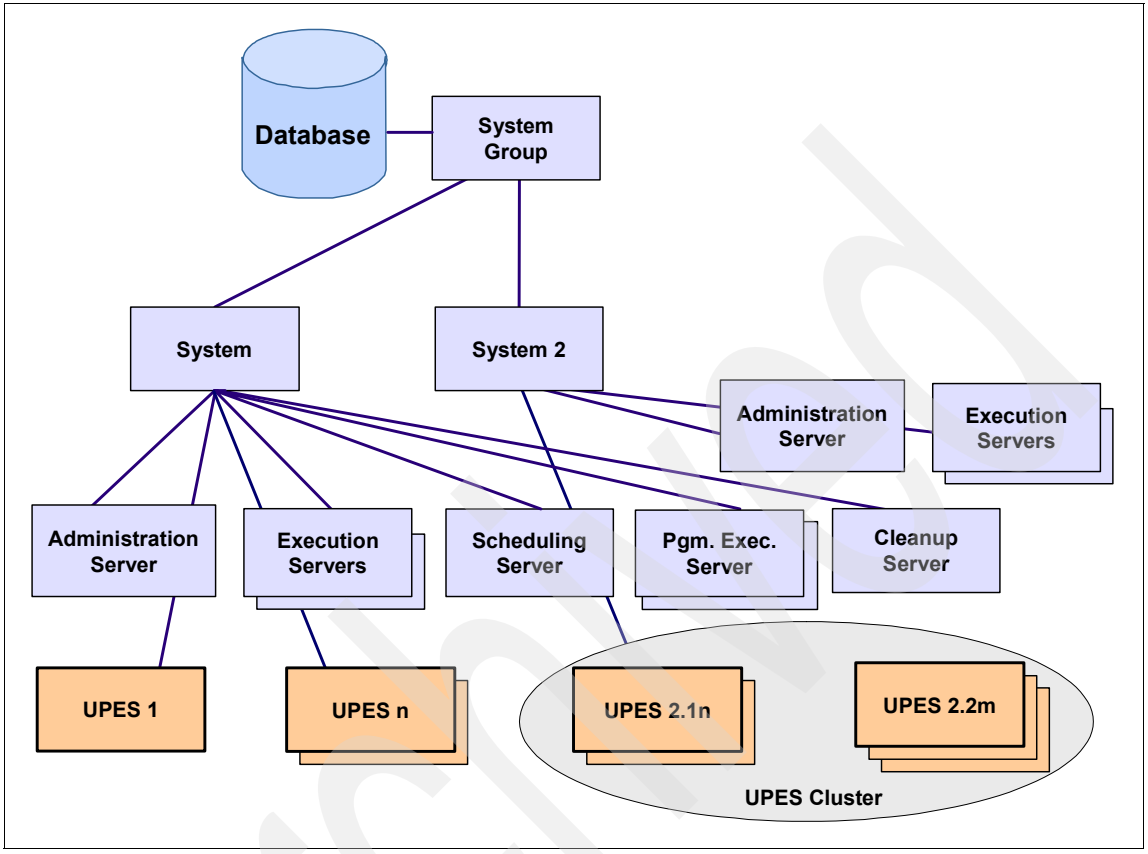


Figure 11-2 WebSphere MQ Workflow user-defined program execution server topology

UPES implementation dynamicity features

The WebSphere MQ Workflow programming model contains several dynamicity features for UPES invocations:

- Manual start/manual exit of an UPES activity.
- Late binding of a UPES. (The input container holds the name of the program execution server and during runtime execution the final UPES, WebSphere MQ queue/queue manager, is resolved.)
- Full qualified and partially qualified UPES name during modeling time. Name resolving during runtime. For example:

```
PROGRAM_EXECUTION_UNIT '<UPESName>.<System>.<SystemGroup>'
ROGRAM_EXECUTION_UNIT '<UPESName>'
```

This feature together with infrastructure customization enables, for example, a UPES load balancing within a system group.

11.2 UPES transition options and considerations

This section discusses different UPES transition options and considerations.

11.2.1 UPES programming model/architecture and transition pattern

In this section we discuss the UPES programming model/architecture and transition pattern.

WebSphere MQ Workflow and UPES

The WebSphere MQ Workflow UPES programming model is illustrated in the upper part of Figure 11-3 on page 259.

Activity implementations are started by the MQ Workflow execution server, which sends an invocation request message to a UPES. Using the message-based interface, WebSphere MQ Workflow sends that invocation request message (ActivityImplInvoke) in XML format to a user-defined WebSphere MQ queue (UPES input queue).

From the point of view of WebSphere MQ Workflow, the WebSphere MQ application listening on that queue must invoke the program that is modeled as the implementation of the activity. For doing so, all the necessary information is passed to the WebSphere MQ application (service application) by means of an XML message. The WebSphere MQ application must return with an appropriate XML response (ActivityImplInvokeResponse) message if requested by WebSphere MQ Workflow. The information in the MQMD ReplyToQ/ReplyToQMgr fields of the ActivityImplInvoke message specifies the queue and queue manager to which the response should be sent. The ReplyToQ information is in the case of WebSphere MQ Workflow the alias of the EXEXMLINPUTQ, for example, FMC.FMCGRP.FMCSYS.EXE.XML.

A WebSphere MQ Workflow correlation ID was provided as part of the ActivityImplInvoke message. The same correlation ID must be filled into the ActivityImplInvokeResponse message. This is required and later on enables WebSphere MQ Workflow to correlate the response message to the correct program activity instance.

One of the execution servers reads the message, correlates it, and completes the transaction with further process navigation or puts the activity into an error state.

WebSphere MQ Workflow optionally allows that a UPES is notified when the process is terminated (TerminateProgram message) and the program activity expires (ActivityExpired message). With custom coding a UPES framework could process such a message and the invoked service application could react on such messages. For example, it could do its own roll-back within the service application). In case of a TerminateProgram or ActivityExpired message WebSphere MQ Workflow does not expect a correlating response message. In such cases, WebSphere MQ Workflow no longer waits for a response message. If it gets one, it is discarded.

If WebSphere MQ Workflow receives an incorrect message (for example, an unknown XML tag), but with correct ReplyToQueue and ReplyToQManager information, then a GeneralError message is sent out to the UPES. If in addition a UPES is modeled as Version 3.4 or later, then a general error message contains the original UPES request message coded in hexadecimal character. This helps to analyze the root cause of the error.

WebSphere Process Server and UPES

If there is a requirement not to change the invoked back-end service application (UPES) when transitioning from WebSphere MQ Workflow to WebSphere Process Server, then the following transition steps apply to fulfill that requirement. They are also illustrated in the lower part of Figure 11-3 on page 259.

The process model must be converted or rewritten. Within the process model the WebSphere MQ Workflow-based program activity must be transitioned to a Business Process Choreographer-based invoke activity.

On the SCA layer an MQ JMS import is required to be able to communicate with a WebSphere MQ-based endpoint. The MQ JMS import can be generated with WebSphere Integration Developer. Make sure that the in the end-point configuration of the MQ JMS import on the Response tab under Receive Destination Properties the Target Client Type is set to the correct value. It must be MQ for a UPES with the “MESSAGE_FORMAT XML” (FDL setting) or “JMS” for “MESSAGE_FORMAT JMS_XML” (FDL setting).

A WebSphere MQ Workflow data binding is needed to convert all the necessary information such as WebSphere Process Server BOs/SDOs into a WebSphere MQ Workflow XML message (ActivityImplInvoke) and vice versa (ActivityImplInvokeResponse).

The Service Component Architecture (SCA) does not allow you to send more than one request message to the invoked service application, like a TerminateProgram message.

WebSphere Process Server's *dynamic* service selection and invocation based on SCA currently only supports SCA's native and Web Services bindings, but no MQ JMS bindings.

The destination (ReplyToQueue/ReplyToQManager) for the ActivityImplInvokeResponse messages changes from the WebSphere MQ Workflow XML input queue (EXEXMLINPUTQ) to the configured destination of the MQ JMS import.

The WebSphere MQ Workflow correlation ID is created by WebSphere Process Server and is still contained in the ActivityImplInvoke message, but this correlation ID is not used by WebSphere Process Server to correlate the ActivityImplInvokeResponse message. Instead of the WebSphere MQ Workflow-specific correlation ID, the WebSphere MQ-based correlation identifier and message identifier are used.

When the UPES creates a reply message, it respects any options that were set in the message descriptor (MQMD) or the MQ JMS header (RFH2) for the JMS-based communication of the message to which the UPES is replying. Report options specify the content of the message identifier (MsgId) and correlation identifier (CorrelId) fields. These fields allow WebSphere Process Server, when receiving the reply, to correlate the reply to the original request. In WebSphere Integration Developer this is called *response correlation scheme* (for example, *Request Message ID to Correlation ID*). For more details see:

- *WebSphere MQ Application Programming Guide,* SC34-6595

  http://www.ibm.com/support/docview.wss?uid=pub1sc34659500

- Using the report options in XML clients and UPES servers

  http://www.ibm.com/support/docview.wss?rs=795&context=SSVLA5&q1=report&uid=swg21171263&loc=en_US&cs=utf-8&lang=en

Summary

Figure 11-3 illustrates and summarizes the transition pattern.

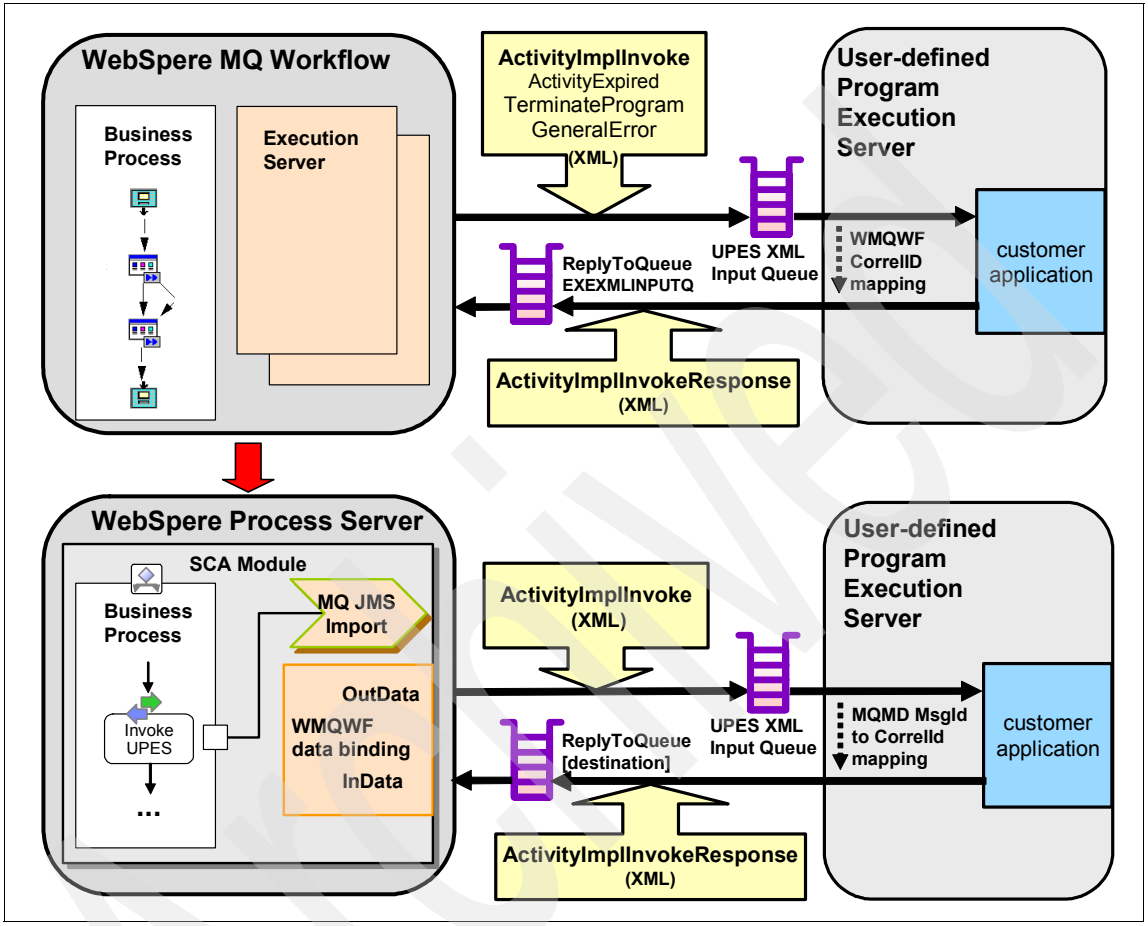


Figure 11-3 UPES transition and re-use pattern

The UPES transition and re-use pattern (re-use of back-ends) illustrated and described above applies to:

- The FDL2BPEL tool-based UPES transition with a built-in WebSphere MQ Workflow data binding
- A custom transition with a custom WebSphere MQ Workflow data binding

11.2.2 UPES-invoked applications

A WebSphere MQ Workflow program activity implemented as a *UPES activity* in most cases invokes a custom-written application. It is very important to understand the semantics of this custom application when doing the transition to decide to which activity type it should be mapped.

A UPES activity in WebSphere Process Server is typically implemented as an invoke activity, but there could be also a better-suited alternative implementation in WebSphere Process Server. This section provides some hints as to when an alternative should be considered or investigated. See Example 11-1.

Example 11-1 UPES implementation that natively does not map to a BPC Invoke activity

```
Some WebSphere MQ Workflow customer implemented a no-op application as a "UPES
activity" before WebSphere MQ Workflow offered an Empty (FMCINTERNALNOOP) activity.
The ideal transition to WebSphere Process Server is not implementing an Invoke
activity, but implementing it as an Empty activity.
```

The FDL2BPEL Conversion tool is only able to perform the default mapping because it does not know the semantics of the invoked customer application. In order to get the best fit, we recommend doing a detailed analysis of all the invoked applications. Table 11-1 shows some examples of the semantics of an application and provides mapping hints.

Table 11-1 UPES invoked applications: Application semantics versus activity type

UPES application semantics	Maps typically to following Business Process Choreographer activity
Remote application	Invoke activity
Database update	Invoke activity Information activity
Message Broker	Invoke activity
Other WebSphere MQ based applications	Invoke activity
Web service	Invoke activity
Business rule	Invoke activity
No-op application	Empty activity
Internal calculation	Java (Snipped) activity, other activity
E-mail escalation of overdue tasks	Task activity with escalation triggered e-mail

UPES application semantics	Maps typically to following Business Process Choreographer activity
Message data manipulation	Java (snipped) activity Assign activity
Process manipulation UPES acts as a client application and uses the API to interact with the workflow engine.	Java (snipped) activity Invoke activity
Event handling	Receive choice (pick) activity Receive Activity Event Handler
Java implementation	Invoke activity Java (snipped) activity
Adapter	Invoke activity

11.2.3 UPES model transition options

Table 11-2 shows different options, advantages, and considerations for the transition of a UPES activity from a modeling point of view.

Table 11-2 UPES model transition options

Option	Advantages	Considerations
Re-write Parts or the complete process model as part of the UPES activity transition	Natural fit in architecture and environment. Typically, the cleanest process model with no additional overhead regards the number of activities and variables.	
FDL2BPEL	Automatic conversion.	► Consider the documented limitations. ► Consider four variables for one UPES activity implemented as an invoke activity. ► Consider additional Java snipped activities, depending on data mapping. ► An additional option is to optimize the FDL-based process before transitioning.

Option	Advantages	Considerations
Combination of re-write and FDL2BPEL Example1: FDL2BPEL to transform the variable. Re-write the activity. Example 2: FDL2BPEL to transform the complete activity. Re-write (optimize) the converted outcome.	► Allows you to optimize (reduce) the number of predecessor and successor Java Snippet activities. ► Allows you to optimize the number and structure of variables. ► Allows you to integrate a different activity that fits the semantics of the invoked program.	

11.2.4 UPES model transition with FDL2BPEL Conversion tool

In this section we discuss a UPES model transition with the FDL2BPEL Conversion tool.

UPES analysis

Before transitioning a UPES with the FDL2BPEL Conversion tool the following must be analyzed to find out whether the modeled WebSphere MQ Workflow UPES qualifies for an automatic transition. See Table 11-3.

Table 11-3 UPES analysis

Topic to check	Documentation hints	Possible actions
Message layer: Check implemented message correlation mechanism for MessageID to CorrelID mapping.	FDL2BPEL documentation Chapter 5 of the *WebSphere MQ Application Programming Guide, Version 6.0*, SC34-6595-00.	Change it.
Message layer: Check for a generic hard-coded ReplyToQ/ReplyToQMgr setting.	WebSphere MQ Application Programming Guide, Version 6.0 (SC34-6595-00).	If ReplyToQ/ReplyToQMgr settings are hard-coded, then change it by using the WebSphere MQ MQMD fields for a generic approach.
Invoked application: Does the application expect and react on WebSphere MQ Workflow message types, such as ActivityExpired or TerminateProgram? Check first if these messages types are enabled. If yes check the invoked application.	The section “Messages sent to a UPES” in Chapter 33 of the *IBM WebSphere MQ Workflow Programming Guide, Version 3.6*, SH12-6291-10.	Undo-operation, for example compensation.

Topic to check	Documentation hints	Possible actions
Invoked application: Does the application use ExternalProcessContext message part?	See FmcXML-....xsd(s) in the bin directory of the WebSphere MQ Workflow installation for message structure. See WebSphere MQ Workflow manuals for *process context*.	Currently not supported by the FDL2BPEL tool.
Invoked application: Does the application use ImplementationData message part?	See FmcXML-....xsd(s) in the bin directory of the WebSphere MQ Workflow installation for message structure. See the WebSphere MQ Workflow Programming Guide for *implementation data*.	Currently not supported by the FDL2BPEL tool.
Invoked application: Does the application use _Process message part? _Process contains the process instance name.	See FmcXML-....xsd(s) in the bin directory of the WebSphere MQ Workflow installation for message structure.	Currently not set by the Runtime environment. Add a Java Snipped activity to set this message part.
Invoked application: Does the application interact with WebSphere MQ Workflow via the API?		Change the application and implement the WebSphere Process Server API instead or look for other alternatives.
Process model: Is the UPES activity modeled as start manual or exit manual? Buildtime: Activity property → Start/Exit → Start manual/Exit manual. FDL: START MANUAL ... EXIT MANUAL ... Note: A WebSphere Process Server invoke activity does not support a manual start and exit.	WebSphere MQ Workflow Buildtime online help.	Change the WebSphere MQ Workflow process model. Instead of one program activity for a UPES, create two or three activities: ► Human task for manual start ► Automatic UPES activity ► Human task for manual exit. With these changes FDL2BPEL should create corresponding process artifacts.

Topic to check	Documentation hints	Possible actions
Process model: Is the UPES activity modeled with dynamic endpoint selection? Buildtime: Activity property → Execution → From container. FDL: PROGRAM_EXECUTION_UNIT TAKEN_FROM "<container member>"	► WebSphere MQ Workflow Buildtime online help. ► *IBM WebSphere MQ Workflow Getting Started with Buildtime Version 3.6*, SH12-6286-10	Currently not possible with FDL2BPEL Conversion tool. Alternative: Rewrite in WebSphere Process Server with dynamic endpoint selection (see 11.2.5, “Dynamic UPES invocation implementation options” on page 264) or use other WebSphere Process Server rewrite alternatives.

When analyzing an existing UPES implementation, use the questionnaire provided in “UPES questionnaire” on page 426.

UPES transition

When a UPES qualifies for an FDL2BPEL transition, refer to the documentation of the FDL2BPEL Conversion tool for conversion details: SupportPac WA73: FDL2BPEL Conversion Tool (Version 6.1 or later):

http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

11.2.5 Dynamic UPES invocation implementation options

A dynamic UPES invocation implementation, also called late binding of a UPES invocation implementation, has the following characteristics:

- The modeled program activity is not associated with a specific UPES. A specific input container member (variable part) is defined that can hold the name of the UPES.
- During runtime execution of the process the name of the UPES must be set in the input container of the UPES program activity before the process navigates to the UPES activity (activity state inactive).
- If the workflow engine has navigated to the UPES program activity the content of the input container member that holds the UPES name is evaluated. If the UPES is defined by the name and the system, then the UPES of this system

is addressed. This applies to both the explicit and implicit (from container) UPES specification. If the UPES is defined only by its name, then the UPES of the current system is used.

- The execution mode is set to synchronous.

The advantage of a dynamic UPES invocation implementation is:

- The versioning of process models and UPES definitions is de-coupled.
- The dynamic invocation aspect allows you to model very flexible processes and to implement a highly dynamic behavior.

How a transition expert can identify a dynamic UPES

Look at the process model in the modeling tool, for example, in the WebSphere MQ Workflow Buildtime. Make sure that the Buildtime not only contains the process model, but also the WebSphere MQ Workflow environment settings. They can be found on the Network tab of the WebSphere MQ Workflow Buildtime navigation pane, as shown Figure 11-4.

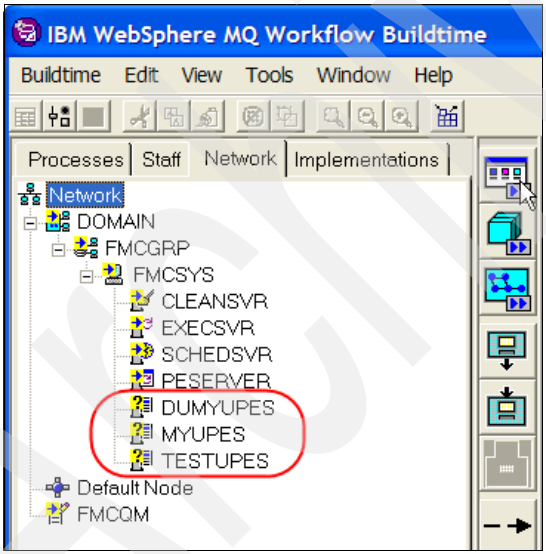


Figure 11-4 WebSphere MQ Workflow Buildtime: UPES environment definition

To analyze the modeled type of a UPES program activity, open a process model. Right-click the program activity in the process diagram and click **Properties**. The program activity properties pane appears. In the program activity properties pane select the **Execution** tab. The (user-defined) program execution server can be specified in one of these fields:

- Server field
- From container field

A dynamic UPES is defined if the radio button From container is selected and an input container member is specified in the entry field, as shown in Figure 11-5.

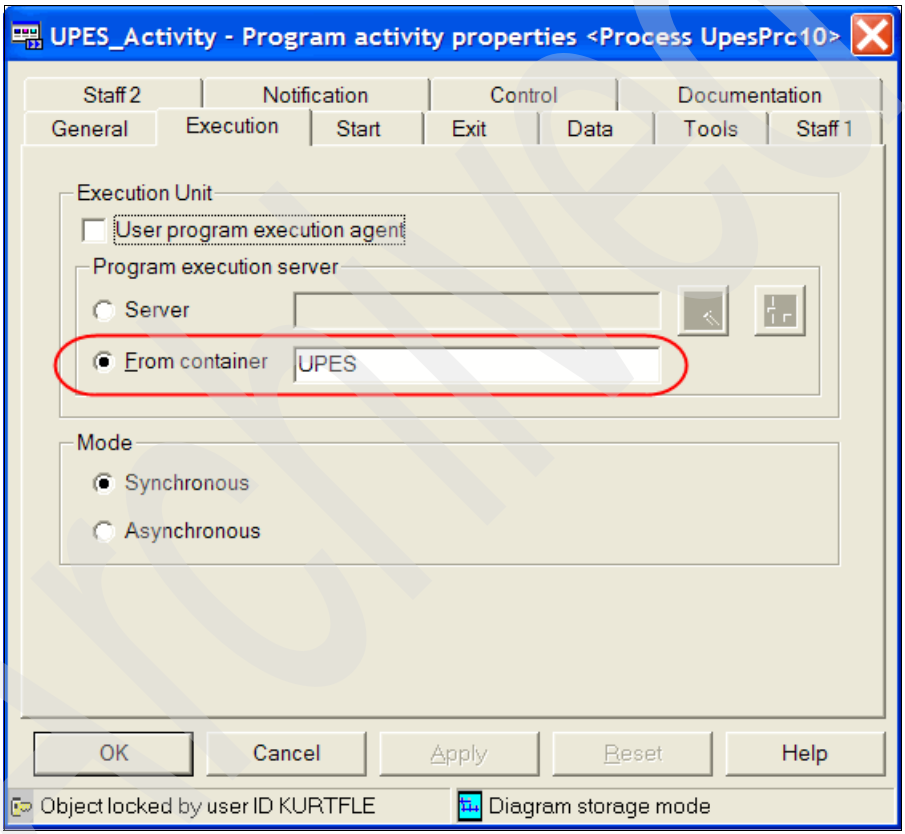


Figure 11-5 WebSphere MQ Workflow program activity properties: UPES definition

Another possibility to analyze whether a dynamic UPES model is given is to browse the FDL of a modeled process and to search for the PROGRAM_EXECUTION_UNIT TAKEN_FROM keyword.

Figure 11-6 shows an example as an FDL extract of a container (variable) definition and a program activity. It contains the From container definition (PROGRAM_EXECUTION_UNIT TAKEN_FROM) with the container member “UPES” and the relationship between Container (STRUCTURE) definition and the program activity (PROGRAM_ACTIVITY).

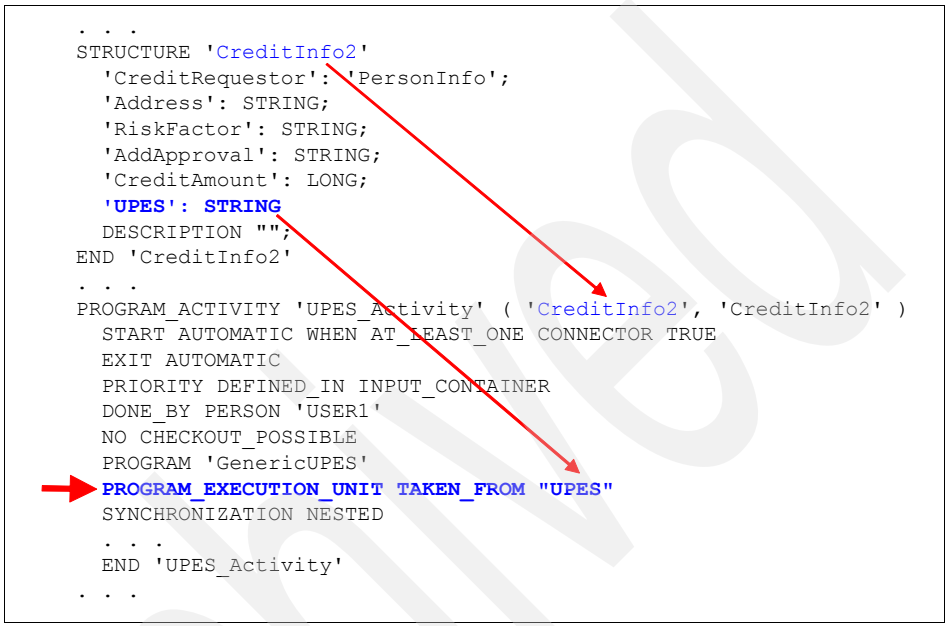


Figure 11-6 Dynamic UPES specification as FDL extract example

If the FDL2BPEL Conversion tool is used to convert the WebSphere MQ Workflow process model into a BPEL process model, a conversion result message (shown in Example 11-2) appears.

Example 11-2 FDL2BPEL conversion result message: Dynamic UPES

```
CWWBM0080W: UPESPrc10_includingUPESs.fdl(108 et seqq.):Option "Program execution
server from container" of program activity "UPES_Activity" cannot be mapped to BPEL.
```

Then a dynamic UPES is identified. This also means there is no tool-based conversion support due to the fact that there is no direct equivalent in BPEL/SOA. However, there are other alternatives to achieve a similar behavior. These alternatives are discussed in this section.

Transition options and considerations

In WebSphere Process Server dynamic service selection and invocation based on SCA is currently only supported for SCA native and Web Services bindings, but not for MQ JMS binding.

If it is a requirement not to change the invoked back-end application, but to transition the WebSphere MQ Workflow dynamic UPES-based process model to a WebSphere Process Server-based process model and to achieve flexibility similar to the previously described WebSphere MQ Workflow capability. Then you must evaluate several transition options. Three of them are discussed in this section:

- UPES invocation via a dynamic subprocess
- UPES invocation via a mediation flow
- UPES invocation via a WebSphere Message Broker flow

The data binding part (BO to WebSphere MQ Workflow XML output message format and UPES XML response message to BO) is not part of this discussion, but it is similar to the one described in the FDL2BPEL Conversion tool and offered by WebSphere Integration Developer and WebSphere Process Server as MQ Workflow data binding.

Figure 11-7 shows the transition options used to describe the WebSphere MQ Workflow UPES scenario.

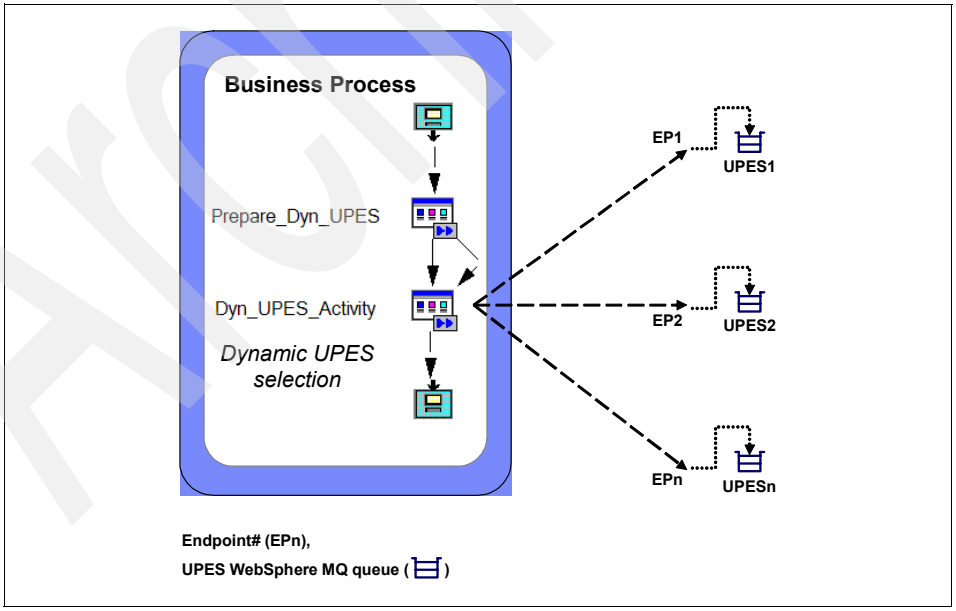


Figure 11-7 WebSphere MQ Workflow dynamic UPES scenario

In the life cycle of a WebSphere MQ Workflow process the following dynamic UPES-related runtime steps occur:

1. UPES registration

   UPESs are already registered in the WebSphere MQ Workflow Runtime environment with UPES name, queue name, optional queue manager name, and other information.

2. UPES name computation (preparation)

   During execution of this WebSphere MQ Workflow process any preceding activity could compute the UPES name based on previous processing and pass that information into the input container of the dynamic UPES activity. It is also possible that the UPES name is passed from the source node to the input container of a dynamic UPES activity via a data connector.

3. UPES endpoint look-up

   The WebSphere MQ Workflow engine reads the UPES name from the input container. It then does a WebSphere MQ Workflow internal lookup to get the matching endpoint (queue name and queue manager name) for a dynamically provided UPES name.

4. UPES invocation

   The WebSphere MQ Workflow execution server sends an activity implementation invoke message (ActivityImplInvoke) to the endpoint (WebSphere MQ queue) evaluated in the previous step, and later handles the response message.

Option 1: UPES invocation via a WebSphere Message Broker flow

In this option the UPES invocation is delegated to a WebSphere Message Broker flow, which allows for an independent versioning of the main process. For more details see Figure 11-8 and description below.

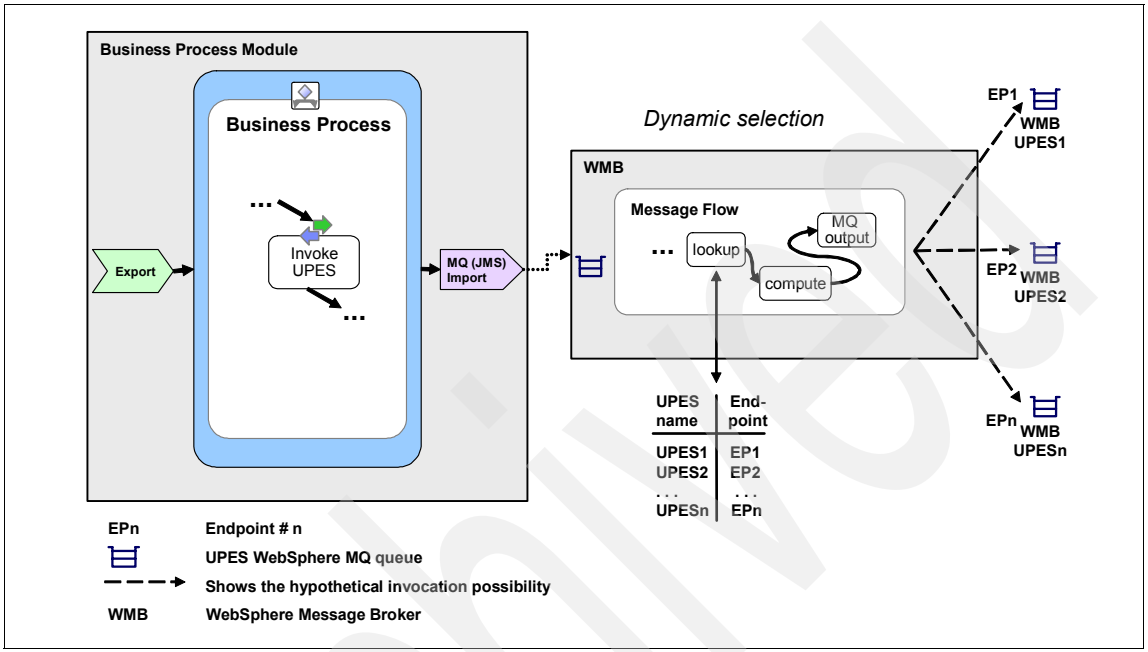


Figure 11-8 UPES invocation via a WebSphere Message Broker flow

When using this option the life cycle of a WebSphere Process Server contains the following dynamic UPES-related runtime steps:

1. UPES registration

   UPESs are not *registered* in the WebSphere Process Server Runtime environment.

2. UPES name computation (preparation)

   Identical to that in WebSphere MQ Workflow.

3. UPES endpoint look-up

   A WebSphere Message Broker flow does an endpoint look-up via a lookup, followed by a compute node that is needed to make use of the lookup result in an MQ node. The endpoint information could be stored, for example, in a simple table, as a flat file, database entry, or in WebSphere Service Registry and Repository (WSRR).

4. UPES invocation

   The WebSphere Process Server main process invokes a WebSphere Message Broker flow via an Invoke activity. The WebSphere Message Broker flow dynamically invokes the final WebSphere MQ endpoint based on the endpoint look-up result.

Summary

Options 2 and 3 have a clear separation of dynamic selection/routing as part of bus logic.

All three options allow versioning of main business process without re-deployment of UPES/UPES invocation.

Option 3 may be considered when a UPES is already implemented as WebSphere Message Broker.

Option 2: UPES invocation via a dynamic subprocess selection

In this option the UPES invocation is wrapped into a single loosely coupled (not wired) subprocess, which allows for an independent versioning of the main process. Each subprocess acts as a proxy for a UPES. For more details see Figure 11-9 and the description below.

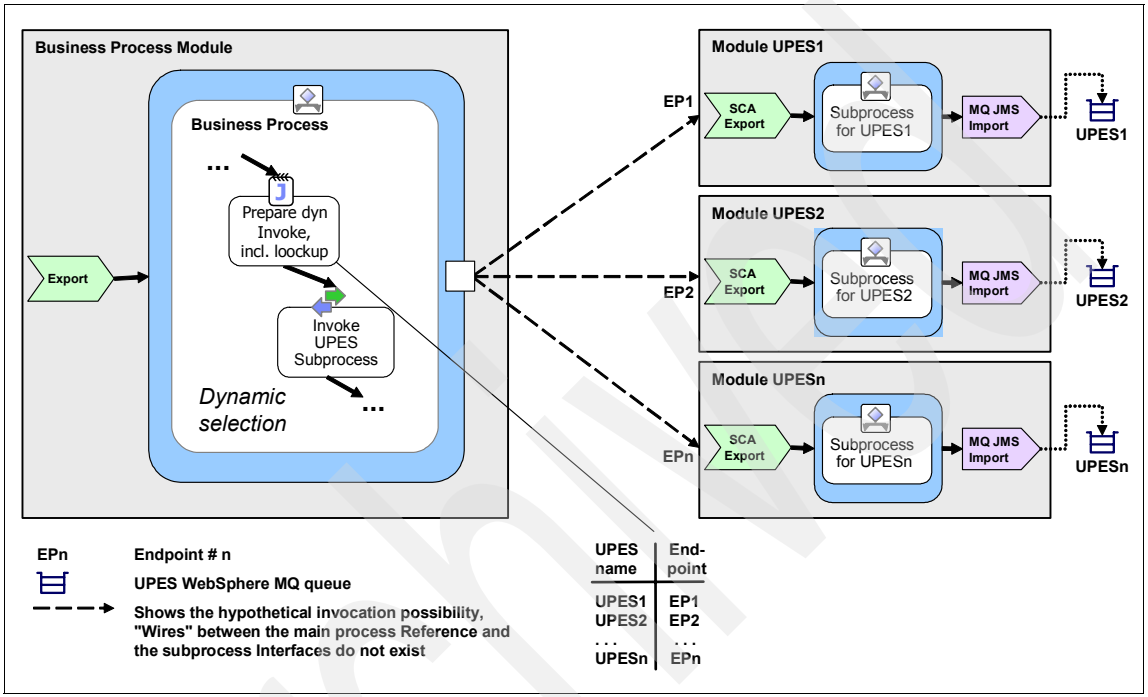


Figure 11-9 UPES invocation via a dynamic subprocess selection

When using this option the life cycle of a WebSphere Process Server process contains the following dynamic UPES-related runtime steps:

1. UPES registration

   UPESs are registered in the WebSphere Process Server Runtime environment as subprocesses. For each UPES a subprocess must be deployed that contains one invoke activity for a single UPES invocation.

2. UPES name computation (preparation)

   Identical to that in WebSphere MQ Workflow.

3. UPES endpoint look-up

   The WebSphere Process Server engine reads the UPES name out of a variable. A preparation activity (for example, an Assign or Java Snippet activity) does a lookup to get the matching endpoint (UPES subprocess

name) for a dynamically provided UPES name. The endpoint information could be stored, for example, in a simple table, as a flat file, database entry, or even in WebSphere Service Registry and Repository (WSRR).

4. UPES invocation

   The WebSphere Process Server main process dynamically invokes a UPES subprocess via an invoke activity and passes all the needed data for the UPES to the subprocess. Each invoked UPES subprocess contains an invoke activity, which is bound via an MQ JMS import to the final, fixed destination (queue name and queue manager name). The subprocess navigates to the invoke activity and does the final invocation of the UPES invoke activity.

Option 3: Dynamic UPES invocation via a mediation flow

In this option the UPES invocation is delegated to a mediation flow of WebSphere Enterprise Service Bus as part of the WebSphere Process Server capabilities to do a mediation from an SCA native binding to a JMS MQ binding. This option allows for an independent versioning of the main process. Figure 11-10 shows the dynamic UPES invocation via a mediation flow.

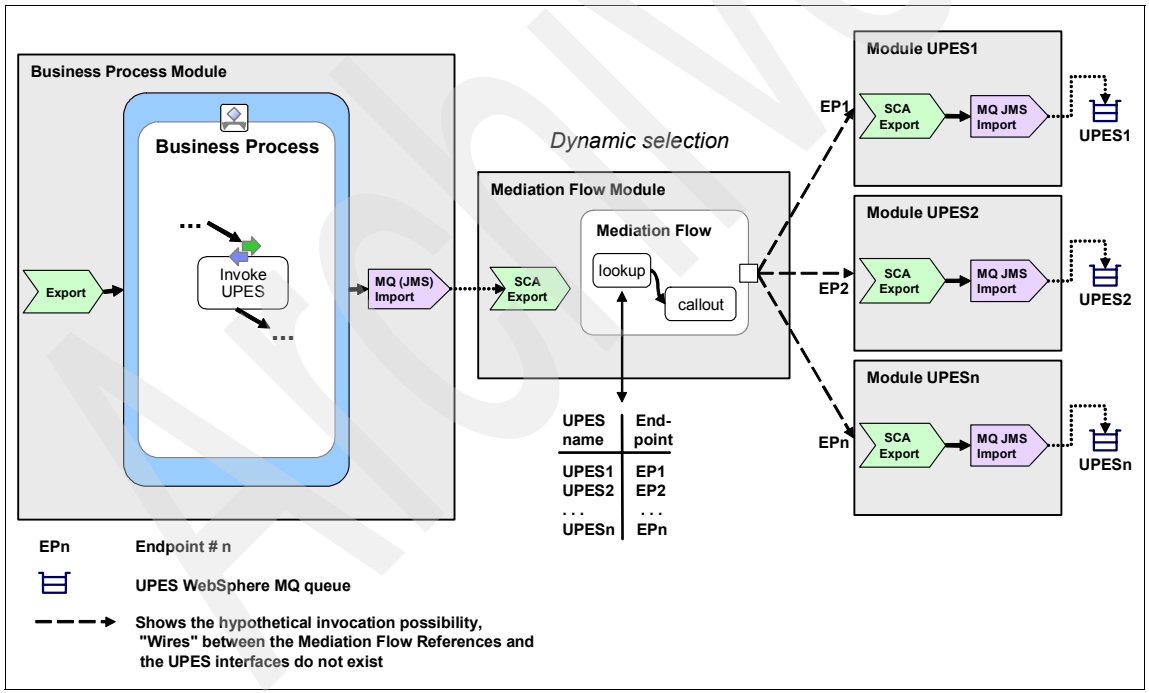


Figure 11-10 Transition option: Dynamic UPES invocation via a mediation flow

When using this option the life cycle of a WebSphere Process Server contains the following dynamic UPES-related runtime steps:

1. UPES registration

   UPESs are registered in the WebSphere Process Server Runtime environment as a set of modules, each containing an SCA export representing the endpoint and an MQ JMS import. For each UPES such a module must be deployed.

2. UPES name computation (preparation)

   Identical to that in WebSphere MQ Workflow.

3. UPES endpoint look-up

   A mediation flow contained in a separate module does an endpoint look-up via a lookup primitive. The endpoint information could be stored, for example, in a simple table, as a flat file, a database entry, or in WebSphere Service Registry and Repository (WSRR).

4. UPES invocation

   The WebSphere Process Server main process invokes a mediation flow via an invoke activity. The mediation flow dynamically invokes the final endpoint based on the endpoint look-up result via a callout primitive. The endpoint is represented by an SCA export of the UPES module. The final, fixed destination (queue name and queue manager name) is handled via an MQ JMS import.

11.2.6 Re-implementing a UPES as a native service

This topic is not covered in this book because there are already published samples available, such as *Service invocation* sample of the *Business Process Management Samples & Tutorials* - Version 6.1, available at:

`http://publib.boulder.ibm.com/bpcsamp/index.html?gettingStarted&gettingStarted/serviceInvocation.html`

See also the related sections “Overview,” “BPEL Process, Build it Yourself,” “Run the Sample,” “and Download” at the above URL.

For further reading see:

- Best practices for Web services: Part 1, Back to the basics

  `http://www.ibm.com/developerworks/webservices/library/ws-best1/`

- *Web Services Architecture* W3C Working Group Note 11 February 2004

  `http://www.w3.org/TR/ws-arch/`

12

Implementing clients based on application programming interfaces

This chapter provides an overview of the interfaces of WebSphere MQ Workflow and WebSphere Process Server and how these interfaces are mapped. It describes the following:

- Client implementation options: Process concepts, such as:
  - Process template queries
  - Starting a process
  - Process instance queries
  - Main process state transitions
  - Managing the life cycle of a process
- Client implementation options: Activity concepts, such as:
  - Activity-related queries
  - Main activity state transitions
  - Managing the life cycle of an activity
  - Activity repair options

- Client implementation options: Task concepts, such as:
  - Task-related queries
  - Main task state navigation
  - Managing the life cycle of a to-do task, to-do task repair
  - Delegation of work concepts
- Data handling in WebSphere MQ Workflow and how to achieve similar functionality in WebSphere Process Server
- Queries, such as methods to query business-process, activity, and task-related objects in the database to retrieve specific properties of these objects
- Authentication implementations in both products and how they can be mapped
- Authorization management in both products and how they correspond
- Application programming interface package overview for WebSphere MQ Workflow and WebSphere Process Server

12.1 Client implementation options: Process concepts

This section describes and compares process templates and process instances of WebSphere MQ Workflow and WebSphere Process Server**.** It discusses API methods to locate and influence the behavior of templates and instances.

A process template represents a modeled business process. After modeling it is deployed to the Runtime environment:

- In WebSphere MQ Workflow, this is done using an FDL import utility.
- In WebSphere Process Server, it is deployed using the administration console or by other means, such as an enterprise application.

To make a deployed process template visible and usable for an application programming interface-based client application, additional administrative actions are needed:

- In WebSphere MQ Workflow, a *translate* step is needed, which puts the process templates into the translated state.
- In WebSphere Process Server, a start of the enterprise application is needed, which puts the process templates into the started state.

When these steps have been done, a programmer can use the provided query functionality to list a filtered and ordered set of process templates as a result of the API call.

This section covers the API functionality and aspects of typical life-cycle actions on processes and compares the WebSphere MQ Workflow API methods and the process behavior with the ones of WebSphere Process Server.

12.1.1 Process template queries

A process template query API method retrieves process templates that are persistently stored in the database. The result of such a query can be used to extract further details, like process template name or other attributes, which are needed to invoke a process (create and start a process instance).

WebSphere MQ Workflow

WebSphere MQ Workflow provides query API methods for process templates, as shown in Example 12-1.

Example 12-1 WebSphere MQ Workflow queryProcessTemplate API method

```
ProcessTemplate[] queryProcessTemplates(
                    java.lang.String filter,
                    java.lang.String sortCriteria,
                    java.lang.Integer threshold )
```

WebSphere Process Server

WebSphere Process Server provides similar API methods, as shown in Example 12-2.

Example 12-2 WebSphere Process Server queryProcessTemplate API method

```
ProcessTemplateData[] queryProcessTemplates(
                        java.lang.String whereClause,
                        java.lang.String orderByClause,
                        java.lang.Integer threshold,
                        java.util.TimeZone timeZone )
```

The result set of the above/listed API methods may differ because of the valid-from attribute restriction in WebSphere MQ Workflow. This means that a WebSphere MQ Workflow-based queryProcessTemplates() call returns only process templates currently valid (no past or future valid-from-based process templates). If the same behavior is needed in WebSphere Process Server, then the where-clause argument shown in Example 12-3 must be added.

Example 12-3 Where-clause statement for valid process templates in WebSphere Process Server

```
PROCESS_TEMPLATE.STATE IN (1) AND PROCESS_TEMPLATE.VALID_FROM = (SELECT
MAX(VALID_FROM) FROM PROCESS_TEMPLATE WHERE NAME=PROCESS_TEMPLATE.NAME
AND VALID_FROM <= CURRENT_TIMESTAMP)
```

This enables the client applications to select valid process templates for further processing, which is, for example, the creation and the start of a process instance that the logged-on user is authorized to use.

12.1.2 Starting a process

A process instance comes into existence when an API method that can start a process is invoked. This applies to WebSphere MQ Workflow as well as to Business Process Choreographer of WebSphere Process Server.

The differentiators in this area are the additional enhanced Business Process Choreographer functionality and the technology of WebSphere Process Server, which are listed Table 12-1.

Table 12-1 Functional differentiator for starting a process

Functionality	WebSphere MQ Workflow	WebSphere Process Server
Unique process starting service (starting activity)	Yes	Yes
Non-unique process starting services (process starts with multiple receive or pick activities)	No	Yes
Long-running process	Yes	Yes
Microflow	No	Yes

These additional functions and the new technology require additional capabilities for starting process instances.

WebSphere MQ Workflow

To create and start a process instance in WebSphere MQ Workflow, typically a process template must be located via the queryProcessTemplates() action API. The returned result can be displayed with the name() accessor API method, for example, as shown in Figure 12-1.

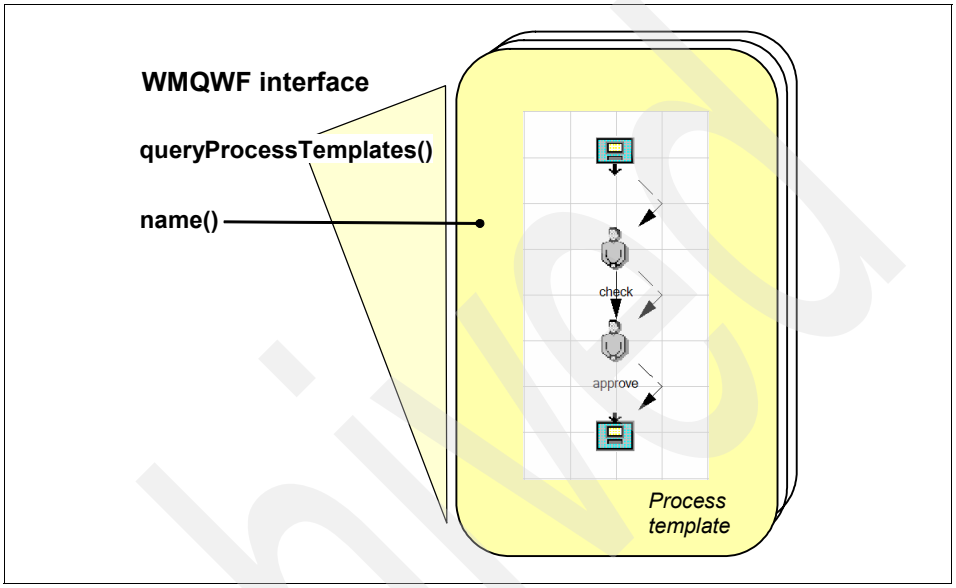


Figure 12-1 WebSphere MQ Workflow process start option

Knowing the name of different process templates enables the application developer to implement the required API methods for the creation and start of process instances. Example 12-4 shows the alternatives. For more details refer to the WebSphere MQ Workflow API documentation.

Example 12-4 WebSphere MQ Workflow process create start options

```
WebSphere MQ Workflow runtime API
Package com.ibm.workflow.api
Interface ProcessTemplate / Interface ProcessInstance

  createInstance()   +   start() / start2() / start3()
  createAndStartInstance()
  createAndStartInstance2()

  executeProcessInstance()
  executeProcessInstance2()
  executeProcessInstance3()

Inbound XML interface

  <ProcessTemplateCreateInstance>
  <ProcessTemplateCreateAndStartInstance>
  <ProcessTemplateExecute>
```

WebSphere Process Server

Depending on the process model conversion from WebSphere MQ Workflow to WebSphere Process Server, you may end up with a process model with one or more starting services (starting activities). If we combine this with the fact that the Business Process Choreographer consists of two components, namely the Business Flow Manager and the Human Task Manager, each with its own interfaces, we end up with different process start scenarios, for example:

- A process start with a unique starting service using the Business Flow Manager interface with the method signature that requires a process template name as a parameter. This is the WebSphere MQ Workflow-invocation style, as described in more detail here:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_macste.html

- A process start with a non-unique starting service using the Business Flow Manager interface. In this case it is required to explicitly identify the starting activity, as described in more detail here:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_macste_nonu.html

Note: The example in the above-referenced link uses a different sendMessage(...) signature than in Figure 12-2 on page 283.

- A process start through an inline invocation task. An inline invocation task can be specified in the context of a receive or pick activity, or an event handler in a business process. The task does not get a representation on the SCA level. Instead, it is part of the SCA component that represents the business process. Nonetheless, the task acts as a client to the business process. Whenever the task is invoked by the Human Task Manager API, the task in turn invokes the business process in the same way it which it was invoked.
- A process start through a stand-alone invocation task. A stand-alone invocation task serves as an access component to an associated SCA service. The association with the service is defined on the SCA level. The task represents an SCA client that is wired to an SCA service component. The invocation of an invocation task involves both Human Task Manager and SCA levels. The invocation task itself is invoked by the Human Task Manager API, either synchronously or asynchronously. The task (SCA client) then invokes the associated SCA service component either synchronously if the task was invoked synchronously or asynchronously if the task was invoked asynchronously.

  The modeling of the association between the task and the service is done on the SCA level. The concepts and mechanisms provided by SCA are therefore available for connecting stand-alone invocation tasks and SCA service components. This includes the SCA means to define the following:

  – Wires that connect an SCA client reference and the interface of a service component.
  – SCA qualifier settings for component references and interfaces that control aspects such as interaction style, transaction behavior, and interaction reliability. You can find more information about synchronous and asynchronous interfaces in the WebSphere Process Server V6.1 information center under the following links:
    - Synchronous interface:

      http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6task_startsynch.html
    - Asynchronous interface:

      http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6task_startasynch.html

The process details, the required query, and the get API methods for the first three scenarios described are illustrated in Figure 12-2.

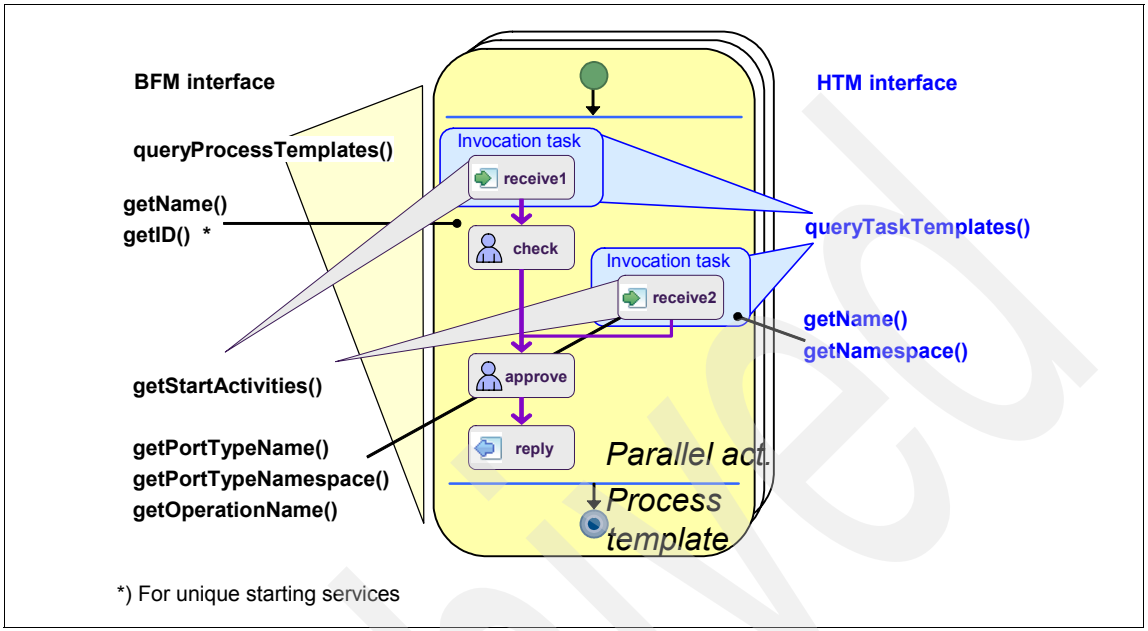


Figure 12-2 WebSphere Process Server process start options

The above-listed scenario-based API implementations are the preparation for the final start implementation. Depending on implementation requirements and on the start options the application developer may choose one of the following API options to finally create and start a process. See Example 12-5.

Example 12-5 WebSphere Process Server process start options

```
Business Flow Manager interfaces: EJB |  WS | JMS
sendMessage()                     yes   yes   yes
initiate()                        yes
call()*                           yes   yes   yes
callAsync()                             yes
*) for a micro flow  (synchronously)

Human Task Manager EJB interface
createAndStartTask()        // asynchronously
createTask() + startTask()  // asynchronously
createAndCallTask()         // synchronously
createTask() + callTask()   // synchronously

Service Component Architecture interface
invoke()
```

In this case the initiate() API method takes care that a valid process template gets started, one where the valid-from date is before or on the current date.

12.1.3 Process instance queries

To look for and to work with the started processes (process instances) query methods to retrieve stored information about business processes may be used. The query method returns objects according to the caller's authorization.

WebSphere MQ Workflow

WebSphere MQ Workflow provides process instance type-specific query API methods, as shown in Example 12-6.

Example 12-6 WebSphere MQ Workflow queryProcessInstances API method

```
ProcessInstance[] queryProcessInstances(
                    java.lang.String filter,
                    java.lang.String sortCriteria,
                    java.lang.Integer threshold)
```

WebSphere Process Server

WebSphere Process Server provides different methods. The most common one, besides the stored query() method, is shown in Example 12-7.

Example 12-7 WebSphere Process Server process instance query API method

```
QueryResultSet query(java.lang.String selectClause,
                    java.lang.String whereClause,
                    java.lang.String orderByClause,
                    java.lang.Integer skipTuples,
                    java.lang.Integer threshold,
                    java.util.TimeZone timeZone)
```

The above-mentioned query() is not type-specific as is the WebSphere MQ Workflow queryProcessInstances(). With query() you can query business-process and task-related objects in the database to retrieve specific properties of these objects (refer to database views). To get a WebSphere MQ Workflow-like type-specific process instance query result set you must specify it in the select-clause argument, as shown in Example 12-8.

Example 12-8 WebSphere Process Server select clause for a process instance query

```
selectClause = "DISTINCT PROCESS_INSTANCE.PIID"
```

For more information see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6query.html

12.1.4 Main process state transitions

The life cycle of a process instance is managed by the flow engine of WebSphere MQ Workflow or by the Business Flow Manager of WebSphere Process Server. During navigation through the business process and due to externally triggered

actions (API calls) a process instance may carry different states. Client applications typically display the state information via API methods to business users and administrators.

Process instance state diagrams provide application developers with an overview of the possible process instance states and the actions that are allowed in those states, provided that the appropriate authority has been granted.

WebSphere MQ Workflow

The typical life cycle of a WebSphere MQ Workflow process instance is navigation from the states Ready → Running → Finished → Deleted or Running → Finished → Deleted, depending on the API methods createInstance() or createAndStartInstance(). If the createInstance() API method is used, then an additional start() API call is needed. The state transition from Running → Finished cannot be influenced via an API call. The state transition from Finished → Deleted depends on modeling (KEEP_PROCESSES). If required, use the delete() API method. See Figure 12-3.

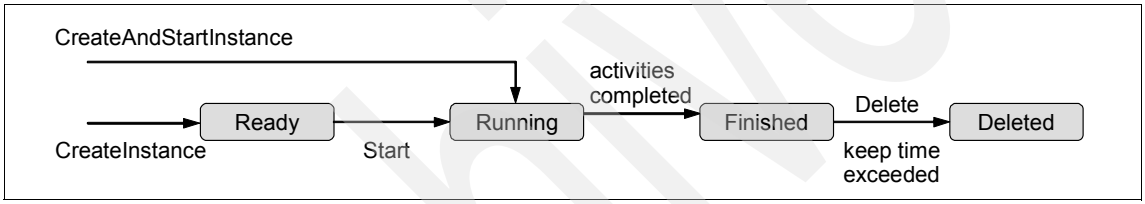


Figure 12-3 WebSphere MQ Workflow process instance state diagram extract

WebSphere Process Server

The typical life cycle of a WebSphere Process Server process instance is navigation from the states Running → Finished. Begin and end states, as in ready and finished in WebSphere MQ Workflow, do not exist. The state transition from finished to the end is similar to WebSphere MQ Workflow and depends on modeling. If required, also use the delete() API method. See Figure 12-4.

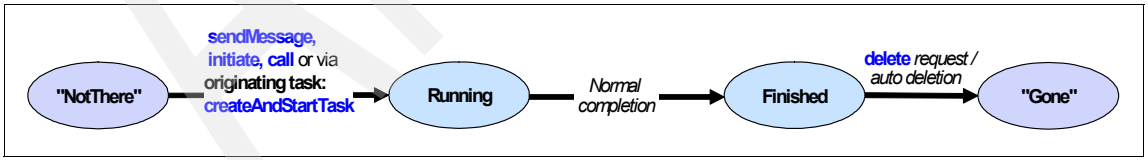


Figure 12-4 WebSphere Process Server process instance state diagram extract

12.1.5 Managing the life cycle of a business process

When a process is started, the navigation of the process instance continues until all of its activities are in an end state. Various actions can be taken on the process instance to manage its life cycle, like suspend and resume of a process. In this case the state transitions could get more complex compared to the previously illustrated figures.

The complete list of life cycle functions is shown in Table 12-2.

Table 12-2 Business process life cycle API methods

Function	WebSphere MQ Workflow API signatures	WebSphere Process Server API signatures

Process creation and start	createAndStartInstance(...)
	createInstance(...) + start()
		sendMessage(...)
		initiate(...)
		call(...) // for microflow
		via a invocation task:
		createAndStartTask(...)
		createTask(...) + startTask(...)

Suspending a process instance	suspend()	suspend(...)
Resuming a process instance	resume()	resume(...)
Terminating a process instance	terminate()	forceTerminate(...)
Restart a process instance	restart()	restart(...)
Deleting process instance	delete()	delete(...)

WebSphere MQ Workflow

Figure 12-5 shows all possible process instance states as well as all possible state transitions, mainly forced by the life cycle action API calls.

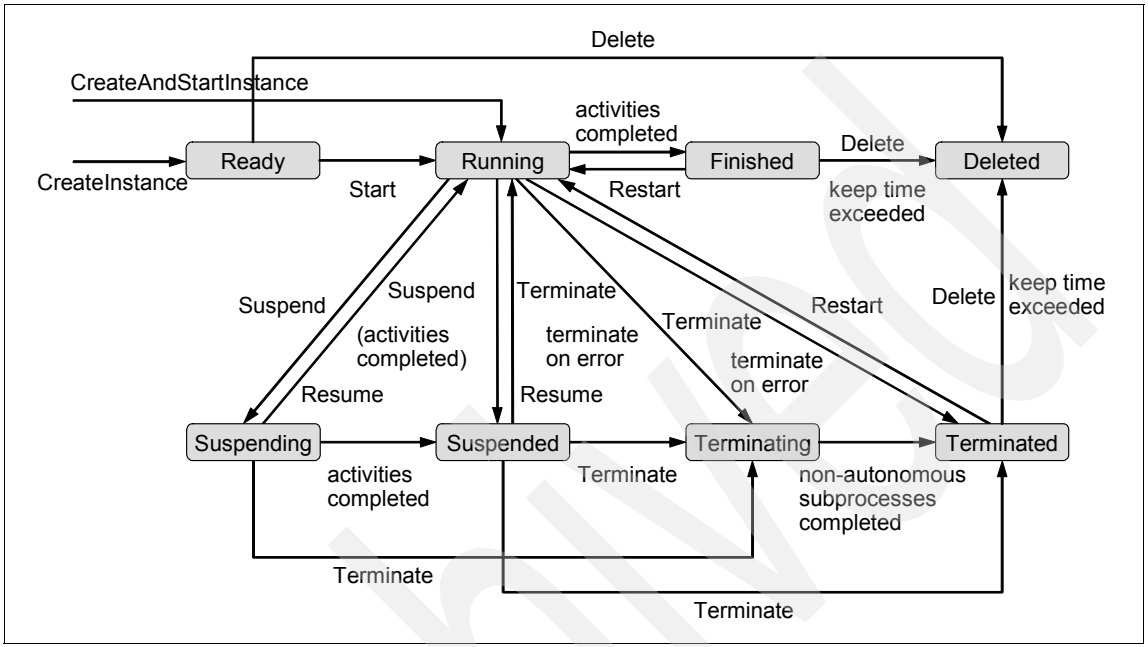


Figure 12-5 WebSphere MQ Workflow process instance state diagram

For more information about managing the life cycle of a business process see Chapter 45, “Process instance actions,” in the *IBM WebSphere MQ WorkflowProgramming Guide Version 3.6*.

WebSphere Process Server

Figure 12-6 shows all possible process instance states as well as all possible state transitions mainly forced by the life-cycle action API calls of a top level with a specified autonomy setting of *peer*.

Note: *Autonomy* is used with long-running processes. When one process is invoked by another, this setting determines its level of independence. There are two possible settings to choose from:

- Peer

  The invoked process is considered to be a peer of the primary process. In this case, both processes run concurrently and happen to have a point where they intersect.

- Child

  The invoked process is considered to be a child, and the primary process its parent. As such, the child is tied to the life-cycle and compensation sphere of the parent.

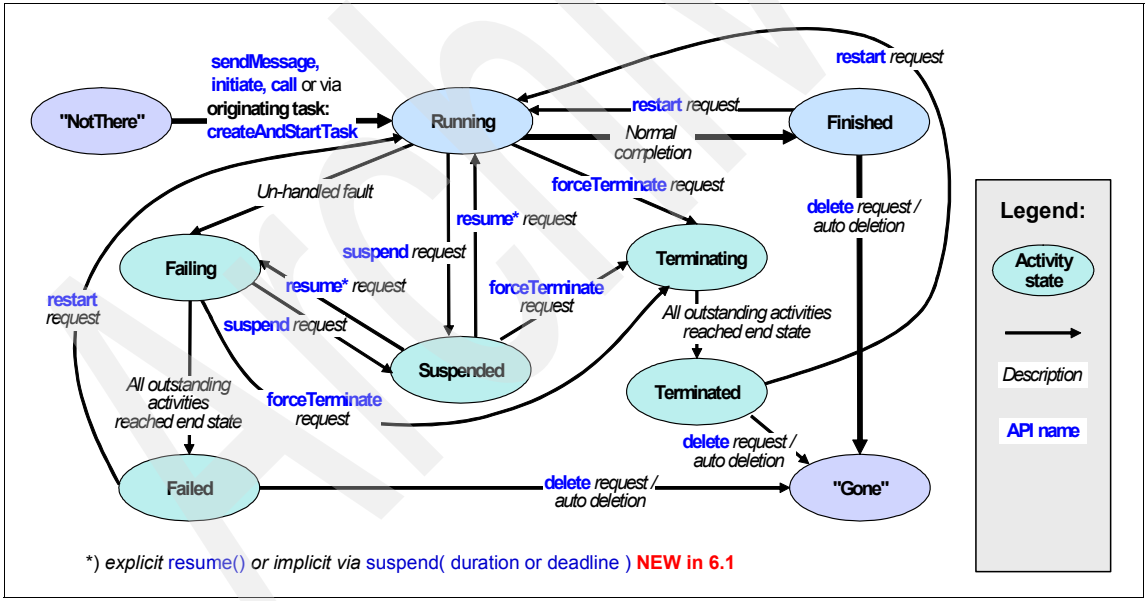


Figure 12-6 WebSphere Process Server process instance state diagram: top level process

The major differences to WebSphere MQ Workflow are the missing begin and end states (ready, deleted) and the additional error states (failing, failed) with the additional state transitions to recover from an un-handled fault.

Figure 12-7 shows a process instance state diagram of a child process. The state transitions are mainly triggered by the top-level process life cycle API calls.

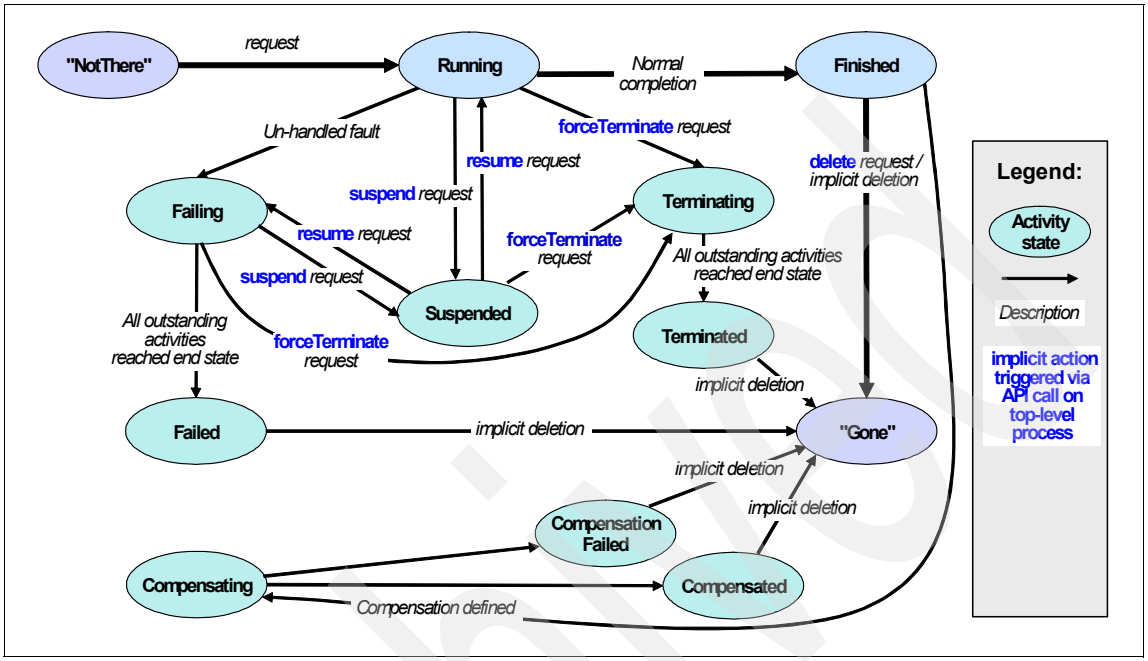


Figure 12-7 WebSphere Process Server process instance state diagram: Child process

For more information about managing the life cycle of a business process see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_lifecycle.html

Summary

WebSphere Process Server offers similar functionality compared to WebSphere MQ Workflow, but it is implemented differently due to the programming model.

12.2 Client implementation options: Activity concepts

A business process is typically built by using a set of activities.

WebSphere MQ Workflow

WebSphere MQ Workflow offers the following activity types:

- Program activity
  - Manually started (typically a human task)
  - Automatically started (typically a PES or UPES type activity)
- Block activity
- Process activity

WebSphere Process Server

WebSphere Process Server offers the following activity types:

- Basic activities:
  - Invoke activity
  - Assign activity
  - Receive activity
  - Reply activity
  - Wait activity
  - Empty action activity
  - Human task activity
  - Snippet activity.
- Structured activities:
  - Choice activity
  - Cyclic flow activity
  - For each activity
  - Parallel activity (BPEL flow activity)
  - Receive choice activity (BPEL pick activity)
  - Scope activity
  - Sequence activity
  - While loop activity.

WebSphere MQ Workflow, as well as WebSphere Process Server, allow for interactions with most of these activities via the application programming interface. The purpose of the interaction depends on the type, state, and other conditions of such an activity. An interaction could be an execution step, life cycle management call, monitoring of the activity state, exception handling, and so on.

This section covers and compares a WebSphere MQ Workflow automatic program activity and a WebSphere Process Server invoke activity and looks at API methods to locate (query) and to influence the behavior of an activity.

The human task activity is covered in 12.3, “Client implementation options: Task concepts” on page 301 to visualize the relationship and differences between a human task activity and a task itself.

12.2.1 Activity-related queries

This section discusses the query API methods to retrieve activity-related objects and properties of WebSphere MQ Workflow and WebSphere Process Server.

WebSphere MQ Workflow

WebSphere MQ Workflow provides for activity instance type specific query API methods, as shown in Example 12-9.

Example 12-9 WebSphere MQ Workflow Activity Instance query

```
ActivityInstance[] queryActivityInstances(
                     java.lang.String filter,
                     java.lang.String sortCriteria,
                     java.lang.Integer threshold )
```

WebSphere Process Server

WebSphere Process Server provides different methods. The most common ones, besides the persistent query() method, are shown in Example 12-10.

Example 12-10 WebSphere Process Server activity instance query

```
QueryResultSet query(java.lang.String selectClause,
                     java.lang.String whereClause,
                     java.lang.String orderByClause,
                     java.lang.Integer skipTuples,
                     java.lang.Integer threshold,
                     java.util.TimeZone timeZone)

QueryResultSet queryAll(java.lang.String selectClause,
                        java.lang.String whereClause,
                        java.lang.String orderByClause,
                        java.lang.Integer skipTuples,
                        java.lang.Integer threshold,
                        java.util.TimeZone timeZone)
```

The above-mentioned query()/queryAll() methods are not type-specific as are the WebSphere MQ Workflow queryActivityInstances(). With query() and queryAll() you can query business-process and task-related objects in the database to retrieve specific properties of these objects (refer to database views). To get a WebSphere MQ Workflow-like type-specific activity instance query result set you must specify it in the select-clause argument, as shown in Example 12-11.

Example 12-11 WebSphere Process Server Activity Instance query select clause

```
selectClause = "DISTINCT ACTIVITY.AIID"
```

You can use the queryAll() method to retrieve the data for all of the objects that are stored in the database, for example, for monitoring purposes. The queryAll() method matches the WebSphere MQ Workflow queryActivityInstances() better than the query() method. Note that the queryAll() method requires a specific J2EE authorization role (see the related Javadoc™).

For more general information about queries see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/c6bpel_query.html

12.2.2 Main activity state transitions

This section discusses the typical state behavior of an WebSphere MQ Workflow automatic program activity and WebSphere Process Server invoke activity and the related API methods to manage the regular work.

WebSphere MQ Workflow

During navigation through a business process an automatic program activity transitions from Inactive → Running state in case of a direct navigation to it. Otherwise the transition is from Inactive → Skipped. Further navigation now depends on the modeling construct *manual* or *automatic exit* and the exit condition of the activity. The API method of interface WorkItem finish() enables a client application to complete an activity with manual exit. The finished state of work items is visible if the activity is modeled as KEEP_WORKITEMS FOREVER or <time based>. The finished state of activities is visible independent of the KEEP_WORKITEMS setting. See Figure 12-8.

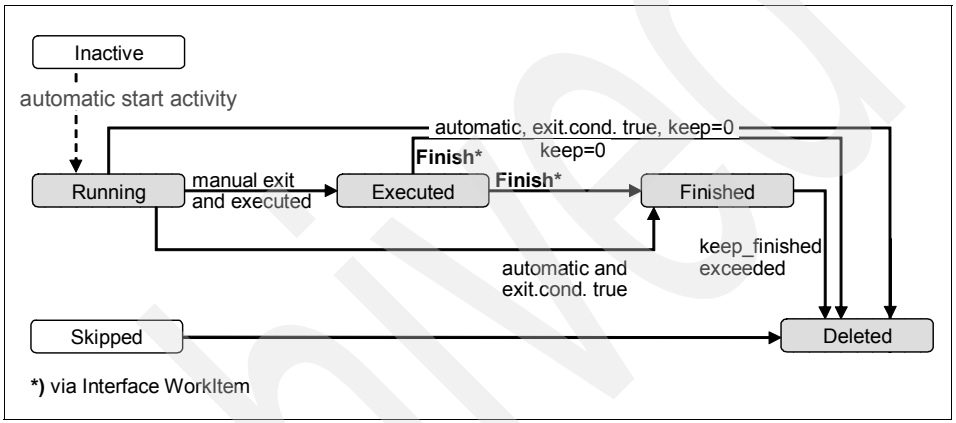


Figure 12-8 WebSphere MQ Workflow activity state diagram extract

WebSphere Process Server

An invoke activity, which is comparable to a WebSphere MQ Workflow automatic program activity, is similar in states and state transitions if we compare a typical life cycle. A manual exit like in WebSphere MQ Workflow does not exist, and therefore an executed state does not exist. Furthermore, the end state is not visible as a deleted state. See Figure 12-9.

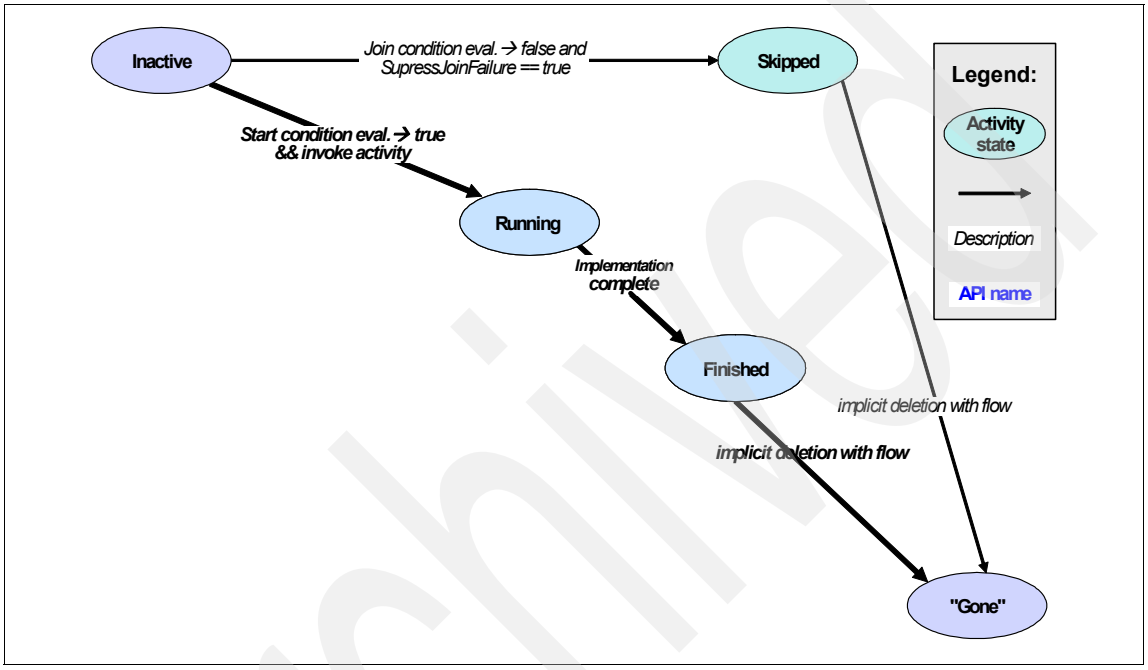


Figure 12-9 WebSphere Process Server activity instance state diagram extract for Invokes

12.2.3 Managing the life cycle of an activity

This section discusses the life cycle of a WebSphere MQ Workflow automatic program activity and a WebSphere Process Server invoke activity and the related API methods to manage the life cycle.

Comparing the full life-cycle possibilities of a WebSphere MQ Workflow automatic program activity with a WebSphere Process Server invoke activity is very difficult for various reasons:

- A WebSphere MQ Workflow automatic program activity can be combined with a manual human task to start and to exit the activity all within *one* activity. This would require three activities (a human task in front of an invoke, the invoke itself, and a human task as a subsequent activity of the invoke) in the WebSphere Process Server programming model.

- Some states and their corresponding state transitions do not exist in WebSphere Process Server:
  - Ready and executed state, due to the constraints described in the first bullet
  - End state deleted and the ForceFinished state, which is different from the finished state
  - InError state, which must be mapped to two more granular states (failed and stopped)
- The WebSphere MQ Workflow documentation does not offer an explicit activity state diagram, but a work item state diagram that covers the automatic activities from a work item perspective, as well as the manual activities (human tasks) in one figure. The work item perspective differs in the following states from an activity perspective:
  - Inactive state does not exist for work items.
  - Skipped state does not exist for work items.
  - Disabled state does not exist for activities.

  For more information refer to Chapter 51, “Work item actions,” in the *IBM WebSphere MQ Workflow Programming Guide Version 3.6*.

WebSphere MQ Workflow

Figure 12-10 shows a WebSphere MQ Workflow work item state diagram for a process instance in running state.

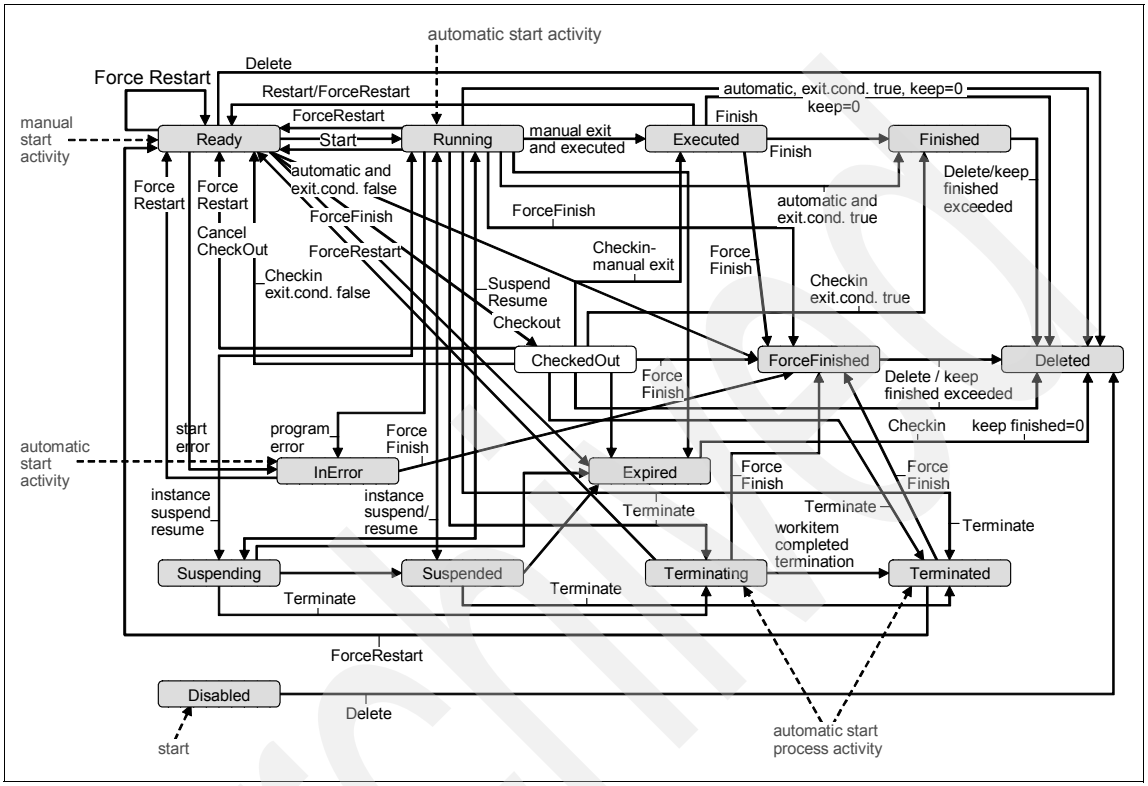


Figure 12-10 WebSphere MQ Workflow work item state diagram for process instance in running state

Figure 12-11 shows a WebSphere Process Server activity instance state diagram for invokes.

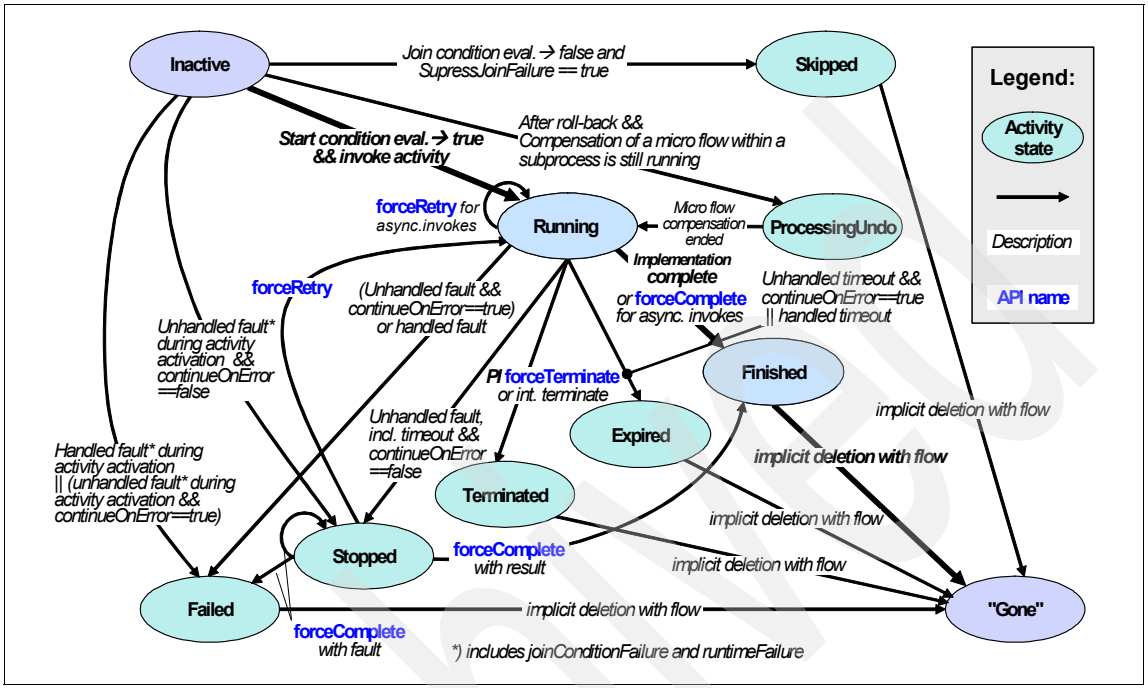


Figure 12-11 Activity instance state diagram for invokes

Summary

WebSphere Process Server offers similar functionality to WebSphere MQ Workflow, but implemented differently due to the programming model.

In addition to the state differences, function differences on an API method level can be observed. These are shown in Table 12-3.

Table 12-3 Activity Instance life cycle API methods of both products

Function	WebSphere MQ Workflow API signatures	WebSphere Process Server API signatures
Manual activity start	start()	NA Can be implemented with a human task as a predecessor activity

Function	WebSphere MQ Workflow API signatures	WebSphere Process Server API signatures
Manual activity finish	finish()	NA Can be implemented with a human task as a successor activity
Force retry	forceRestart() + start()	forceRetry()
Force complete	forceFinish()	forceComplete()
Activity terminate	terminate() of the process instance	forceTerminate() of the process instance
Delete	delete()	NA

Note: Most of the described APIs are available with different signatures. Some of them, especially the repair APIs, deal with data and have API arguments like *Container inputContainer* or *Container outputContainer* in WebSphere MQ Workflow. These API arguments would map to ClientObjectWrapper inputMessage or ClientObjectWrapper outputMessage in WebSphere Process Server.

12.2.4 Activity repair

WebSphere MQ Workflow automatic program activities, as well as WebSphere Process Server invoke activities, are subject to repair in the following cases:

- The invoked service knows that there is a problem and reports the error or fault to the workflow engine.
- The invoked service is not able to get back to the workflow engine. For example, the correlation information was lost due to a break-down.

Both cases can be handled by WebSphere MQ Workflow as well as by WebSphere Process Server.

WebSphere MQ Workflow

Case 1: The activity state changes from Running → InError. In this case the invoked service is able to feed back the error information. This allows the application programmer to retrieve and display the error reason via the API method FmcError errorReason() of Interface WorkItem. Use forceRestart() + start() or use forceFinish() depending on the error situation. Notice that forceRestart() + start() is always a two-step approach. During the repair the

activity state changes from InError → Ready → Running or from InError → ForceFinished.

Case 2: The activity state does not change because the invoked application does not return. Use forceRestart() + start() or use forceFinish() depending on the error situation. The repair API methods are the same as used in case 1. Due to the missing feedback from the invoked service the error reason cannot be returned and displayed. The possible state transitions are Running → Start or Running → ForceFinished

WebSphere Process Server

Case 1: The activity state depends on the modeling construct ContinueOnError. A continueOnError is a flag that indicates what should happen in case of an error, that is, when a system exception or an unhandled fault occurs. *True* states that navigation should continue (that is, the activity instance is set into execution state failed and the error is propagated to the associated process instance). *False* states that the activity instance should stay in execution state stopped.

To get the WebSphere MQ Workflow error behavior you must set ContinueOnError=false in the activity properties of the process model. If it is set to false, then the activity state transitions from Running → Stopped. Otherwise, it transitions from Running → Failed. The stopped state allows a repair via API methods, while the failed state does not. The forceRetry() API method can be used to repair (redo) the invoked service. Note that this is a one-step repair compared to the WebSphere MQ Workflow two-step repair (forceRestart() plus start()).

The repair alternative could be a forceComplete() API call. Forcing the completion of an activity instance states that navigation of the process instance can continue. An output message is passed to denote the successful execution of user processing. The activity instance is put into the finished execution state.

If user processing did not complete successfully, that is, if a fault message is passed, the activity instance is put into the failed execution state when the fault is handled or when the continueOnError flag is set to true. It remains in the stopped execution state when the fault is not handled and the continueOnError flag is set to false.

In this case the invoked service is able to feed back the exception information. This allows the application programmer to retrieve and display the fault reason via a fault message instead of an output message.

The following API method retrieves the fault message of an activity instance: ClientObjectWrapper getFaultMessage(...). The repair triggers state transitions from stopped → running or stopped → finished.

The forceRetry() as well as forceComplete() methods allow you to overwrite the modeled continueOnError settings.

Case 2: The activity state does not change because the invoked service does not return. Use forceRetry() or forceComplete() depending on the fault situation. The repair API methods are the same as used in case 1. Due to the missing feedback from the invoked service the fault reason cannot be returned and displayed. The possible state transitions are running → running or running → finished.

For more information refer to the following WebSphere Process Server V6.1 information center links:

- Repairing activities

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6activity_repair.html

- Handling exceptions and faults

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_errors.html

Note: From WebSphere Process Server Version 6.1.2 on, the continue-on-error behavior of activities and business processes have been enhanced. For more information refer to:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.612.doc/doc/bpc/cfaulthandling_continueonerror.html

12.3 Client implementation options: Task concepts

A human task is a unit of work that involves a person. Human tasks support people via the API when they interact with business processes. The interactions that can take place over the lifetime of a task depending on the type of task. Certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the human task.

WebSphere MQ Workflow and WebSphere Process Server offer different types of tasks:

- To-do task

  This is where a business process assigns a task to a person as something for that person to do.

- Administration task

  This type of task grants a person administrative powers such as the ability to suspend, terminate, restart, force-retry, or force-complete a business process.

- Invocation task

  This is when a person can invoke a business process.

The above-listed task type names are WebSphere Process Server-specific, but the represented functionality also exists in WebSphere MQ Workflow. Additional task types like collaboration task are not listed here because they are not supported by WebSphere MQ Workflow.

In WebSphere MQ Workflow the human tasks that are defined as part of a business process are called inline tasks. Inline tasks can be to-do tasks, invocation tasks, and administration tasks. WebSphere Process Server also offers the inline tasks and stand-alone tasks that follow the SOA pattern and are loosely coupled with the components that invoke them (to-do tasks) or the components that are invoked by them (invocation tasks).

The API methods, API arguments, and functional behavior depend on the task type and the task implementation (inline versus stand-alone).

The representation of human tasks in the programming model is different in WebSphere MQ Workflow than in WebSphere Process Server. A WebSphere MQ Workflow work item is similar to a WebSphere Process Server task. A WebSphere Process Server work item is more an authorization concept than a task. For more information refer to Chapter 4, “Planning for human interaction in business processes” on page 57.

This section discusses the API functionality of inline to-do tasks and some aspects of administration tasks and compares the WebSphere MQ Workflow API methods and the task behavior with that of WebSphere Process Server.

12.3.1 Tasks-related queries

To look for to-do tasks and to work with them, the query method to retrieve stored information about human tasks can be used. The query method returns objects according to the caller's authorization.

WebSphere MQ Workflow

WebSphere MQ Workflow provides for work item type-specific query API methods, as shown in Example 12-12.

Example 12-12 WebSphere MQ Workflow query API method for work items

```
WorkItem[] queryWorkItems(java.lang.String filter,
                          java.lang.String sortCriteria,
                          java.lang.Integer threshold)
```

The query returns an array of work items. These work items can directly be used for further processing, like check out (in WebSphere Process Server terms, claim).

WebSphere Process Server

WebSphere Process Server provides different methods. The most common one, besides the persistent query() method, is shown in Example 12-13.

Example 12-13 WebSphere Process Server query API method for tasks

```
QueryResultSet query(java.lang.String selectClause,
                     java.lang.String whereClause,
                     java.lang.String orderByClause,
                     java.lang.Integer skipTuples,
                     java.lang.Integer threshold,
                     java.util.TimeZone timeZone)
```

The above-mentioned query() is not type-specific as is the WebSphere MQ Workflow queryWorkItems() with an implicit select clause for work items. With query(), you can query business-process and task-related objects in the database to retrieve specific properties of these objects (refer to database views).

Another difference is that a work item in a Business Process Choreographer/WebSphere Process Server context has a different meaning than in WebSphere MQ Workflow. In WebSphere MQ Workflow, a work item is like a personalized copy of a manual activity with the necessary authorization rights. Work items can directly be used for further processing, like performing a check-out (In WebSphere Process Server terms, claim a to-do).

The definition of a work item in Business Process Choreographer is only a relationship between an entity (for example, task) a reason (for example, potential owner), and user (or group). A work item grants an authorization to perform work. An API action call (for example, claim()) on a work item itself, like in WebSphere MQ Workflow, is not possible. Due to this new programming

model, a select clause specification could contain, for example, a task instance ID and not a work item as in WebSphere MQ Workflow to use this information later on to perform a claim(), as shown in Example 12-14.

Example 12-14 WebSphere Process Server query API parameters for tasks

```
QueryResultSet result =
     task.query("TASK.TKIID",                                    // select clause
                "TASK.STATE = TASK.STATE.STATE_READY AND
                 TASK.KIND = TASK.KIND.KIND_PARTICIPATING AND
                 WORK_ITEM.REASON =
                   WORK_ITEM.REASON.REASON_POTENTIAL_OWNER", // where clause
                (String)null, (Integer)null, (TimeZone)null);
```

For more information see the following topics:

- Queries on business-process and task-related objects:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6query.html

- WebSphere Process Server 6: Business Process Choreographer query() and queryAll() - How to access processes, tasks and work items through the API and JDBC:

  http://www-1.ibm.com/support/docview.wss?rs=2307&uid=swg27010849

- Authorization and staff resolution in Business Process Choreographer: Part 1: Understanding the concepts and components of staff resolution:

  http://www.ibm.com/developerworks/websphere/techjournal/0710_lind/0710_lind.html

- Part 2: Understanding the programming model for staff resolution:

  http://www.ibm.com/developerworks/websphere/techjournal/0711_lind/0711_lind.html

12.3.2 Main task state navigation

In this section we discuss main task state navigation.

WebSphere MQ Workflow

A program activity with a manual start condition represents a human to-do task. When a business process navigates to such program activity that was started manually, the activity instance transitions from state inactive → ready in case of a direct navigation to it. Otherwise, the transition is from inactive → skipped.

Further navigation now depends on modeling and mainly on API actions on the associated work item. Figure 12-12 illustrates the typical main navigation steps of a manual starting (to-do task).

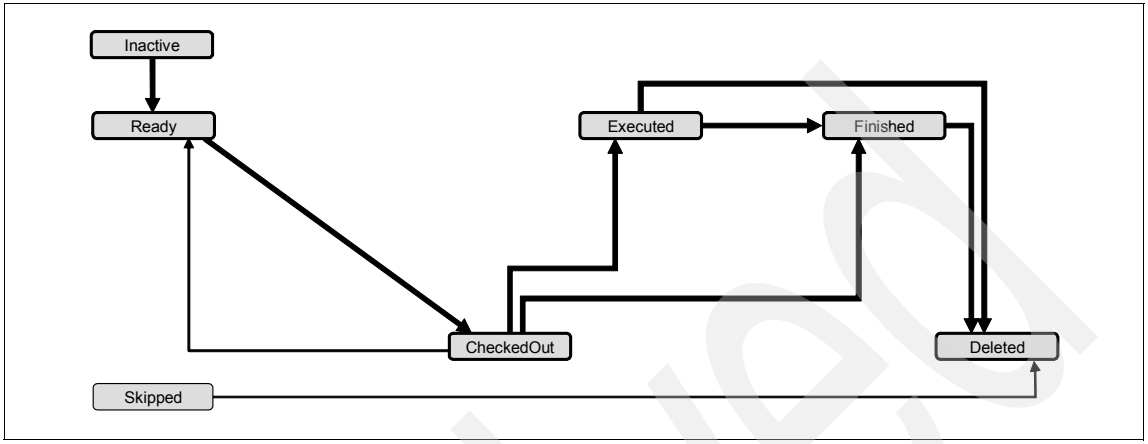


Figure 12-12 WebSphere MQ Workflow activity instance state diagram for a to-do task (extract)

The interface ActivityInstance does not offer API action calls for the activity instance state transitions illustrated above. All the API action calls on WebSphere MQ Workflow to-do tasks are done via the WorkItem interface or item interface.

Figure 12-13 shows the work item state diagram with the related API action methods of the WorkItem interface.

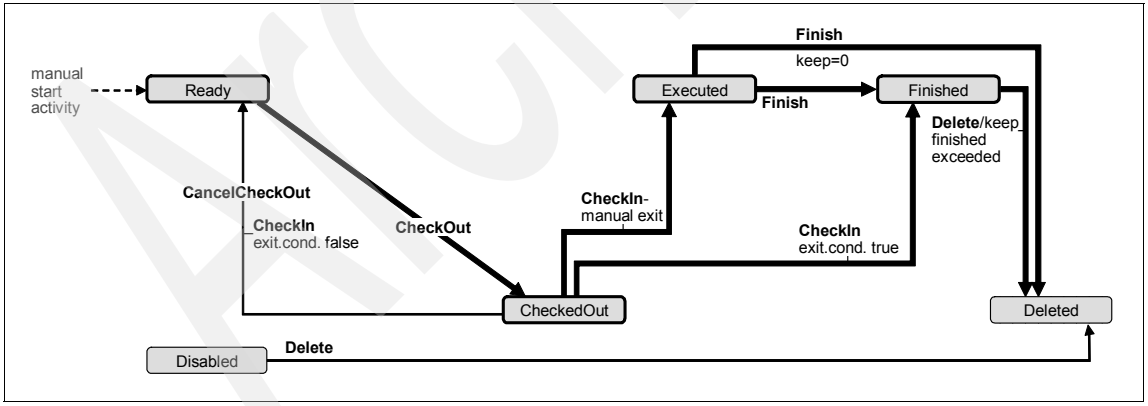


Figure 12-13 WebSphere MQ Workflow work item state diagram for a to-do task (extract)

The main action API method calls are checkOut(), then work on the to-do task, and checkIn(). The state transition is ready → CheckedOut → deleted. Further transitions now depend on the modeling constructs of a to-do task, such as:

- Manual exit.
- Exit condition.
- Keep work items.

WebSphere Process Server

WebSphere Process Server provides different flavors of to-do tasks. Here the focus is on the inline to-do task, which is closely related to a WebSphere MQ Workflow to-do task (manual starting program activity).

The Business Process Choreographer of WebSphere Process Server has, similar to WebSphere MQ Workflow, different views on to-do tasks. It is an activity instance and a task (instead of a work item) view. See Figure 12-14.

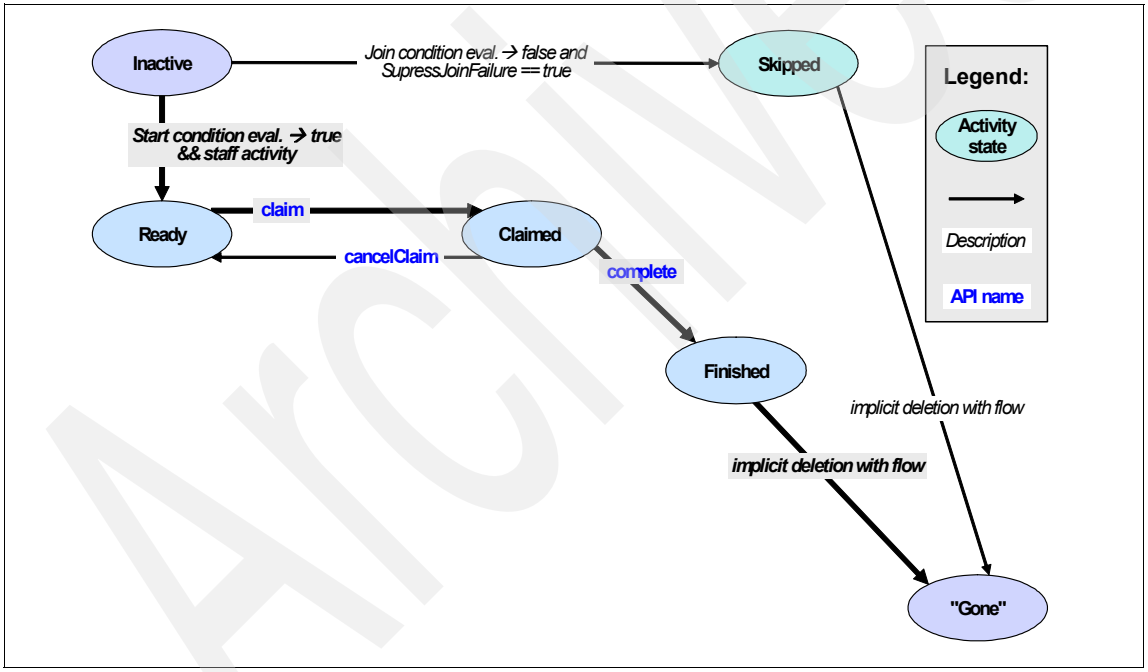


Figure 12-14 WebSphere Process Server task activity state diagram extract for a to-do task

The main action API method calls are claim(), then work on the to-do task and complete() it on the interface BusinessFlowManager. The state transitions are inactive → ready → claimed → finished in case of a direct navigation to it. Otherwise the transition is from inactive → skipped.

The main API action methods, claim() and complete(), are also available on the Interface HumanTaskManager. The state transitions are ready → claimed → finished. An activity state skipped does not exist on this interface. Figure 12-15 shows an extract of an inline to-do task state diagram.

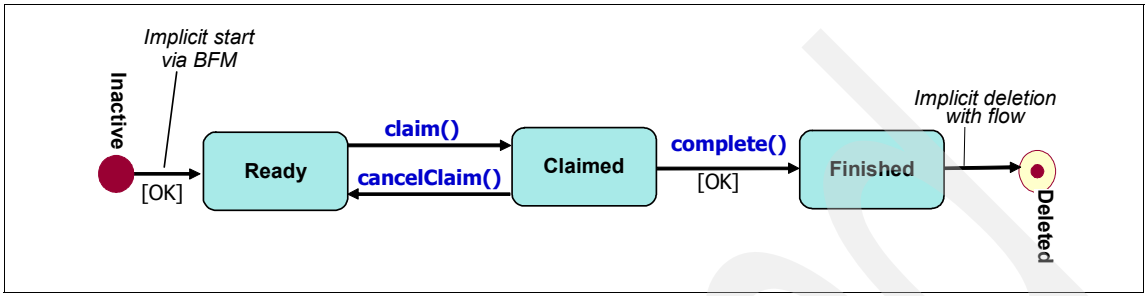


Figure 12-15 WebSphere Process Server inline to-do task state diagram extract

An alternative to an inline to-do task is a stand-alone to-do task. Figure 12-16 demonstrates the differences in programming model as well as in the naming, which differs in WebSphere Process Server 6.0.x and 6.1. These different technical terms are also important to know when specifying the select, where, and order-by clauses of queries.

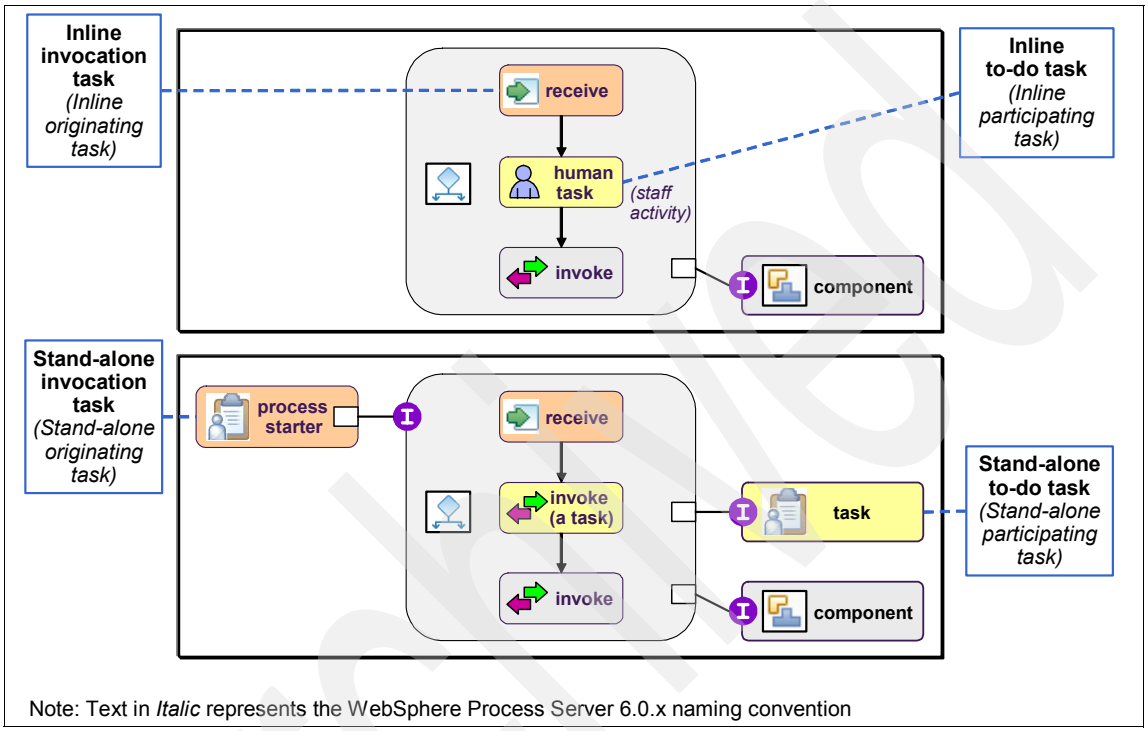


Figure 12-16 WebSphere Process Server inline versus stand-alone tasks

If a WebSphere MQ Workflow to-do task (always inline to the process) is transitioned to a stand-alone task in WebSphere Process Server as an alternative to an inline task, then we must look at the following figures to understand the implementation, the API action methods, and the state transitions.

A stand-alone to-do task is implemented as invoke activity on a process level. This invocation in the wiring diagram of WebSphere Integration Developer points to a stand-alone do-do task as a service implementation. Due to this different programming model the available API action methods as well as the state and the transition behavior are different from the inline to-do task version. See Figure 12-17.

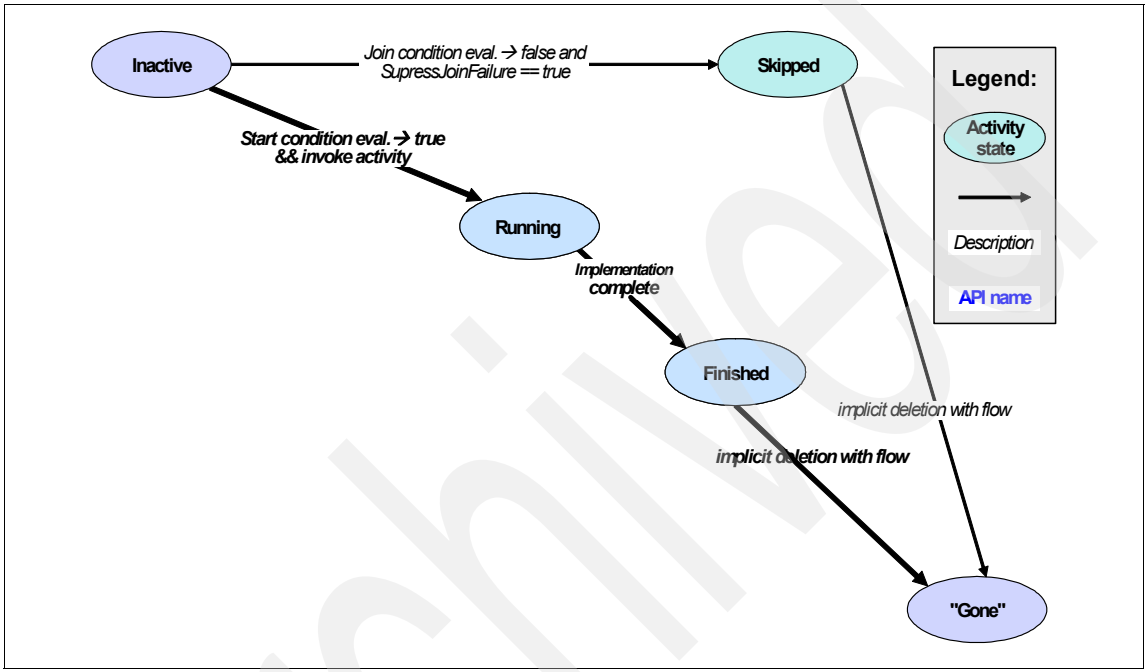


Figure 12-17 WebSphere Process Server invoke activity state diagram (extract) for stand-alone to-do task

The main action API method calls are claim(), then work on the to-do task and complete() it on the Interface BusinessFlowManager.

The state transitions are inactive → running → finished in case of a direct navigation to it. Otherwise the transition is from inactive → skipped. API action methods are not needed for the normal processing of this stand-alone task-related invocation activity.

The main API action methods, claim() and complete(), are only available on the Interface HumanTaskManager, as shown in Figure 12-18. The state transitions are inactive → ready → claimed → finished. An activity state skipped does not exist on this interface.

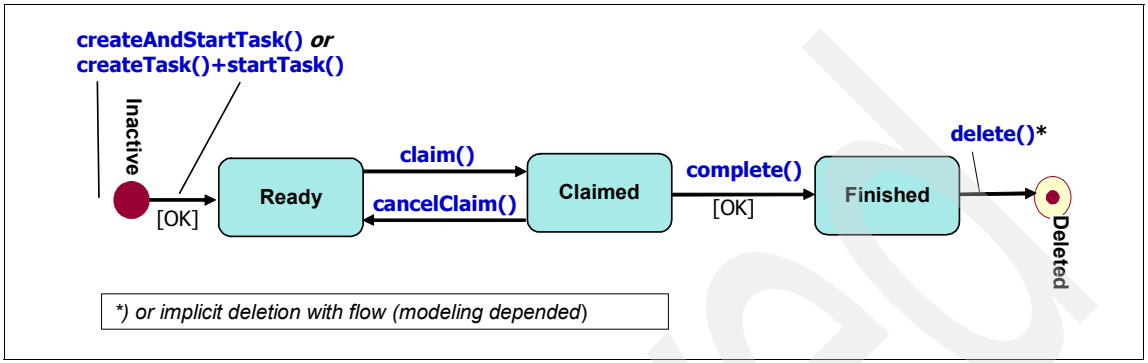


Figure 12-18 Stand-alone to-do task state diagram extract

Summary

The the normal processing of a to-do task in WebSphere MQ Workflow and in WebSphere Process Server is similar:

- Manual starting program activity maps to an inline to-do task.
- Check out of a work item maps to claim of a to-do task.
- Check in of a work item maps to complete of a to-do task.

Functional differences are due to modeling constructs (manual exit, exit condition, keep work items).

Implementation differences are due to different programming models.

Enhancements on the WebSphere Process Server side are the stand-alone to-do tasks as an alternative to the inline to-do tasks. Stand-alone to-do tasks are implemented as a service in the SCA programming model. This allows the modeler to very easily replace that SCA-based human service, for instance with an automatic service implementation, without changing the process model.

12.3.3 Managing the life cycle of a to-do task

In this section we discuss managing the life cycle of a to-do task.

WebSphere MQ Workflow

The complete life cycle of a program activity with a manual start condition (to-do task) is illustrated in Figure 12-19. Note, however, that the diagram also contains the behavior of a program activity with an automatic start condition, which does not make it easy to read. For more details refer to Chapter 51, “Work item actions,” in the *IBM WebSphere MQ Workflow Programming Guide Version 3.6*.

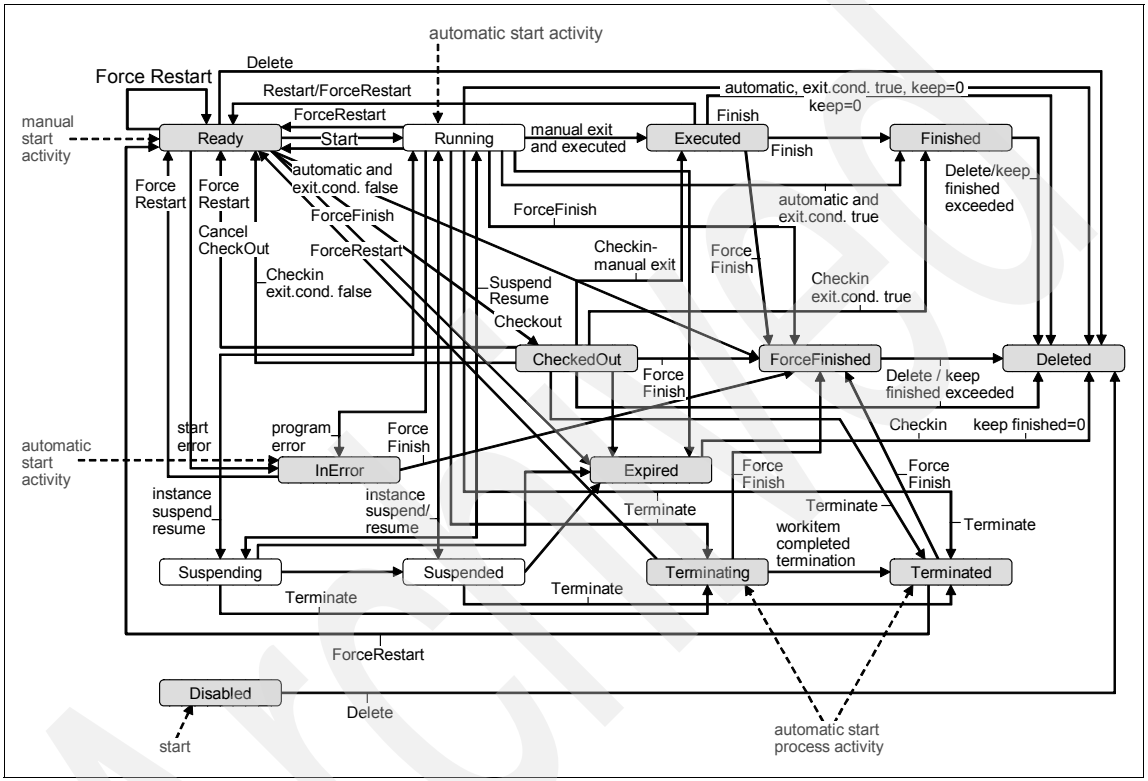


Figure 12-19 WebSphere MQ Workflow work item state diagram

WebSphere Process Server

The complete life cycle of an inline to-do task seen from the Business Flow Manager interface is illustrated in Figure 12-20. For more details refer to the WebSphere Process Server V6.1 information center:

- Processing human task activities

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_staff.html

- Processing a single person workflow

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_singleworkflow.html

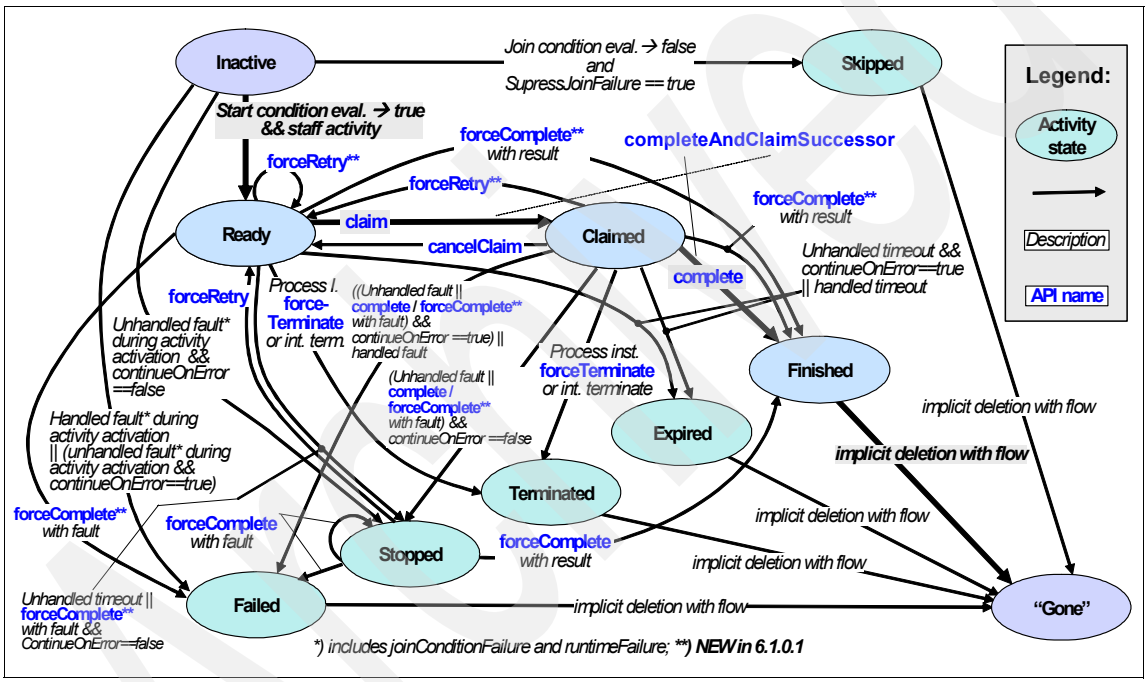


Figure 12-20 WebSphere Process Server Task activity state diagram

An application programmer has two possible interfaces available to program an application for human task interactions. Figure 12-20 shows the possibilities via the Business Flow Manager (BFM) interface (Package com.ibm.bpe.api).

Figure 12-21 shows the relationship between an inline task activity of the Business Flow Manager (BFM) programming model and the inline to-do task of the Human Task Manager (HTM) programming model. In addition, Figure 12-21 helps to get an understanding of how the states and the transitions are mapped between the mentioned programming models.

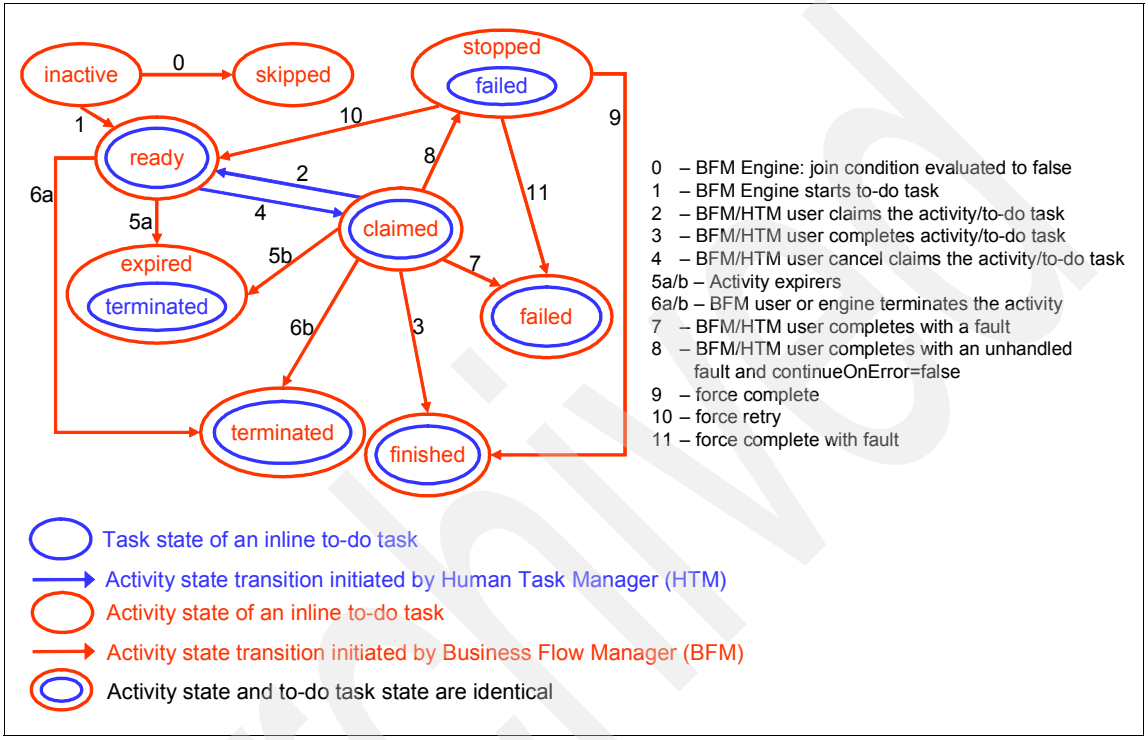


Figure 12-21 WebSphere Process Server inline task activity versus inline to-do task state and transitions

The complete life cycle of an inline to-do task seen from the Human Task Manager interface is illustrated in Figure 12-22. For more details refer to the WebSphere Process Server V6.1 information center:

- Developing applications for human tasks

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6tasks.html

- Javadoc of Package com.ibm.task.api

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/task/api/package-summary.html

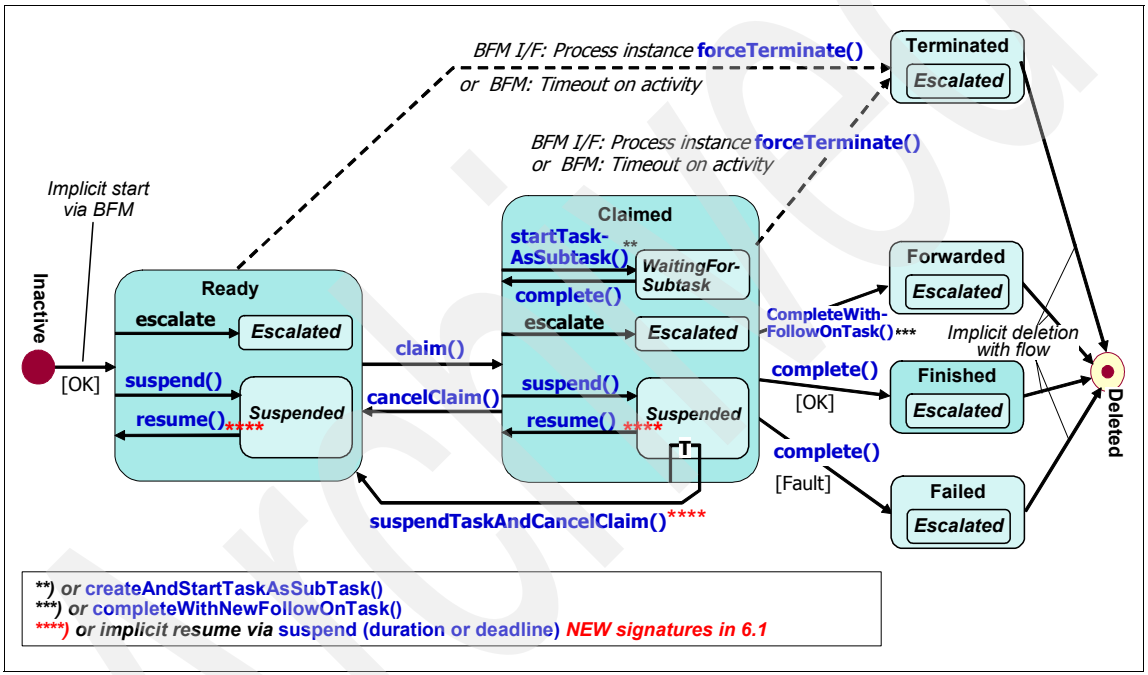


Figure 12-22 WebSphere Process Server inline to-do task state diagram

Summary

WebSphere Process Server offers similar functionality to WebSphere MQ Workflow, but implemented differently due to the programming model. In some areas, enhanced functions are available (for example, to suspend a task). An application architect or programmer must decide which interfaces best fulfill the requirements.

The function differences on an API method level are shown in Table 12-4.

Table 12-4 Functional comparison of inline do-do tasks life cycle API methods

Functions	WebSphere MQ Workflow workitem interface	WebSphere Process Server BFM interface	WebSphere Process Server HTM interface
Claim.	checkOut()	claim()	claim()
Cancel claim.	cancelCheckOut()	cancelClaim()	cancelClaim()
Complete.	checkIn()	complete()	complete()
Complete and claim successor.	Not available	completeAndClaimSuccessor()	Not available
Suspend/resume.	Not available	Not available	suspend()/resume()
Complete with follow-on task.	Not available	Not available	completeWithFollowOnTask()
Start as subtask.	Not available	Not available	startAsSubtask()
Restart.	restart()	Not available	Not available
Force retry.	forceRestart()	forceRetry(), but currently not for Ready state	Not available
Force complete.	forceFinish()	forceComplete()	Not available
Force terminate.	terminate()	forceTerminate()	Not available
Manual finish.	finish() Depends on the modeling option manual exit	Not available	Not available
Delete.	Yes, but depends on modeling	Not available Only implicit delete	Not available Only implicit delete

For a detailed functional description of these API methods, refer to the following publications:

- WebSphere MQ Workflow work item interface

  Chapter 51, “Work item actions,” in the *IBM WebSphere MQ Workflow API Programming Guide, Version 3.6*

  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SH12-6291-10

- WebSphere Process Server Business Flow Manager interface

  com.ibm.bpe.api interface BusinessFlowManagerService:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/package-summary.html

- WebSphere Process Server Human Task Manager interface

  com.ibm.task.api interface HumanTaskManagerService:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/task/api/HumanTaskManagerService.html

12.3.4 To-do task repair

If the repair of an inline to-do task is needed, use the API methods offered by the Business Flow Manager interface:

- forceRetry()
- forceComplete()
- forceFinish()

For more information refer to repairing activities at:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6activity_repair.html

12.3.5 Delegation of work concepts

Delegation of work from a client point of view is typically done by a transfer of the ownership. This concept is implemented in WebSphere MQ Workflow as well as in WebSphere Process Server. The needed authorizations as well as the transfer rules partly differ due to the different programming models. Table 12-5 provides an overview over the delegation of work concepts.

Table 12-5 Delegation of work concept overview

Function/topic	WebSphere MQ Workflow API signatures/authorization/rules	WebSphere Process Server API signatures/authorization/rules
Transfer a work item API method.	transfer(java.lang.String userID).	transferWorkItem(TKIID tkiid, int assignmentReason, java.lang.String fromOwner, java.lang.String toOwner) Additional methods with other signatures are available.
Transfer work item authorization.	The user who transfers the item must be the owner of the item or have work item authorization for the owner of the item and have work item authorization for the new owner.	The caller must be an owner, starter, originator, or administrator of the task instance.

Function/topic	WebSphere MQ Workflow API signatures/authorization/rules	WebSphere Process Server API signatures/authorization/rules
Transfer work item rules.	Notifications can always be transferred. A work item must be in states ready, InError, executed, suspending, suspended, or terminated and the associated process instance in states running, suspending, suspended, or terminating. Work items in states InError or terminated can only be transferred to the process administrator. A work item implemented by a process can also be transferred if it is in the running state.	The task can be escalated. suspended, or waiting for subtasks. The following specific rules apply for the transfer of work items: ► Work items assigned to *everybody* cannot be transferred. ► The owner of a task instance can transfer the owner work item to a potential owner or an administrator of the task instance. ► The starter of a task instance can transfer the starter work item to a potential starter or an administrator of the task instance. A starter work item can be transferred in the expired, terminated, finished, failed, and running state. ► The originator of a task instance can transfer the originator work item to a potential instance creator or an administrator of the task instance. ► The originator of a task instance can transfer a potential starter work item to any person. A potential starter work item can be transferred in the inactive state. ► The administrator of a task instance can transfer all work items to any person. A starter work item can be transferred in the expired, terminated, finished, failed, and running state. A potential starter work item can be transferred in the inactive state. A reader or administrator work item can be transferred in all but the inactive state. A potential owner or editor work item can be transferred in the ready or claimed state.
Check the latest product documentation for updates regarding authorization and rules.		

WebSphere Process Server offers additional enhanced capabilities based on additional APIs, for example:

- Bulk transfer work (transferWorkItem(TKIID[ ] tkiid, ...))
- Creation of additional work authorizations (createWorkItem())

Figure 12-23 shows some of the new capabilities in combination with a common capability.

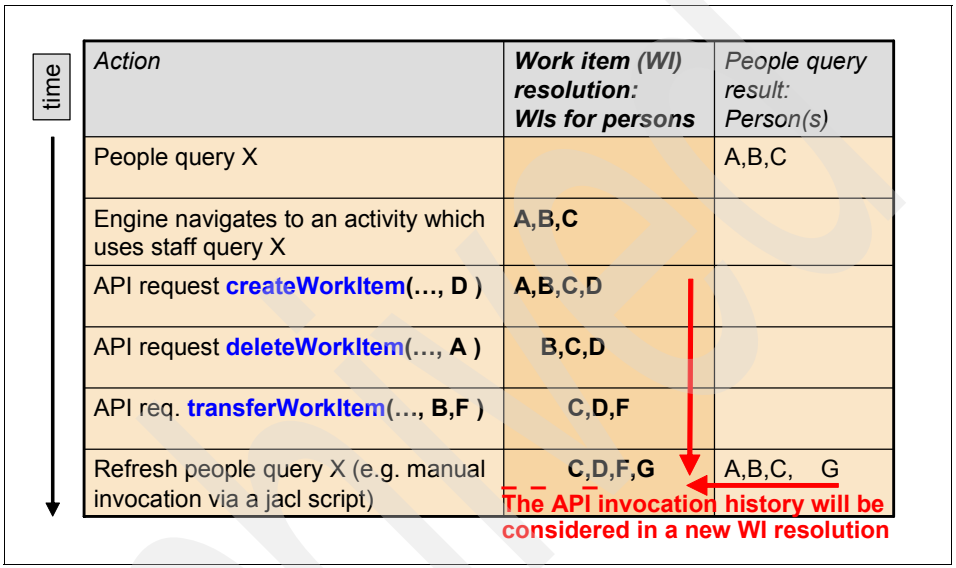


Figure 12-23 WebSphere Process Server delegation of work example

12.4 Data handling

Table 12-6 on page 320 provides an overview of the data handling in WebSphere MQ Workflow and WebSphere Process Server.

Table 12-6 Data handling

	WebSphere MQ Workflow	WebSphere Process Server
Data definition	In MQ Workflow, data structure definitions describe the contents of the input and output data containers of: ► Processes ► Activities ► Blocks Or it describes a global data container of a process. Any data that is used as input or output, or referred to in exit or transition conditions, must be described in a data structure definition. Each data structure consists of members. For example, a data structure used to define an address might have members for the street name and the city name. The data type of a data structure member can be one of the basic MQ Workflow data types: ► String ► Long ► Floating point ► Binary Or it can refer to another, previously defined data structure. A data structure that refers to another data structure is called a nested data structure.	Variables hold the data that constitutes the state of a business process. Data in BPEL is written to, and read from, typed variables. Variables are typed using: ► WSDL message types ► XML schema types ► XML schema elements Variables are either: ► Messages typed, also known as interface variable: This variable stores either the input, output, or fault parameter for a particular operation of an interface. ► Data typed: This variable points to a business object.Technically, this can be an XSD Type or an XSD Element. The BPEL implementation allows you to specify simple data and business object variable types. XPath is the default language for manipulating and querying variables. Variables are declared by specifying a name, a WSDL message type, an XML schema type, or an XML schema element. A variable belongs to the scope in which it is declared. If it is created in the global process scope then it is a global variable, and thus visible to the process as a whole. Those that are created within nested scopes are called scoped or local variables, and can only be seen by objects within the scope in which it was created.

	WebSphere MQ Workflow	WebSphere Process Server
Mapping basic FDL data types to XML Schema type	► STRING ► LONG ► FLOAT ► BINARY	► xsd:string ► xsd:long ► xsd:double ► xsd:base64Binary
	For more information see also SupportPac WA73: FDL2BPEL Conversion Tool (documentation): http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en	
Data usage in process models	► Input container ► Output container ► Global container Error related information is handled as: ► Return code as predefined data member _RC or as user-defined data member as part of the output container for the API-based interface ► Exception element of an ActivityImplResponse XML message ► GeneralError XML message	► Input message ► Output message (See the related information about query properties.) ► Fault message

	WebSphere MQ Workflow	WebSphere Process Server

Data retrieval API methods	com.ibm.workflow.api
	// Interface ProcessTemplate
	ReadWriteContainer initialInContainer()
	
	// Interface ProcessInstance
	ReadWriteContainer inContainer()
	ReadOnlyContainer outContainer()
	ReadOnlyContainer globalContainer()
	
	// Interface ActivityInstance
	ReadOnlyContainer inContainer()
	ReadOnlyContainer outContainer()
	
	// Interface ProgramData
	ReadOnlyContainer inContainer()
	ReadWriteContainer outContainer()
	
	// Interface WorkItem
	ReadOnlyContainer checkOut()
	ReadOnlyContainer inContainer()
	ReadWriteContainer outContainer()
	
	// Interface ProgramTemplate
	ReadWriteContainer execute()
	ReadWriteContainer execute2()
	ReadWriteContainer initialInContainer()
	ReadWriteContainer initialOutContainer()
		// Interface BusinessFlowManager
		ClientObjectWrapper createMessage()
		ClientObjectWrapper call()
		ClientObjectWrapper claim()
		ClientObjectWrapper getInputMessage()
		ClientObjectWrapper getOutputMessage()
		ClientObjectWrapper getFaultMessage()
		ClientObjectWrapper getVariable()
		
		// Interface HumanTaskManager
		ClientObjectWrapper callTask()
		ClientObjectWrapper claim()
		ClientObjectWrapper createAndCallTask()
		ClientObjectWrapper createFaultMessage()
		ClientObjectWrapper createInputMessage()
		ClientObjectWrapper createOutputMessage()
		ClientObjectWrapper getFaultMessage()
		ClientObjectWrapper getInputMessage()
		ClientObjectWrapper getOutputMessage()

Data access API classes and interfaces	com.ibm.workflow.api Interface Container Interface ReadOnlyContainer Interface ReadWriteContainer Interface ContainerElement	com.ibm.bpe.api Class ClientObjectWrapper com.ibm.task.api Class ClientObjectWrapper Package commonj.sdo

WebSphere Process Server

To illustrate the data retrieval and a simple data access in WebSphere Process Server, see the code snippet in Example 12-15.

Example 12-15 WebSphere Process Server data handling code example

```
//create a message for unique starting service of a process
ClientObjectWrapper input = process.createMessage(
                                template.getID(),
                                template.getInputMessageTypeName() );
DataObject myMessage = null; // Package commonj.sdo
if ( input.getObject() != null &&
```

```
    input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  // set the strings in the message, for example, a customer name
  myMessage. setString( "CustomerName", "Smith");
}
```

Summary

In addition to the WebSphere MQ Workflow data handling functionality, WebSphere Process Server offers additional capabilities concerning the following:

- The definition of data types (not limited to string, long, floating point, binary)
- Data manipulation API methods, for instance, a copy of a business object.

12.5 Queries

The application programming interface provides methods to query business-process, activity, and task-related objects in the database to retrieve specific properties of these objects. WebSphere MQ Workflow, as well as WebSphere Process Server, offer different conceptual query implementations.

You can perform a one-off (ad-hoc) query to retrieve a specific property of an object. You can also save queries (persistent/stored query) that you use often and include these stored queries in your application. The stored query can be either a query that is available to all users (public query) or a query that belongs to a specific user (private query). Table 12-7 provides an overview of the API methods. More details about stored queries are provided in the next section.

Table 12-7 .Query concepts and query API methods overview

Query concept	WebSphere MQ Workflow	WebSphere Process Server
Ad hoc	queryProcessTemplates() queryProcessTemplatesResultSize()	queryProcessTemplates() Not available
	not available	queryTaskTemplates()
	queryProcessInstances() queryProcessInstancesResultSize()	query() For query count results use the COUNT argument in a selectClause.
	queryActivityInstanceNotifications() queryActivityInstanceNotificationsResultSize() queryItems() queryItemsResultSize() queryProcessInstanceNotifications() queryProcessInstanceNotificationsResultSize() queryWorkItems()	
	queryActivityInstances() queryActivityInstancesResultSize()	queryAll() For query count results use the COUNT argument in a selectClause. In addition, be aware of the following additional API methods: ► getAllActivities() ► getAllWorkItems()

Query concept		WebSphere MQ Workflow	WebSphere Process Server
Persistent	Public	createProcessTemplateList() queryProcessTemplates() queryProcessTemplatesResultSize()	Not available as persistent query. Use ad-hoc version of queryProcessTemplates() instead. Not available.
		createProcessInstanceList() queryProcessInstances() queryProcessInstancesResultSize()	createStoredQuery() query() For query count results use the COUNT argument in a selectClause.
		createWorkList() queryActivityInstanceNotifications() queryActivityInstanceNotificationsResultSize() queryItems() queryItemsResultSize() queryProcessInstanceNotifications() queryProcessInstanceNotificationsResultSize() queryWorkItems()	
		createActivityInstanceList() queryActivityInstances() queryActivityInstancesResultSize()	Not available as persistent query. Use ad-hoc version of queryAll() instead.
	Private	createProcessTemplateList() queryProcessTemplates() queryProcessTemplatesResultSize()	Not available as persistent query. Use ad-hoc version of queryProcessTemplates() instead. Not available
		createProcessInstanceList() queryProcessInstances() queryProcessInstancesResultSize()	createStoredQuery() query() For query count results use the COUNT argument in a selectClause.
		createWorkList() queryActivityInstanceNotifications() queryActivityInstanceNotificationsResultSize() queryItems() queryItemsResultSize() queryProcessInstanceNotifications() queryProcessInstanceNotificationsResultSize() queryWorkItems()	
		createActivityInstanceList() queryActivityInstances() queryActivityInstancesResultSize()	Not available as persistent query. Use ad-hoc version of queryAll() instead.

Table 12-7 on page 324 also provides a high-level mapping to WebSphere Process Server (Business Process Choreographer) query API methods. Use the listed API method names as a pointer into the documentation (Javadoc as well as concepts documentation) to look for further details and additional enhanced functional capabilities.

One of the enhanced capabilities is the skip tuples concept. The skipTuples parameter specifies the number of records from the beginning of a result set that are skipped and are not returned in the query result set.

This parameter can be used together with the threshold parameter to implement paging. Some WebSphere MQ Workflow customers were forced to implement this behavior in the client application. It is now available in WebSphere Process Server with the regular application programming interface without circumventions.

12.5.1 Stored query

To understand the stored query concept in WebSphere MQ Workflow and in WebSphere Process Server one needs to clarify the different terminology. In WebSphere MQ Workflow it is called persistent list. In WebSphere Process Server it is called stored query.

The implementation concept in both products is similar, but the available application programming interface classes, interfaces, methods, and method arguments have different names and signatures. Table 12-8 provides an overview of the available methods and functions.

Table 12-8 Stored query application programming interface method overview

Description	WebSphere MQ Workflow API method	WebSphere Process Server API method
Create a query definition and persistently store it in the database for general usage.	**createProcessInstanceList**(name, type, owner, description, filter, sortCriteria, threshold) **Note:** Additional API methods are available, for example, for another object type like a work item.	**createStoredQuery**(storedQueryName, selectClause, whereClause, orderByClause, threshold, timeZone, storedQueryProperties, clientType) **Note:** Additional signatures are available, for example, for a private query.

Description	WebSphere MQ Workflow API method	WebSphere Process Server API method
List the available stored query object or query names.	**queryProcessInstanceLists()** **Note:** Additional API methods are available, for example, for another object type like a work item.	**getStoredQueryNames**(kind) **Note:** Additional signatures are available.
Perform a persistent query and retrieve.	queryProcessInstances() **Note:** Additional API methods are available, for example, for another object type like a work item.	**query**(userID, **storedQueryName**, skipTuples, threshold, parameters) **Note:** Additional signatures are available.
Count the number of potentially qualifying tuples in the database for a given query filter independent of a query threshold size.	**queryProcessInstancesResultSize**() **Note:** Additional API methods are available, for example, for another object type like a work item.	**query**(userID, **storedQueryName**, skipTuples, threshold, parameters) **Prerequisite: Use the COUNT keyword, when specifying a where-clause.** **Note:** Additional signatures are available.
Get access to the details of a stored query.	description() filter() isEmpty() name() ownerOfList() persistentOid() refresh() sortCriteria() threshold() type()	getStoredQuery getClientType() getCreator() getKind() getName() getOrderByClause() getOwner() getSelectClause() getStoredQueryProperties() getThreshold() getWhereClause()
Set the details of a stored query.	setDescription(description) setFilter(newFilter) setSortCriteria(sortCriteria) setThreshold(threshold)	Not available. Use deleteStoredQuery(...) and then recreate it with createStoredQuery() and new parameters.
Delete a stored query.	delete()	deleteStoredQuery(...)

When transferring a query application from WebSphere MQ Workflow to WebSphere Process Server, it is necessary to also consider the different authorization concepts of both products and their influences on the result set of the query API methods.

Further consideration should be given to the filtering and sorting aspects of queries. These aspects are described in the following section.

12.5.2 Filter and sort criteria language and specification

A query application programming interface method typically returns all objects it is authorized for. The number of objects returned to the client can be restricted by a filter and a threshold. The order within the returned result set can be specified via sort criteria.

WebSphere MQ Workflow filter criteria map to WebSphere Process Server where clause specifications.

WebSphere MQ Workflow sort criteria map to WebSphere Process Server order-by clause specifications.

WebSphere MQ Workflow

WebSphere MQ Workflow offers syntax diagrams to specify the filter and the sort criteria for a query application programming interface method. A filter and a sort specification is a character string specifying criteria that must follow the rules stated by the filter/sort syntax diagrams.

Figure 12-24 shows a process instance filter syntax diagram extract.

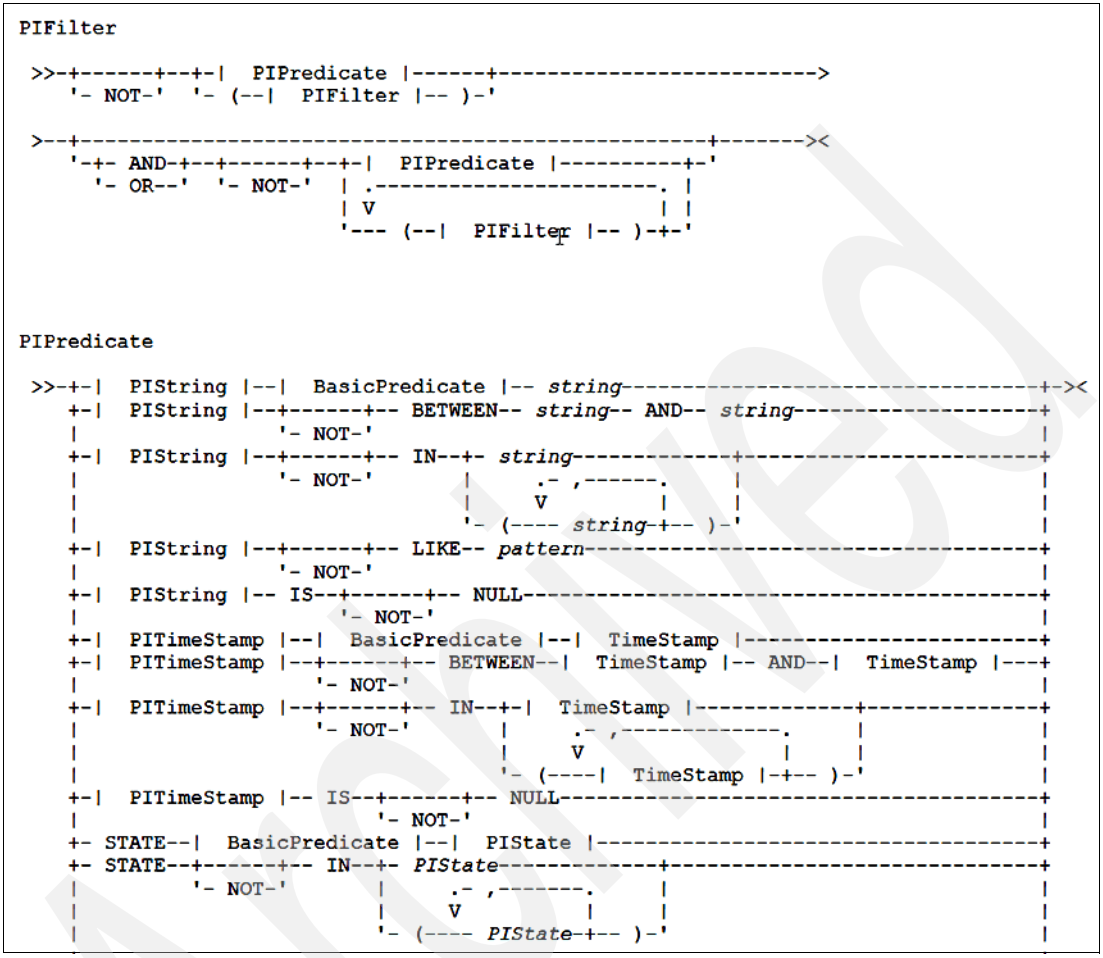


Figure 12-24 Query process instances filter syntax extract

WebSphere Process Server

WebSphere Process Server provides predefined database views for Business Flow Manager and Human Task Manager API-based queries. Use these views when you query reference data for business-process and human-task objects.

When you use the predefined views, you do not need to explicitly add join predicates for view columns. These constructs are added automatically for you.

You can use the generic query function of the service API (BusinessFlowManagerService or HumanTaskManagerService) to query this data. You can also use the corresponding method of the

HumanTaskManagerDelegate API or your predefined queries provided by your implementations of the ExecutableQuery interface. For example, use the PROCESS_INSTANCE view as a Business Process Choreographer database view. This predefined database view can be used for queries on process instances.

Table 12-9 shows the columns in the PROCESS_INSTANCE view.

Table 12-9 Columns in the PROCESS_INSTANCE view

Column name	Type	Comments
PTID	ID	The process template ID.
PIID	ID	The process instance ID.
NAME	String	The name of the process instance.
STATE	Integer	The state of the process instance. Possible values are: ► STATE_READY (1) ► STATE_RUNNING (2) ► STATE_FINISHED (3) ► STATE_COMPENSATING (4) ► STATE_INDOUBT (10) ► STATE_FAILED (5) ► STATE_TERMINATED (6) ► STATE_COMPENSATED (7) ► STATE_COMPENSATION_FAILED (12) ► STATE_TERMINATING (8) ► STATE_FAILING (9) ► STATE_SUSPENDED (11)
CREATED	Timestamp	The time the process instance is created.
STARTED	Timestamp	The time the process instance started.
COMPLETED	Timestamp	The time the process instance completed.
PARENT_PIID	ID	The ID of the parent process instance.
PARENT_NAME	String	The name of the parent process instance.
TOP_LEVEL_PIID	ID	The process instance ID of the top-level process instance. If there is no top-level process instance, this is the process instance ID of the current process instance.
TOP_LEVEL_NAME	String	The name of the top-level process instance. If there is no top-level process instance, this is the name of the current process instance.

Column name	Type	Comments
STARTER	String	The principal ID of the starter of the process instance.
DESCRIPTION	String	If the description of the process template contains placeholders, this column contains the description of the process instance with the placeholders resolved.
TEMPLATE_NAME	String	The name of the associated process template.
TEMPLATE_DESCR	String	Description of the associated process template.
RESUMES	Timestamp	The time when the process instance is to be resumed automatically.

For a complete list of available Business Process Choreographer views see:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/c6bpel_view.html

A code example of a query based on the PROCESS_INSTANCE view is shown in Example 12-16.

Example 12-16 Query using arguments of PROCESS_INSTANCE view

```
QueryResultSet result =
   bfm.query(
      "DISTINCT PROCESS_INSTANCE.PIID",                              //select clause
      "PROCESS_INSTANCE.STATE = PROCESS_INSTANCE.STATE.STATE_RUNNING AND " +
      " ( WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_STARTER OR " +
      "   WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_ADMINISTRATOR ) AND " +
      "PROCESS_INSTANCE.TEMPLATE_NAME = 'OrderApprovalProcess' ",     // where clause
      null,                                                          // order-by clause
      null,
      new Integer(1),                                                // threshold
      null
);
```

For more information about filtering and sorting on Business Process Choreographer based queries see the white paper Business Process Choreographer query() and queryAll() - How to access processes, tasks and work items through the API and JDBC:

http://www-1.ibm.com/support/docview.wss?rs=2307&uid=swg27010849

12.5.3 Queries based on business data

A frequent task in managing business processes is to locate (query/sort) business processes, activities, or tasks within a business process based on business data.

WebSphere MQ Workflow introduced the Global Data Container to fulfill this requirement (and others).

The Business Process Choreographer (BPC) of WebSphere Process Server introduced query properties (QP) to locate BPC-related objects based on business data. An additional approach or alternative is to use custom properties.

Comparison

Table 12-10 provides an introduction to and a high-level comparison of Global Data Container and query properties.

Table 12-10 Comparison of Global Data Container and query properties

WebSphere MQ Workflow Global Data Container	Business Process Choreographer query properties
A Global Data Container is a container that exists on the process instance level.	Query properties are parts of variables that exist on the process instance level. Variables below the process level cannot be defined as QP.
It can contain data from the input container of the process instance as well as data from the output containers of activity instances.	It can contain data from the input message of a process start as well as data from processing of activities or tasks.
A Global Data Container is updated whenever a container providing input (via data mapping) for the global container is changed.	A query property is updated whenever the flow engine updates an associated variable. The trigger for that could be an activity, such as Receive/Assign/Invoke/Java snippet/etc.

WebSphere MQ Workflow Global Data Container	Business Process Choreographer query properties
A Global Data Container and its usage is defined during modeling, for example, with the WebSphere MQ Workflow Buildtime: ► Its data structure. ► Its database table and table index names (that is, the storage of derived global container instances). ► Its references by process templates. A global container can be referred to by a single process template or by multiple process templates. In runtime, there exists one global container instance per process instance derived from a process template using a global container. ► The data mapping from the source of the process template or from the output containers of activities.	Query properties are defined during modeling, for example, with the WebSphere Integration Developer: ► Its name. ► Its assignment to a part of a process-level variable. ► Its namespace (implicit creation when using WebSphere Integration Developer). ► No additional database tables are created for query properties. ► The runtime values of query properties can be influenced by modeling constructs, like Java snippets, Assign activities, and so on.
In Runtime, there is one Global Data Container instance per process instance. It is derived from a process template that uses a Global Data Container.	In Runtime, there can be more than one Business Process Execution Language variable at the process-level. The query property can be defined on each part of these variables.
Data can be mapped from the source of the process template or from the output containers of activities to the Global Data Container.	For the query property no additional mapping mechanism is needed on process instance level variable parts. Since parts of inner scope variables cannot be defined as query property, a special data mapping is needed. This can be done via features like *assign activities* and Java snippets.
The Global Data Container stores business-relevant data for a process and its activities and supports: ► Process instance queries ► Work item (item) queries ► Activity instance queries ► Audit trailing	Query properties store business-relevant data for a process and its activities and inline tasks. Query properties can be combined with all entities supported by the query() application programming interface method, including: ► Process instances ► Activity instances ► Inline tasks

WebSphere MQ Workflow Global Data Container	Business Process Choreographer query properties
A global data container is storage for data that is shared between all activities in a process.	Query properties are additional storage for parts of process-level BPEL variables that are to be used in queries.

Model definition details

This section describes the Global Data Container modeling and definition aspects in WebSphere MQ Workflow and query property modeling and definition aspects in WebSphere Process Server, and shows how the model-related definitions are mapped from WebSphere MQ Workflow to WebSphere Process Server.

Figure 12-25 shows an overview of Global Data Container modeling and definition aspects in WebSphere MQ Workflow.

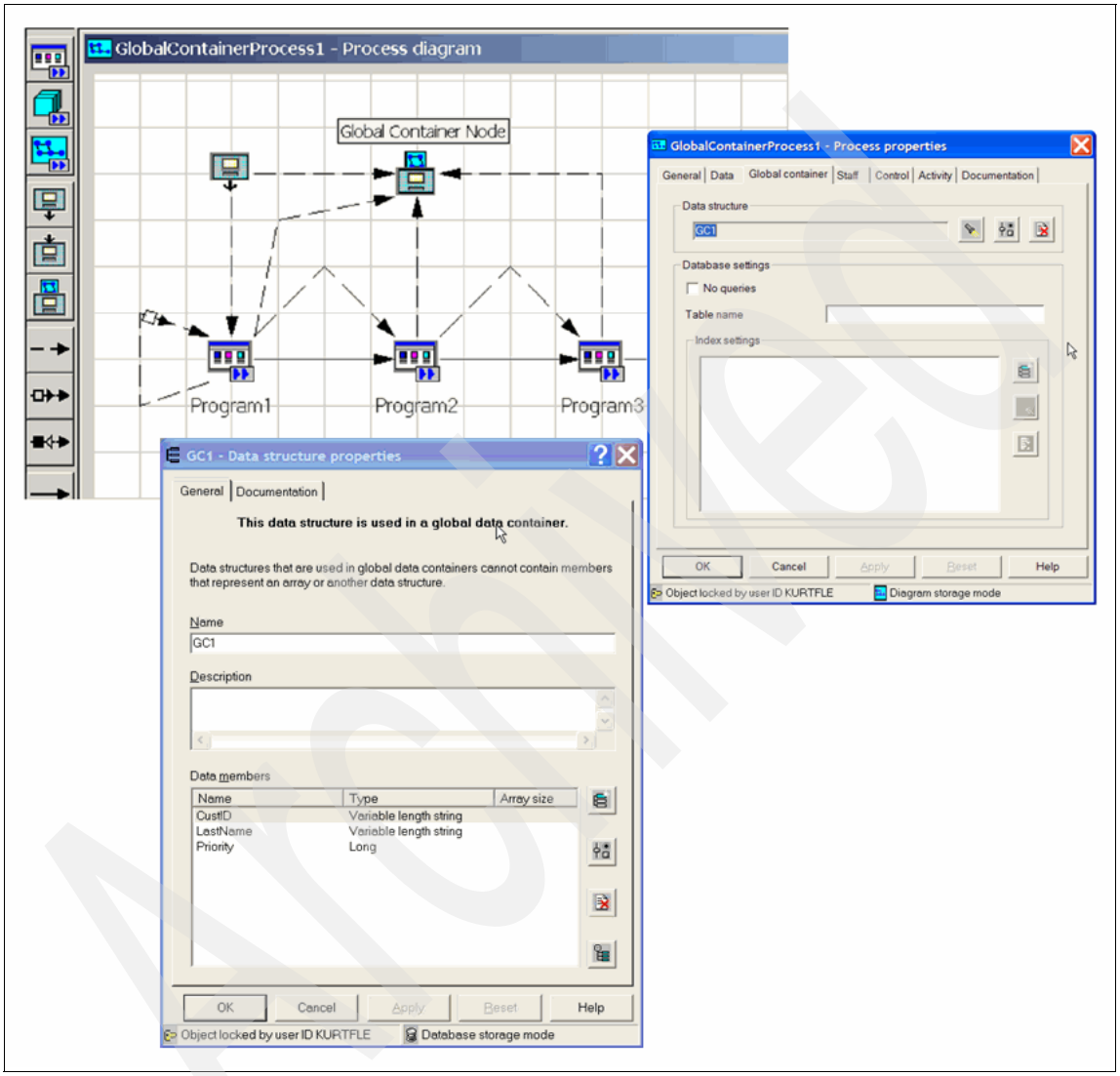


Figure 12-25 Global data container-based process model

Figure 12-26 shows an overview of query property modeling and definition aspects in WebSphere Process Server.

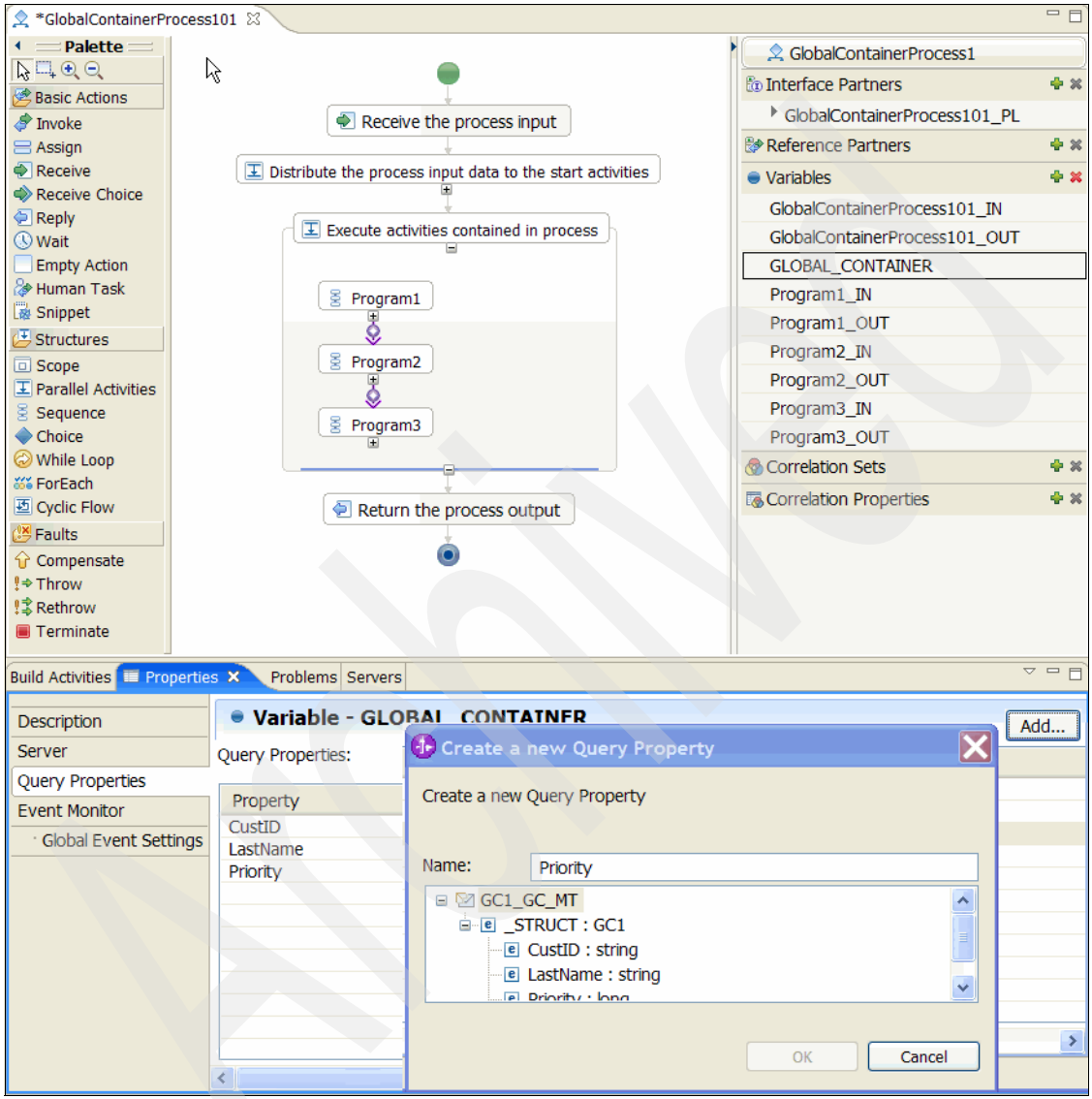


Figure 12-26 WebSphere Process Server query properties-based process model

Figure 12-27 shows how the data definition is mapped from WebSphere MQ Workflow to Business Process Choreographer of WebSphere Process Server. On the left it shows the MQWorkflow Definition Language (FDL) extract and on the right it shows the Business Process Execution Language (BPEL) extracts.

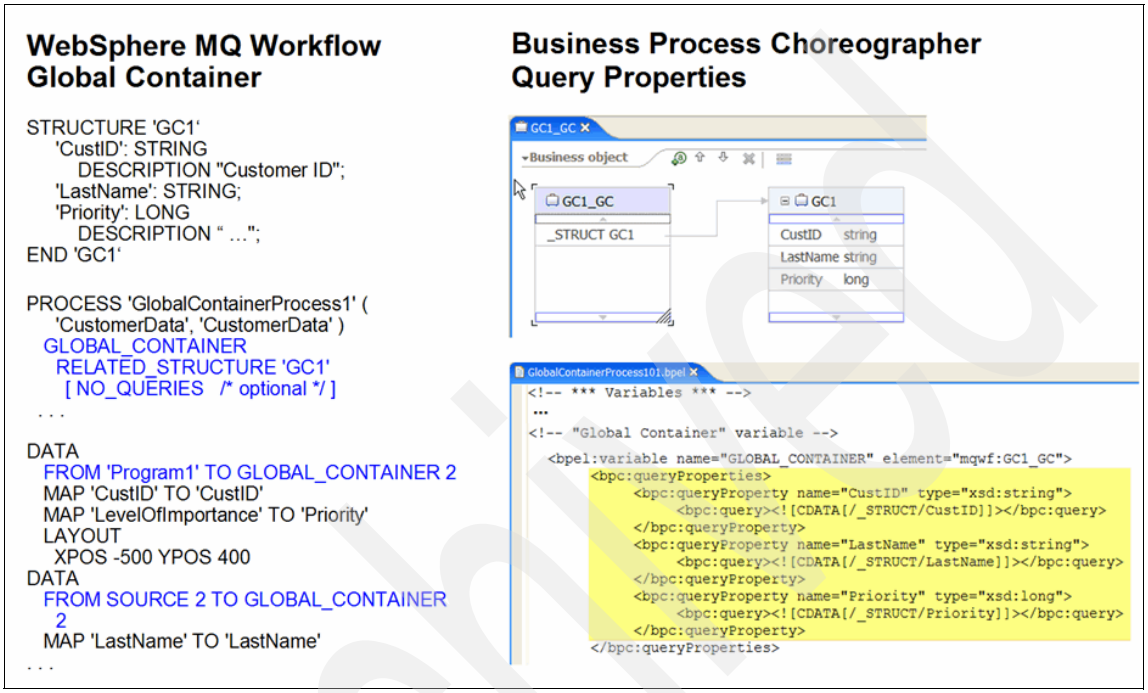


Figure 12-27 FDL-based Global Data Container definition versus BPEL-base query property definition

Model transition possibilities relating to Global Data Container

Re-implementation of the process model can be done by creating a new WebSphere Process Server-like process model and then defining query properties or custom properties for runtime query purposes.

For more details about how this is done refer to Chapter 9, “Business Process conversion” on page 187, or to the FDL-to-BPEL conversion by using the WebSphere Integration Developer FDL import functionality. Also see the documentation of the SupportPac WA73: FDL2BPEL Conversion Tool:

http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

API methods to discover Global Data Container and query properties

Table 12-11 provides an overview of the available APIs to retrieve a Global Data Container and query properties.

Table 12-11 API methods to retrieve Global Data Container and query properties

WebSphere MQ Workflow Global Data Container-based API methods	Business Process Choreographer query properties-based API methods
Package com.ibm.workflow.api Interface ProcessInstance public ReadOnlyContainer globalContainer() throws FmcException This API call retrieves the global container associated with the process instance from the MQ Workflow execution server (action call).	Package com.ibm.bpe.api Interface BusinessFlowManager public java.util.List getQueryProperties(PTID ptid) throws … Retrieves the query properties of the specified process template using the process template ID. Returns a list of query properties.
API methods of the following com.ibm.workflow.api Interfaces could be used to extract the details: ► Container ► ReadOnlyContainer ► ContainerElement	APIs of the following com.ibm.bpe.api Interface could be used to extract the details: QueryProperty. For example: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_queryvariable.html

Filter and sort criteria syntax for business-data-based queries

This section describes the filter criteria syntax (where clause) for a query in WebSphere MQ Workflow and WebSphere Process Server.

WebSphere MQ Workflow

Figure 12-28 shows the filter criteria syntax for a query of:

- Process instances
- Activity instances
- Work items (tasks)

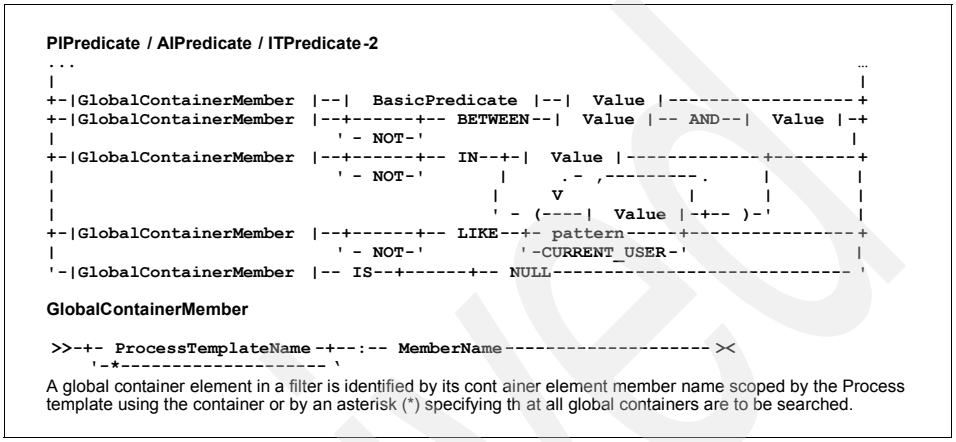


Figure 12-28 WebSphere MQ Workflow filter syntax

Figure 12-29 shows the sort criteria syntax for query of:

- Process instances
- Activity instances
- Work items (tasks)

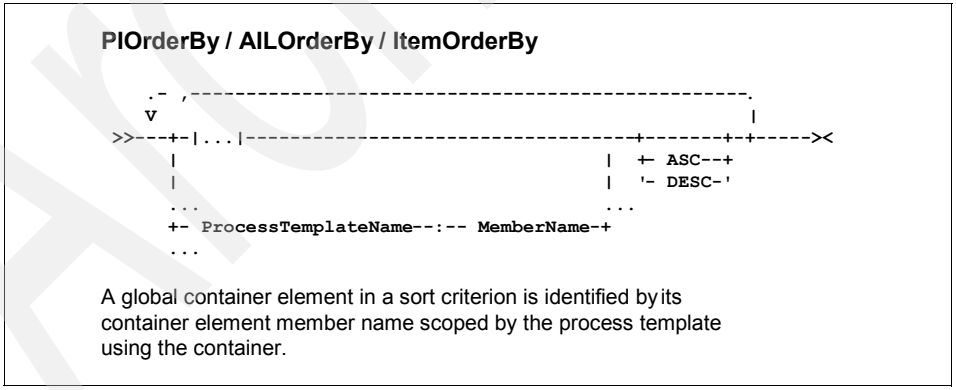


Figure 12-29 WebSphere MQ Workflow sort criteria syntax

WebSphere Process Server

The filter and sort criteria syntax is SQL-like. The filter criteria is called the where clause and the sort criteria is called the order-by clause.

Query properties can be combined with all entities supported by the query() API, including process instances, activity instances, and inline tasks.

The syntax for filter criteria (where clause) and sort criteria (order-by clause) for queries based on query properties is specified by the predefined QUERY_PROPERTY view of Business Process Choreographer in WebSphere Process Server. This view is based on the modeled process-level variables.

Table 12-12 shows the QUERY_PROPERTY view.

Table 12-12 Columns in the QUERY_PROPERTY view

Column name	Type	Comments
VARIABLE_NAME	String	The name of the process-level variable.
PIID	ID	The process instance ID.
NAME	String	The name of the query property.
NAMESPACE	String	The namespace of the query property.
GENERIC_VALUE	String	A string representation for property types that do not map to one of the defined types: STRING_VALUE, NUMBER_VALUE, DECIMAL_VALUE, or TIMESTAMP_VALUE.
STRING_VALUE	String	If a property type is mapped to a string type, this is the value of the string.
NUMBER_VALUE	Integer	If a property type is mapped to an integer type, this is the value of the integer.
DECIMAL_VALUE	Decimal	If a property type is mapped to a floating point type, this is the value of the decimal.
TIMESTAMP_VALUE	Timestamp	If a property type is mapped to a timestamp type, this is the value of the timestamp.

When using multiple query properties within arguments of a query(…) API, add a one-digit counter to the view name. This counter can take the values 1 through 9.

Filter and sort criteria for queries are mapped to SQL where and order-by clauses. Combinations of multiple entities in the query are implemented using join predicates for the underlying view.

Query examples for process instances

In this section we describe query examples for process instances.

WebSphere MQ Workflow

This section discusses query parameters of WebSphere MQ Workflow. Table 12-13 shows typical query parameters with example values.

Table 12-13 WebSphere MQ Workflow process instance query parameters

Query parameter	Query parameter values (example)
Select clause	Via typed-based API, for example, queryProcessInstances(…)
Where clause	GlobalContainerProcess1:CustID = ‘5711'
Order-by clause	GlobalContainerProcess1:Priority ASC

WebSphere Process Server

This section discusses query parameters of WebSphere Process Server. Figure 12-14 shows the typical query parameters with example values.

Table 12-14 WebSphere Process Server process instance query parameters

Query parameter	Query parameter values (example)
Select clause	QUERY_PROPERTY1.PIID, PROCESS_INSTANCE.NAME
Where clause	QUERY_PROPERTY1.NAME = 'CustID' AND QUERY_PROPERTY1.STRING_VALUE = ‘5711' ANDQUERY_PROPERTY1.NAMESPACE = 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/’ AND QUERY_PROPERTY2.NAME = 'Priority' AND QUERY_PROPERTY2.NAMESPACE = 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/’
Order-by clause	QUERY_PROPERTY2.NUMBER_VALUE ASC

Query examples for activity instances

In this section we describe query examples for activity instances.

WebSphere MQ Workflow

This section discusses query parameters of WebSphere MQ Workflow. Table 12-15 shows typical query parameters with example values.

Table 12-15 WebSphere MQ Workflow activity instance query parameters

Query parameter	Query parameter values (example)
Select clause	Via typed-based API, for example queryActivityInstances(…)
Where clause	GlobalContainerProcess1:CustID = ‘5711‘ AND NAME = ‘Program2’
Order-by clause	GlobalContainerProcess1:Priority ASC

WebSphere Process Server

This section discusses query parameters of WebSphere Process Server. Table 12-16 shows typical query parameters with example values.

Table 12-16 WebSphere Process Server activity instance query parameters

Query parameter	Query parameter values (example)
Select clause	ACTIVITY.AIID, ACTIVITY.TEMPLATE_NAME
Where clause	QUERY_PROPERTY1.NAME = 'CustID' AND QUERY_PROPERTY1.STRING_VALUE = ‘5711' ANDQUERY_PROPERTY1.NAMESPACE = 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/’ AND QUERY_PROPERTY2.NAME = 'Priority' AND QUERY_PROPERTY2.NAMESPACE = 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/’ AND ACTIVITY.TEMPLATE_NAME = ‘Program2’
Order-by clause	QUERY_PROPERTY2.NUMBER_VALUE ASC

Query examples for inline tasks

In this section we describe query examples for inline tasks.

WebSphere MQ Workflow

This section discusses query parameters and shows a query code snippet. Table 12-17 shows typical query parameters with example values.

Table 12-17 WebSphere MQ Workflow work item query parameters

Query parameter	Query parameter values (example)
Select clause	Via typed-based API, for example queryWorkItems(…), queryItems(…), createWorklist()
Where clause	GlobalContainerProcess1:CustID = ‘5711' AND STATE IN (READY) AND OWNER=CURRENT_USER
Order-by clause	GlobalContainerProcess1:Priority ASC

Example 12-17 shows a code snippet of a work item query example based on the WebSphere MQ Workflow Java API.

Example 12-17 WebSphere MQ Workflow work item query (code snippet)

```
WorkItem[] wiList = queryWorkItems(                              // select
                    “GlobalContainerProcess1:CustID = ‘5711' AND  // where
                        STATE IN (READY) AND OWNER=CURRENT_USER”,
                    “GlobalContainerProcess1:Priority ASC”,       // order
                    100 );                                        // threshold
```

WebSphere Process Server

This section discusses query parameters and shows a query code snippet. Table 12-18 shows typical query parameters with example values.

Table 12-18 WebSphere Process Server task query parameters

Query parameter	Query parameter values (example)
Select clause	TASK.TKIID, TASK_TEMPL.NAME
Where clause	QUERY_PROPERTY1.NAME = 'CustID' AND QUERY_PROPERTY1.STRING_VALUE = ‘5711' AND QUERY_PROPERTY1.NAMESPACE = 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/’ AND QUERY_PROPERTY2.NAME = 'Priority' AND QUERY_PROPERTY2.NAMESPACE = 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/’ AND TASK.STATE = TASK.STATE.STATE_READY AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER

Query parameter	Query parameter values (example)
Order-by clause	QUERY_PROPERTY2.NUMBER_VALUE ASC

Table 12-18 shows a code snippet of a task query example of the WebSphere Process Server EJB-based Java API of the Human Task Manager.

Example 12-18 WebSphere Process Server task query (code snippet)

```
QueryResultSet result =
  process.query(
                "TASK.TKIID,  TASK_TEMPL.NAME",                          // select
                "QUERY_PROPERTY1.NAME = 'CustID'                         // where
                 AND QUERY_PROPERTY1.STRING_VALUE = '5711'
                 AND QUERY_PROPERTY1.NAMESPACE =
                   'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/'
                 AND QUERY_PROPERTY2.NAME = 'Priority'                   // order prep.
                 AND QUERY_PROPERTY2.NAMESPACE =
                   'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/'
                 AND TASK.STATE = TASK.STATE.STATE_READY
                 AND WORK_ITEM.REASON =
                   WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                "QUERY_PROPERTY2.NUMBER_VALUE ASC",                      // order-by
                null, 100, null );                                       // threshold, …
```

Recommendation

Keep the number of query properties as small as possible, especially in the filter and sort criteria (where and order-by clause) of queries to avoid performance impacts.

References

See the following WebSphere MQ Workflow Global Data Container references:

- *BM WebSphere MQ Workflow Concepts and Architecture, Version 3.6,* GH12-6285, Chapter 4: Running your business processes: Monitoring, locating, and analyzing processes

  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=GH12-6285-07

- *IBM WebSphere MQ Workflow Getting Started with Buildtime, Version 3.6*, SH12-6286

  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SH12-6286-10

- Buildtime Online Help
- *IBM WebSphere MQ Workflow API Programming Guide, Version 3.6*, SH12-629:
  - Chapter 10: Global containers
  - Chapter 40: Execution service actions: queries and create lists
  - Chapter 45: Process instance actions: GlobalContainer() API

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/ref/rquery.html
- WebSphere MQ Workflow JavaDoc: <WMQWF_InstallDirectory>\DOC\javaapi\index.html
- *IBM MQ Workflow Administration Guide, Version 3.6,* SH12-6289-08, Chapter 8: How to use an audit trail
- *IBM WebSphere MQ Workflow Installation Guide, Version 3.6*, SH12-6288-11, Appendix D: Optimizing the Runtime database
- IBM WebSphere MQ Workflow online publication:

  http://www-1.ibm.com/support/search.wss?tc=SSVLA5&rs=795&rank=8&dc=DB520+D800+D900+DA900+DA800&dtm
- SupportPac WC01: WebSphere MQ Workflow Data Base Tools:

  http://www-1.ibm.com/support/docview.wss?rs=171&q1=IH06&uid=swg24008199&loc=en_US&cs=utf-8&lang=en

The following are WebSphere Process Server/WebSphere Integration Developer query property references:

- QUERY_PROPERTY view

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/r6bpel_viewqueryprops.html
- Filtering data using variables in queries

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_queryvariable.html
- JavaDoc

  Package com.ibm.bpe.api:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/package-summary.html

Package com.ibm.bpe.api Interface BusinessFlowManager:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/BusinessFlowManager.html

Interface QueryProperty:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/QueryProperty.html

- WebSphere Integration Developer online help

  Query properties tab: business process editor:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/ref/rquery.html

  Declaring a query property for a variable:

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/tasks/tdecqprp.html

- SupportPac WA73: FDL2BPEL Conversion Tool

  http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

- *Query Properties* sample of the *Business Process Management Samples & Tutorials* - Version 6.1

  http://publib.boulder.ibm.com/bpcsamp/index.html?gettingStarted&processInteraction/queryProperties.html

12.6 Authentication implementations

This section describes the authentication implementations in WebSphere MQ Workflow and WebSphere Process Server.

WebSphere MQ Workflow

Authentication of users who interact with a WebSphere MQ Workflow Runtime environment is part of the WebSphere MQ Workflow runtime engine and is handled by the WebSphere MQ Workflow administration server. To communicate with an MQ Workflow server, a session must have been established between the user (API-based client application) and this server. The session is established by logging on.

In the API-based client application you are required to allocate a service object that represents the server from which the services are requested. Once the service object is allocated, logon can be performed.

Logon establishes a session between the user logging on and the server represented by the service object. All subsequent calls requiring client/server communication run through this session.

Logon requires that the administration server is running on the selected system because the administration server manages sessions and checks the authentication of the user.

For client applications WebSphere MQ Workflow offers different API methods for the authentication:

- logon1(...)
- logon 2(...)

WebSphere MQ Workflow also provides an exit for a third-party authentication, with is called authentication exit. To communicate with this exit the following client APIs exist:

- logon3(...)
- logon4(...)

The authentication exit itself also provides an application programming interface.

Fore more details refer to *IBM WebSphere MQ Workflow Programming Guide Version 3.6:*

- Chapter 6: An MQ Workflow session
- Chapter 7: Using an authentication exit
- Chapter 16: An MQ Workflow client application
- Chapter 40: Execution service actions (Logon/Logoff)

WebSphere Process Server

WebSphere Process Server itself does not provide authentication services. It uses the WebSphere Application Server infrastructure, which provides this kind of service, the Java Authentication and Authorization Service (JAAS). For more details refer to the following publications:

- *Reference Guide for the Java 2 SDK, Standard Edition, v 1.4*

  http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html

- Customizing application login with Java Authentication and Authorization Service

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_j2clogin.html

WebSphere Application Server offers different logins for authentications, for example:

- WAS programmatic login

  Developing programmatic logins with the Java Authentication and Authorization Service:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_pacs.html
  Example: Programmatic logins
  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/xsec_jaas.html

- WAS form-based login

  Customizing Web application login:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_pofolo.html
  Example: Form login
  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/xsec_formlogin.html

- JAAS exit

  IBM WebSphere Developer Technical Journal: Advanced authentication in WebSphere Application Server, Managing user authenticity and privileges in a distributed application server environment:

  http://www-128.ibm.com/developerworks/websphere/techjournal/0508_benantar/0508_benantar.html

12.7 Authorization management

This section discusses authorization management of WebSphere MQ Workflow and WebSphere Process Server.

12.7.1 WebSphere MQ Workflow

In WebSphere MQ Workflow, authorization and staff resolution are different concepts. In WebSphere MQ Workflow, authorizations can be granted explicitly or implicitly. Authorization is handled with the WebSphere MQ Workflow execution server based on a valid session.

Any objects that are retrieved or created belong to the session in which they have been queried or created. They carry the session identification so that further actions on these objects are executed in the same session with the authorization of the logged-on user.

After a successful logon, action API calls in order to query or manage MQ Workflow objects can be issued for which authorization is established.

For more details refer to 4.1.3, "Tools to access and modify the WebSphere MQ Workflow staff repository" on page 60, or to the *IBM WebSphere MQ Workflow Programming Guide Version 3.6*:

- Chapter 6: An MQ Workflow session
- Chapter 12: Authorization considerations
- Chapter 16: An MQ Workflow client application.

12.7.2 WebSphere Process Server

Users interact either with the Business Process Choreographer Explorer or with another client based on the Business Process Choreographer APIs. These client applications send their requests to either the Business Flow Manager Process API or the Human Task Manager API. Authentication is provided by the WebSphere Application Server security component, and the instance-based and rule-based authorization is provided by the Human Task Manager using work items.

Human Task Manager authorization management coordinates other components when evaluating the modeled authorization rules and creates work items that store the result of this evaluation. When a client invokes a Business Flow Manager or Human Task Manager API, the authorization manager checks the stored work items for a matching authorization of the calling user. The same applies to the result set entries when a user queries items on the Business Process Choreographer database view using one of the query() API methods.

Figure 12-30 illustrates how WebSphere Process Server uses the JAAS login context provided by the WebSphere Application Server base services when performing a client-side API method and how the Human Task Manager authorization management checks the authority.

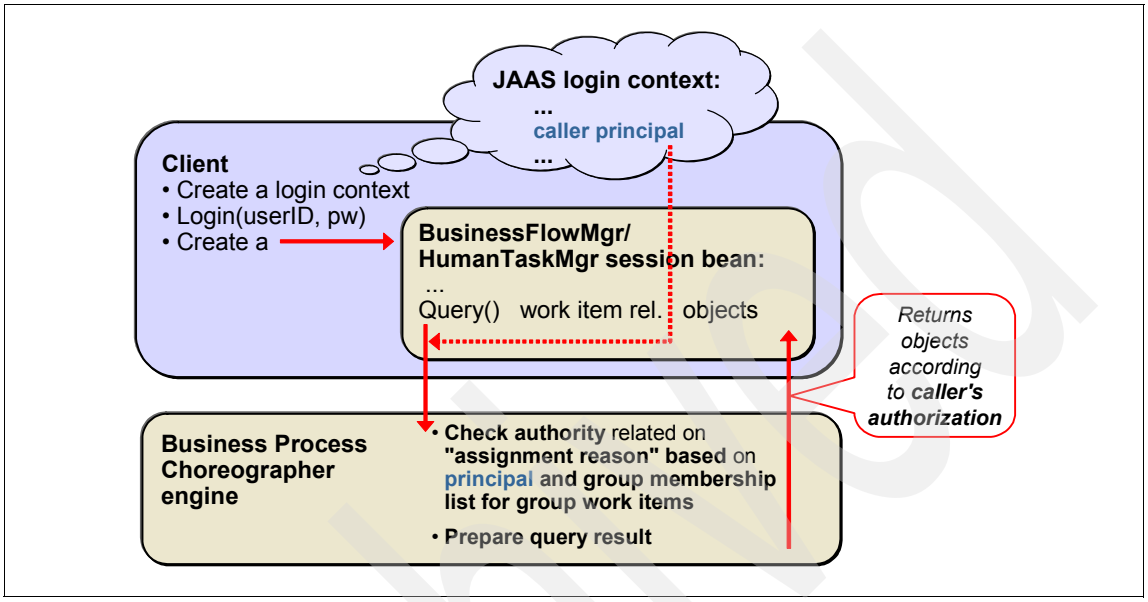


Figure 12-30 WebSphere Process Server authentication and authorization

For more information see 4.2.4, “Authorization concepts” on page 72, and *Authorization and staff resolution in Business Process Choreographer: Part 1: Understanding the concepts and components of staff resolution* at:

http://www.ibm.com/developerworks/websphere/techjournal/0710_lind/0710_lind.html

12.8 Application programming interface package overview

WebSphere MQ Workflow and WebSphere Process Server offer different application programming interfaces for the different components and the different functional areas. The following graphics and table provide an overview of the functional areas and how they are mapped from WebSphere MQ Workflow to WebSphere Process Server.

Figure 12-31 shows the WebSphere MQ Workflow architecture with clients and application programming interface.

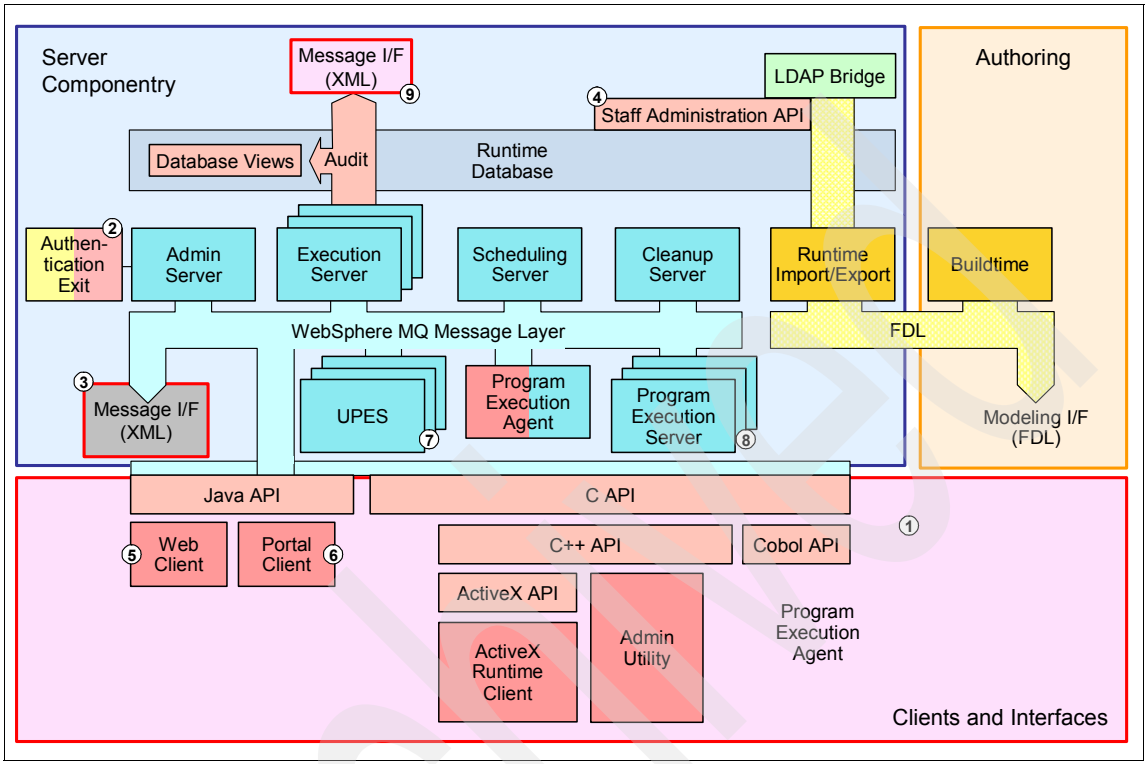


Figure 12-31 WebSphere MQ Workflow architecture with clients and application programming interface

Note: The numbers in Figure 12-31 are referred to in the API overview table below (Table 12-19).

Figure 12-32 shows the Business Process Choreographer architecture of WebSphere Process Server.

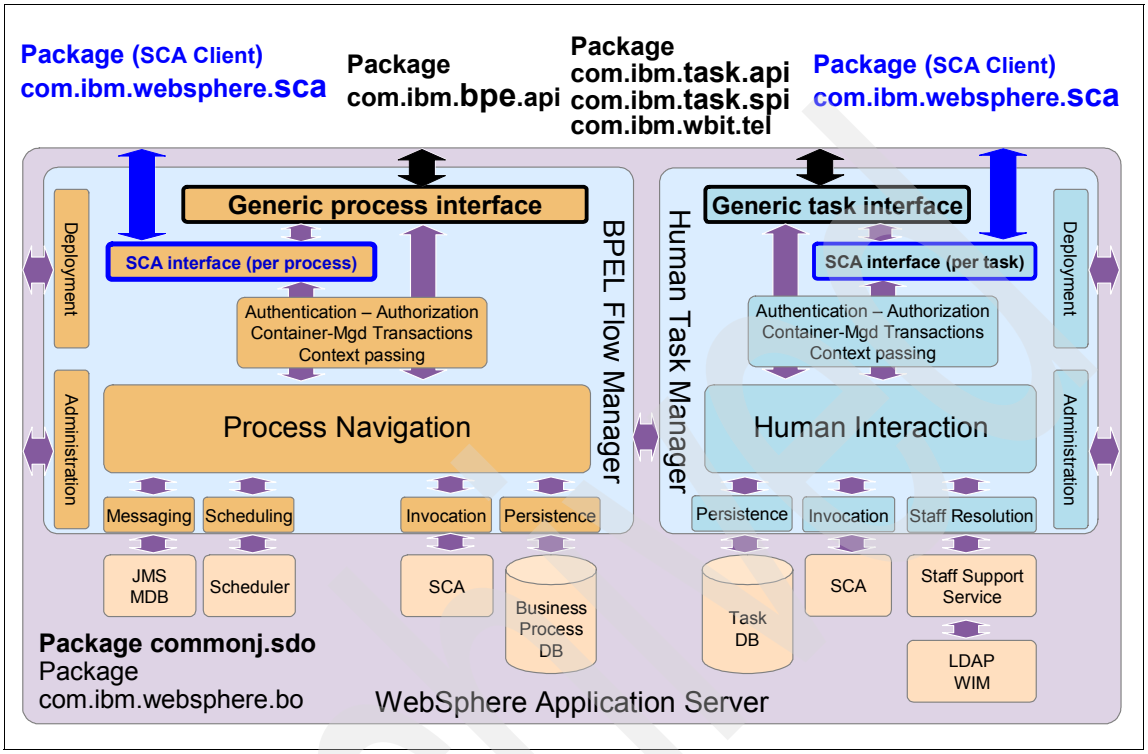


Figure 12-32 Business Process Choreographer architecture of WebSphere Process Server

Application programming interface package mapping

Table 12-19 provides an overview of different API packages of both products and how these API packages are mapped on a high level.

Table 12-19 Application programming interfaces packages overview

API area	WebSphere MQ Workflow	WebSphere Process Server
Processes, activities	WebSphere MQ Workflow runtime API (1) Package com.ibm.workflow.api <MQWFInstallationDirectory>/doc/javaapi/index.html	Business Flow Manager application programming interface Package com.ibm.bpe.api http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/package-summary.html
Human tasks		Human Task Manager application programming interface Package com.ibm.task.api http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/advanced/content.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/task/api/package-summary.html Additional functionality: ► com.ibm.task.spi ► com.ibm.wbit.tel
Data		Service Data Object application programming interface Package commonj.sdo Additional functionality: Business Object application programming interface Package com.ibm.websphere.bo http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wsps.602.javadoc.doc/doc/com/ibm/websphere/bo/package-summary.html
Authentication		WebSphere Application Server foundation services based on Java Authentication and Authorization Service (JAAS)
	Authentication exit API (2)	Java Authentication and Authorization Service exit http://www-128.ibm.com/developerworks/websphere/techjournal/0508_benantar/0508_benantar.html

API area	WebSphere MQ Workflow	WebSphere Process Server
Service Component Architecture		Service Component Architecture application programming interface Package com.ibm.websphere.sca
Inbound message driven interface	Inbound MQ-based XML interface (3): FmcXML-Client.xsd, FmcXML-Server.xsd, FmcXML-MsgSample.xsd <MQWFInstallationDirectory>\bin	Java Message Service application programming interface
Staff administration	Staff administration application programming interface (4) Package com.ibm.workflow.staff <MQWFInstallationDirectory>/doc/StaffAPI/index.html	API of the various people directory providers
Web client customizing	Web client framework application programming interface (5) Package com.ibm.workflow.servlet.client <MQWFInstallationDirectory>/doc/WebClient/api/index.html	Process Explorer framework application programming interface Packages: ► com.ibm.bpc.clientcore ► com.ibm.bpc.clientcore.converter ► com.ibm.bpc.clientcore.exception ► com.ibm.bpe.clientmodel ► com.ibm.bpe.clientmodel.bean ► com.ibm.bpe.jsf.component.taglib ► com.ibm.bpe.jsf.util ► com.ibm.bpe.jsf.handler ► com.ibm.task.clientmodel ► com.ibm.task.clientmodel.bean
Portal client framework	Package com.ibm.workflow.portlet.client (6) <MQWFInstallationDirectory>/doc/PortalClient/javadoc/index.html	► Portlet generator ► Portlet factory

API area	WebSphere MQ Workflow	WebSphere Process Server
Back-end interface: User-defined program execution server	► User-defined program execution server XML interface (7) FmcXML-Upes.xsd, FmcXML-Types.xsd, FmcXML-MsgSample.xsd <MQWFInstallationDirectory>\bin ► User-defined program execution server framework interface (8) Packages com.ibm.workflow.catalog.upes com.ibm.workflow.schema com.ibm.workflow.schema.source com.ibm.workflow.upes com.ibm.workflow.util.msg com.ibm.workflow.util.msg.fmt <MQWFInstallationDirectory>/doc/UpesFramework/index.html	► Service Component Architecture application programming interface (SCA) Package com.ibm.websphere.sca ► In case of user-defined program execution server (UPES) re-use (FDL2BPEL), a special WebSphere MQ Workflow SCA binding is used.
Back-end interface: PES	► “COMMAREA (best practices) ► Container API within Package com.ibm.workflow.api ► MDL	
Back-end interface: Program execution agent	Container API within Package com.ibm.workflow.api	

13

Implementing operational aspects

This chapter provides the detailed information required to assess the WebSphere MQ Workflow topology in place and set up the desired target topology in a WebSphere Process Server environment. It helps you define what is in place for WebSphere MQ Workflow for:

- The WebSphere MQ Workflow topology
- The WebSphere Process Server topology
- High availability, workload management, and scalability in WebSphere Process Server
- Monitoring and auditing
- Cleanup of processes and tasks
- Best practices and recommendations

We recommend first reading Chapter 7, “Planning for operational aspects” on page 135, which provides information about the basic architecture of WebSphere MQ Workflow and WebSphere Process Server and to use Appendix A, “Transition planning worksheets” on page 411, to note the major planning decisions.

13.1 The WebSphere MQ Workflow topology

This section provides information and a planning sheet for gathering information about the infrastructure currently in place for WebSphere MQ Workflow. In particular:

- The machines in the setup
- The WebSphere MQ Workflow setup type, including the database setup and performance considerations
- The implemented high-availability and security options

13.1.1 Gathering information about the machines in the setup

It is important to make a list of all the machines involved in the setup. For each system group, make a copy of Table 13-1. Note which machines are in use and what roles they have. It is also useful to draw a picture showing all machines, their roles, and relationships. For more information about different setup types, see 13.1.2, “Determining the WebSphere MQ Workflow setup type in place” on page 360.

Table 13-1 Assessment sheet for the machines in the WebSphere MQ Workflow setup

MQ Workflow machine types in system group:	**Host name**			
Buildtime hosts[1]				
Buildtime database hosts[2]				
Runtime database host	[3]			
Workflow systems (servers[4 9])		[5]	[5]	[5]
Web servers[6]				

MQ Workflow machine types in system group:	Host name			
Application servers with Web client[6]				
Portal server with portal client[7]				
Client concentrators[8] [9]				
Clients				
LDAP server				
...				
...				

Notes on Table 13-1 on page 358:

1. The Buildtime tool only runs on Windows. Since it is not transitioned to the WebSphere Process Server environment, the Buildtime component can be disregarded.
2. On Windows, the Buildtime database can be Microsoft Access or DB2. The Buildtime database can also be DB2 on a remote UNIX® machine. The Buildtime database can also be disregarded for transition to WebSphere Process Server.
3. Each system group has exactly one Runtime database. In a two-tier setup, the first server in the system group also hosts the Runtime database. In a three-tier setup, the Runtime database is hosted on a dedicated machine that has no WebSphere MQ Workflow components installed on it.

4. A WebSphere MQ Workflow system can be a Runtime server, a Buildtime server, database utilities, or a client with queue manager. For example, to create a new WebSphere MQ Workflow system, the server option is selected when installing and configuring, but in fact, several different types of servers are installed and configured together.
5. Additional WebSphere MQ Workflow systems in a system group always use the existing Runtime database.
6. Depending on your setup, the Web server and application server can be on the same machine, LPAR, or installation as your WebSphere MQ Workflow server.
7. The WebSphere MQ Workflow Portal client allows WebSphere MQ Workflow systems to be accessed via a portal server.
8. Any WebSphere MQ Workflow client with its own queue manager can act as a client concentrator. The client concentrator hosts the client queues, receives client messages, and sends them to the WebSphere MQ Workflow server input queues defined in the MQ cluster.

13.1.2 Determining the WebSphere MQ Workflow setup type in place

WebSphere MQ Workflow can be set up in many different ways. The setup type determines, for example, the level of availability, failover, and scalability. The following scenarios briefly describe the essential elements of the most common setups to help assess the target topology required:

- Stand-alone server
- Standard client/server (two-tier)
- Client/servers with a remote database (three-tier)
- Client with queue manager
- Client concentrator

Stand-alone setup

The stand-alone workstation setup is mostly used to become familiar with WebSphere MQ Workflow functions for demonstration, evaluation, and development purposes. Figure 13-1 shows an example of a stand-alone setup.

Figure 13-1 Example of a stand-alone setup

Note: In the target environment, this is most likely represented by a WebSphere Integration Developer with built-in WebSphere Process Server.

For a Runtime environment, a single system is most likely represented by a stand-alone server or managed server in WebSphere Process Server.

Setup for standard client/server (two-tier)

In the standard client/server scenario, the WebSphere MQ Workflow server and client components are installed on different machines. The Runtime database is hosted on the same machine as the MQ Workflow server. More WebSphere MQ Workflow servers may have been added that share the same Runtime database.

Figure 13-2 illustrates a standard WebSphere MQ Workflow client/server setup using just one machine for the server components.

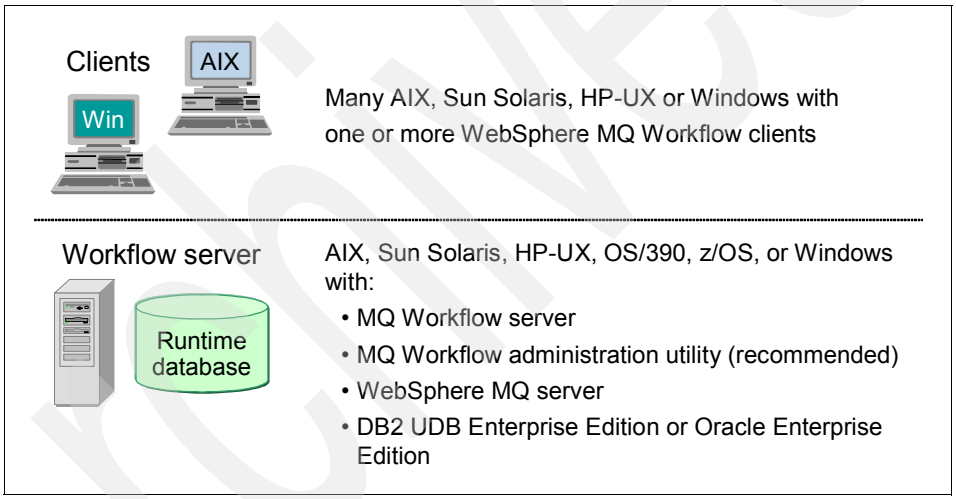


Figure 13-2 Example of a standard client/server setup (two-tier)

Note: In the target environment, this could be represented by a topology with a single WebSphere Process Server server or a cluster of servers with remote clients.

See 13.2.6, “WebSphere Process Server cluster patterns” on page 378, for more information about the available cluster topology options.

Setup for client/servers with a remote database (three-tier)

With this type of setup, the Runtime database is in a third tier, separate from the WebSphere MQ Workflow server tier. This setup is used for simplified management of WebSphere MQ Workflow data and allows for easy scaling of the workflow and database, as well as for workload distribution.

Figure 13-3 illustrates a remote database setup.

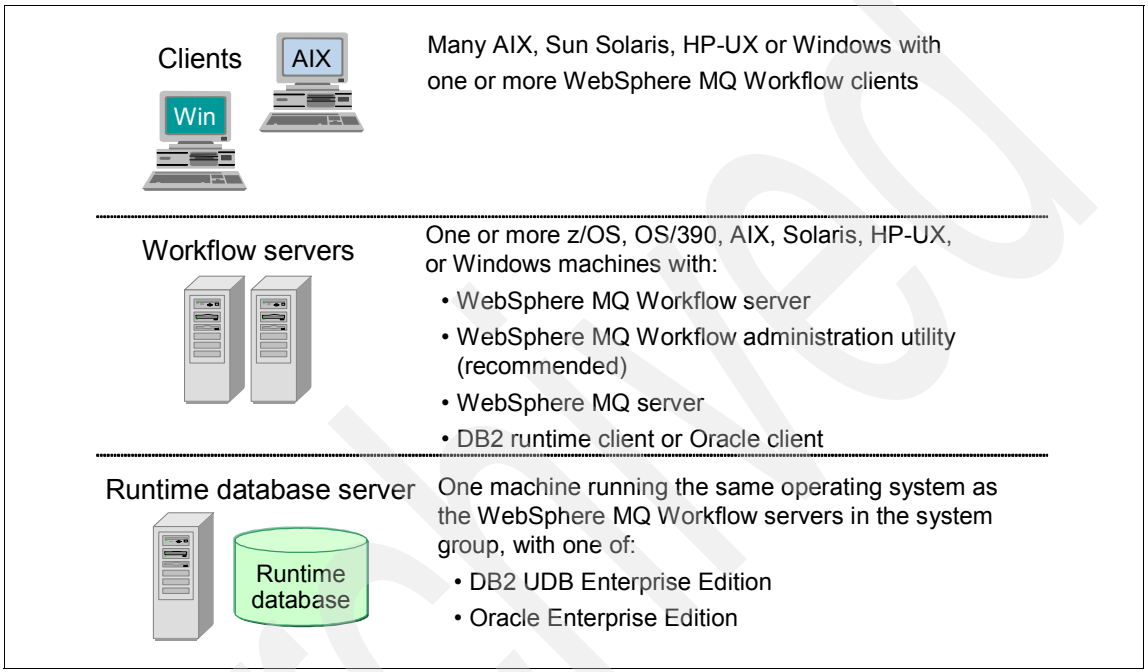


Figure 13-3 Example of a client/servers setup with a remote database (three-tier)

The WebSphere MQ Workflow clients may connect to different WebSphere MQ Workflow servers, but the requests from one client are always handled by the same server. Without client concentrators, workload can only be balanced by making a static allocation of clients to servers. All WebSphere MQ Workflow servers access the same database. All WebSphere MQ Workflow servers and the database server must be running the same operating system.

Note: In the WebSphere Process Server target topology, this would most likely be implemented as a clustered WebSphere Process Server environment with remote clients.

See 13.2.6, “WebSphere Process Server cluster patterns” on page 378, for more information about the available cluster topology options.

Setup scenario for client, client with queue manager, and client concentrator

The WebSphere MQ Workflow client can be set up either to use a WebSphere MQ client to communicate with WebSphere MQ Workflow servers or to use its own queue manager.

We recommend the MQ *client* connection for small numbers of clients without workload balancing. A *client with queue manager* is used for clients that also act as servers, such as the Web server in the Web client configuration. This is done if:

- Clients must host their own queues to relieve the servers of this work.
- A client concentrator setup is used, as described below.
- WebSphere MQ Workflow detects a WebSphere MQ server installed at the time the WebSphere MQ Workflow client is configured.

In a client concentrator setup, the clients use the WebSphere MQ client to connect to a dedicated queue manager, which then uses message channels to communicate with the WebSphere MQ Workflow server. In this setup, no clients connect directly to the server. Typically, such a configuration would consist of a few client concentrator queue managers, which are connected to several systems belonging to the same WebSphere MQ cluster. This setup is used for large numbers of clients, as well as for setups where workload is balanced across the WebSphere MQ Workflow systems in a cluster.

Figure 13-4 shows a client concentrator setup.

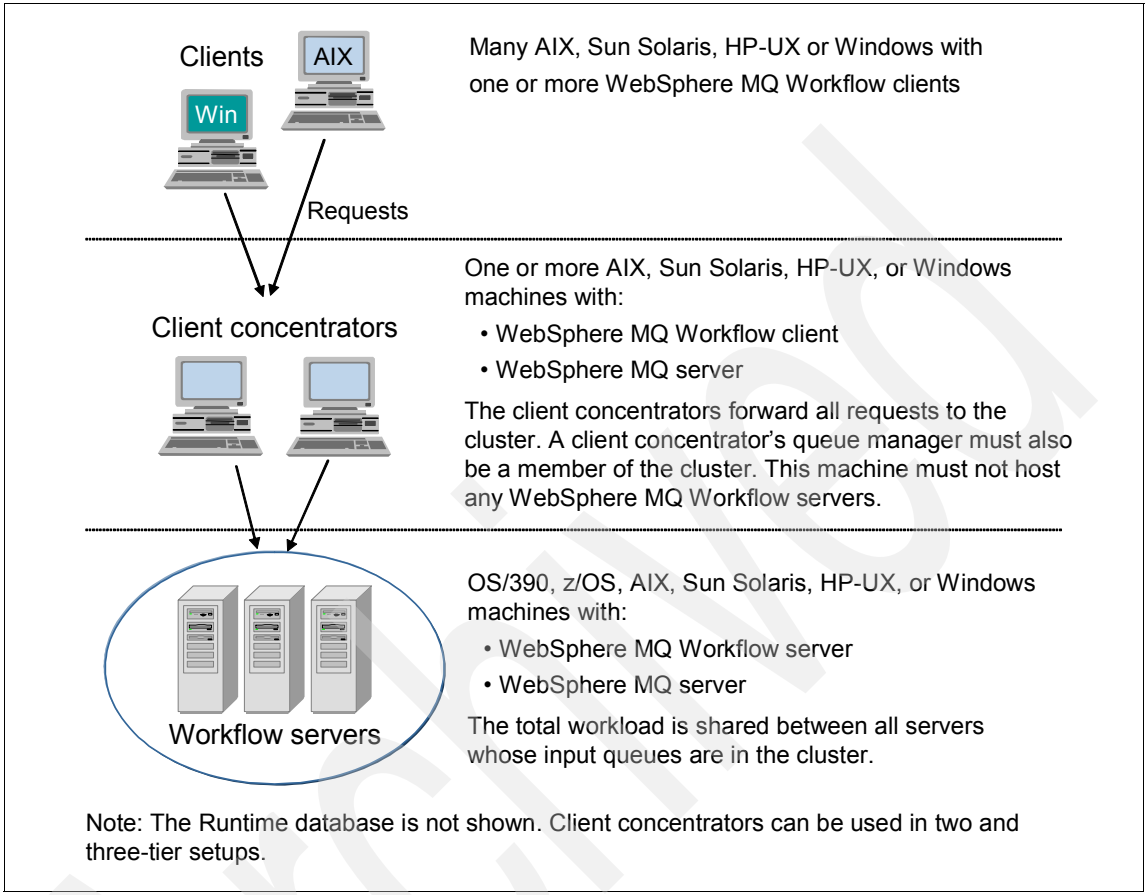


Figure 13-4 Example of a client concentrator setup

Note: In the WebSphere Process Server target topology, this would most likely be implemented as a clustered WebSphere Process Server environment with remote clients that have been set up for high availability and workload distribution.

See 13.2.6, "WebSphere Process Server cluster patterns" on page 378, for more information about the available cluster topology options.

Web clients

The Web client is a servlet and Java Server Pages based interface to WebSphere MQ Workflow. It runs on the application server (server-side Java).

On the client side, only a browser with JavaScript™ support is required. The Web client provides access to activity lists, process template lists, process instance lists, work lists, user and list settings, and also to object properties (input and output containers). It features full process, full worklist control, and process monitoring. It has the following scalability features:

- Several Web servers attached to one MQ Workflow system group
- Several MQ Workflow systems within one MQ Workflow system group
- Load distribution through WebSphere MQ queue manager clustering

The Web client is usually installed as a multi-tier setup in a WebSphere Application Server Network Deployment environment. This setup is suitable for workload distribution, increased availability, and scalability of the Web client application. The Web client is installed in a WebSphere Application Server cluster.

Figure 13-5 e shows this type of setup.

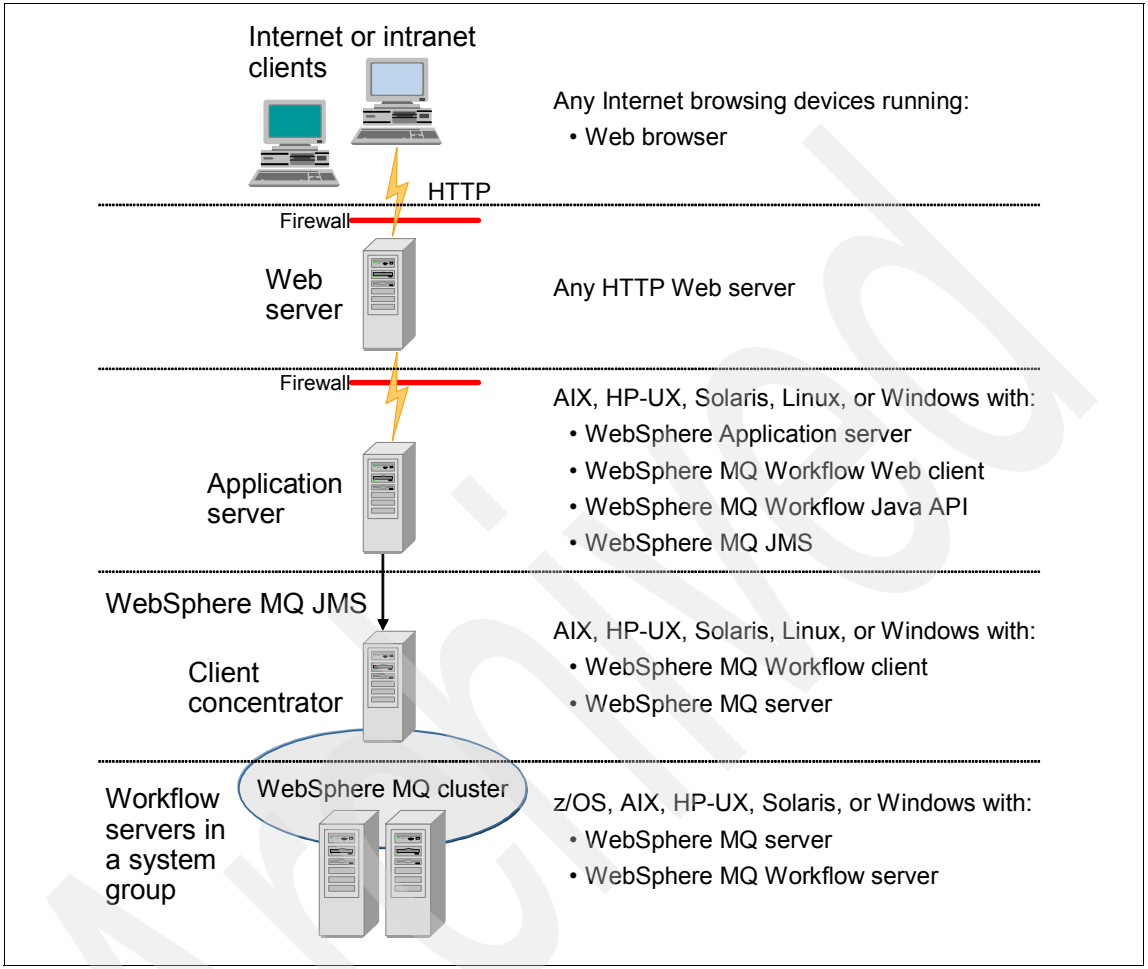


Figure 13-5 Web client in a WebSphere Application Server Network Deployment environment

Note: In the WebSphere Process Server target topology, this would most likely be implemented as a clustered WebSphere Process Server environment with remote Web clients that have been set up for high availability and workload distribution.

See 13.2.6, “WebSphere Process Server cluster patterns” on page 378 for more information about the available cluster topology options.

Portal-based client using the JAVA API

The portal-based client component also installs the Java API. The portal client must have access to a custom Java client configuration. The Java API can only use WebSphere MQ JMS to communicate with the WebSphere MQ Workflow system. The Java API cannot use any other communication interfaces, for example, JNDI. The portal client can use a WebSphere MQ client or a WebSphere MQ Server. Figure 13-6 shows a portal-based client setup.

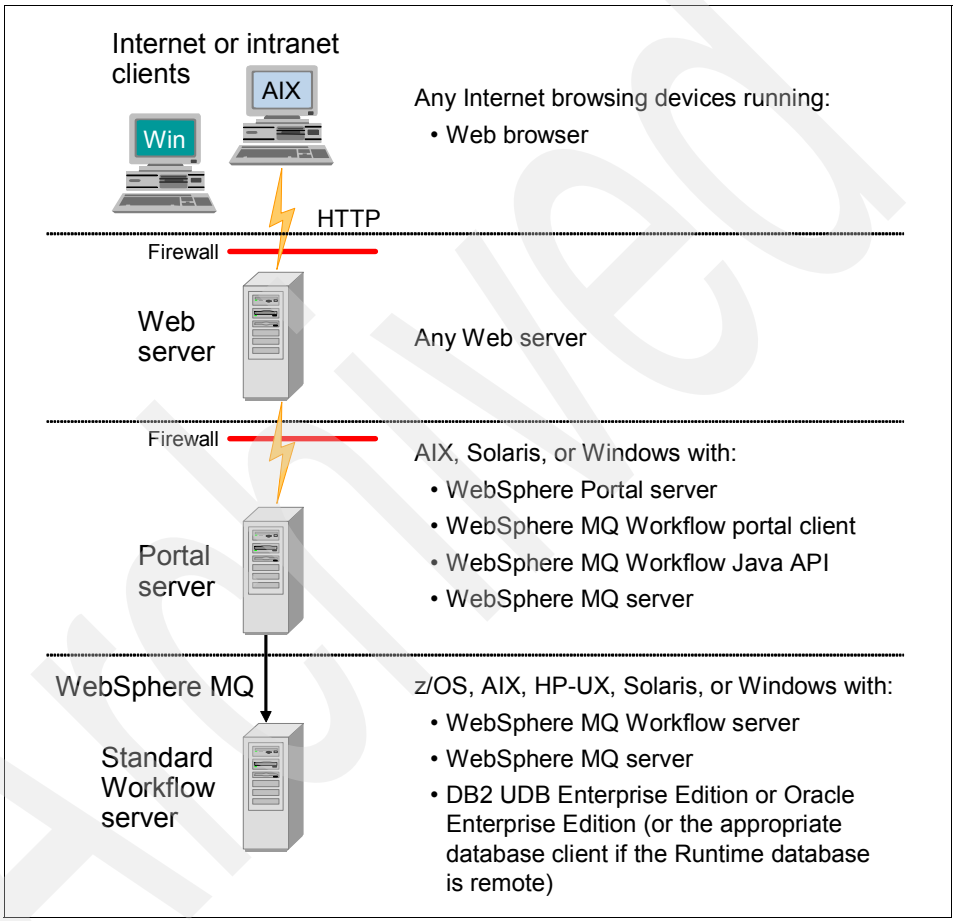


Figure 13-6 Example of the portal-based client using WebSphere MQ

Note: In the WebSphere Process Server target topology, this would most likely be implemented with WebSphere Portal and WebSphere Process Server. This could be either a WebSphere Portal with Business Process Choreographer in a single installation, or a WebSphere Portal server or cluster connecting to a WebSphere Process Server server or cluster.

Three basic configurations are available that support combining a portal and Business Process Choreographer:

- Portal without Business Process Choreographer runtime: This can be as a standalone server or in a portal cluster. In this case you install only the WebSphere Process Server client on the portal and support your business processes by connecting to a remote server. This way the portal server machine does not have the additional load of the process server.
- Portal with Business Process Choreographer runtime as a standalone server: You can use this for small installations or similar purposes.
- Portal with Business Process Choreographer runtime in a cluster: Use this configuration to support your scalability and availability requirements.

For more information about integrating portal and business processes, refer to the WebSphere Portal V6 information center:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp?topic=/com.ibm.wp.ent.doc/wps/bizproc_6012.html

13.2 The WebSphere Process Server topology

WebSphere Process Server leverages the infrastructure capabilities of WebSphere Application Server. The target topology for WebSphere Process Server therefore needs to be rethought in WebSphere Application Server concepts like cells, nodes, clusters, buses, and so on, instead of network, domain, and system, depending on the specific requirements.

This section describes these concepts and the topology options available. For information about transitioning a WebSphere MQ Workflow environment to one in WebSphere Process Server, refer to 13.6, “Best practices and recommendations” on page 398.

13.2.1 Stand-alone server and managed server

When you have installed WebSphere Process Server on server hardware, you can create profiles for one or more individual stand-alone servers, a deployment manager, or a managed server or group of servers.

A *stand-alone server* profile has its own administrative console and all sample applications (if you performed a complete installation or selected to install the sample applications gallery feature during a custom installation). Each stand-alone server is fully operational and is managed independently from all other servers. Application servers host zero or more J2EE applications.

There are two main types of stand-alone server installations:

- One stand-alone server on one hardware system
- Multiple stand-alone servers on a single installation on one hardware system

Note: A stand-alone server by default uses Derby as the persistent datastore. You can, however, modify this default and use a real database system.

We recommend a stand-alone server for development and small test environments, and for production setups that can live with the restriction of limited power, no scalability, performance limitations, and no high availability.

If the environment starts small but needs to scale, we recommend starting with a managed server or a network deployment cell.

A network deployment cell consists of one deployment manager and one or more managed application servers on federated nodes. To create a node, create a profile using the profile wizard, the **`manageprofiles`** command, or the Profile Management Tool.

A node can become a managed node in two ways:

- By federating a stand-alone profile
- By federating a custom profile into the cell

The deployment manager is the single point of administration for all of the managed nodes in the cell. It maintains the configuration files for nodes that it manages and deploys applications to those managed nodes.

The main reasons to use a network deployment cell rather than using a number of stand-alone servers are:

- The deployment manager provides a centralized administration for the cell.
- A managed server in a cell can scale to a cluster.

Note: If you started with a stand-alone server with Derby for the messaging database, promoting the server to a cluster is only possible if you do the following:

- Replace Derby with an external database (DB2, Oracle, and so on).
- Federate the stand-alone profile that contains the server.
- Create a cluster using the server as the first server in the cluster.

13.2.2 Cluster fundamentals

In WebSphere Process Server production environments, the system is expected to cater to the following requirements:

- Be failure tolerant.
- Allow for maintenance without the loss of data.
- Ensure scalability, for example, by adding processing capacity, to grow the environment to meet increased user demand.
- Ensure availability to always have the applications available to client applications.
- Support workload management.
- Provide for easy administration.

WebSphere Process Server clustering provides the ability to create logical groups of servers that cater for these requirements.

WebSphere Process Server exploits the functionality of WebSphere Network Deployment for distributed environments. Figure 13-7 shows the key elements of a WebSphere Network Deployment cell.

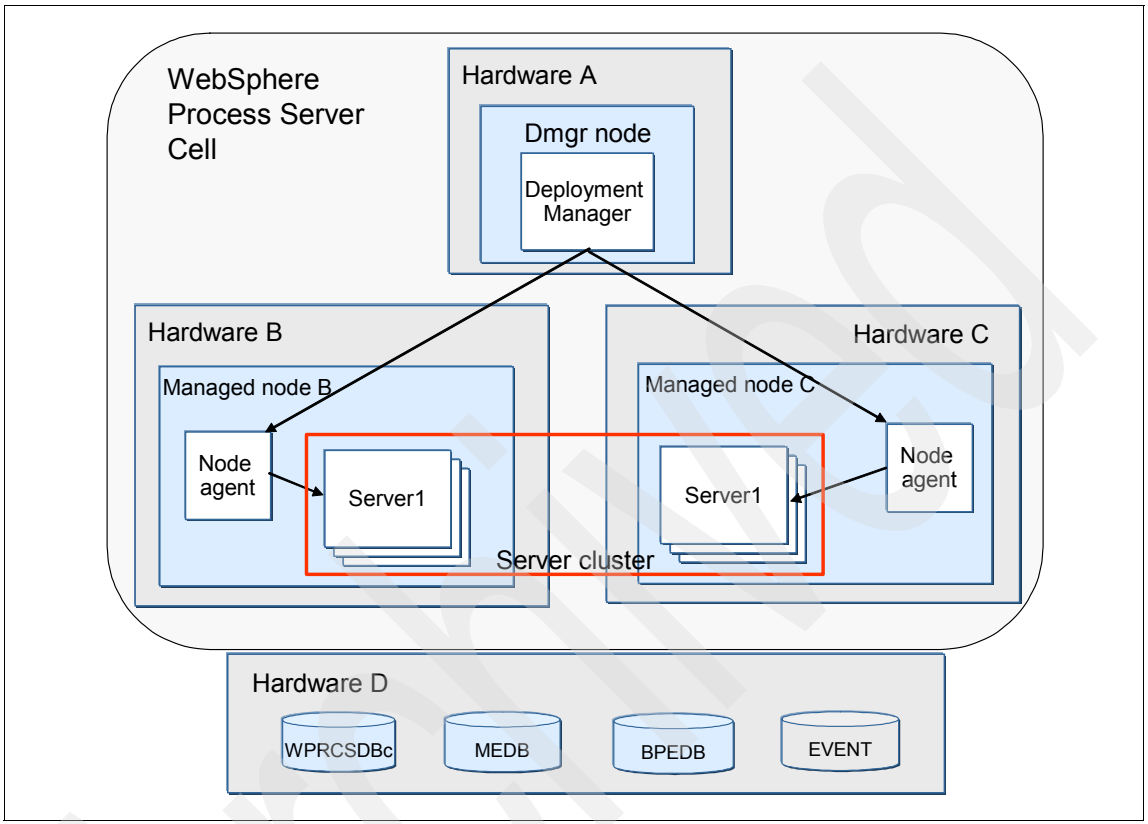


Figure 13-7 An example of a WebSphere Process Server cell

The deployment manager provides a single administrative interface to a logical group of servers on one or more machines. This logical group of servers is known as a cell. A deployment manager has an administrative console from which you can manage servers. Although the deployment manager is a type of server, you cannot deploy solutions to the deployment manager itself, and the deployment manager cannot have a sample applications gallery.

Each deployment manager has a cell into which custom profiles can be federated to create a managed node. From WebSphere Process Server V6.0.2 on you can also federate a standalone profile, provided that it is the first node in the deployment manager cell. A deployment manager manages the configuration for all of the managed nodes in its cell and deploys applications to servers or server

clusters in the cell. The deployment manager is defined by a deployment manager profile and has a First Steps console.

A WebSphere cell is a logical grouping of nodes that are centrally managed and have access to shared resources. Nodes within a cell typically run one or more application servers that each host one or more applications that are similar in terms of business requirements or non-functional requirements.

A *node* is a logical grouping of managed servers. A node usually corresponds to a logical or physical hardware machine with a distinct IP host address. Nodes can be on multiple machines or on the same one, but an individual node cannot span multiple machines.

A node agent is an administrative unit that represents a node to your system and manages the servers on that node. Node agents monitor various servers (application servers, Web servers, JMS servers) on a host system and route administrative requests to servers. A node agent is a server that is created automatically when a node is added to a cell. A node agent runs on every hardware system that participates in the network deployment environment. You can view information about a node agent, stop and start the processing of a node agent, stop and restart application servers on the node that is managed by the node agent, and so on. A node agent is purely an administrative agent and is not involved in application-serving functions. A node agent also hosts other important administrative functions, such as file transfer services, configuration synchronization, and performance monitoring.

Clusters are sets of managed servers that provide high availability and workload balancing for applications. Members of a cluster can be servers located on different host machines or servers located on the same hardware machine. On the same machine, they may be located on the same or different nodes. Clusters give your applications more capacity and availability than a single server.

Application servers host zero or more J2EE applications. An application server instance can be configured as a stand-alone application server with its own administrative console as a single application server that resides on a node belonging to a cell or as a member of a cluster.

Server configuration files define the available application servers, their configurations, and their contents. A cell-level repository stores configuration data for the entire cell and is managed by a file repository service that runs in the deployment manager. The nodes have a local copy of the relevant parts of the central repository of the deployment manager.

Figure 13-7 on page 371 shows a WebSphere Process Server cell with the following characteristics:

- The deployment manager is located on a machine of its own.
- The two node agents are located on two separate machines.

> **Note:** For production environments, we recommend putting the deployment manager on a different machine from those hosting the application servers. This ensures that the administrative console is still available should a server machine go down. Consider whether you must make the deployment manager highly available. If it goes down, the cell continues to be operational. However, the administrative interface will no longer be available.
>
> Consider distributing the node agents over several machines. Each node agent manages several identically configured servers. The servers are part of a cluster. If hardware C must be taken off-line for maintenance or fails, the servers on hardware B are still available to handle service requests.

13.2.3 Vertical and horizontal clusters

If the application servers in a server cluster are located on the same machine, this is called vertical scaling. Figure 13-8 shows a vertical cluster.

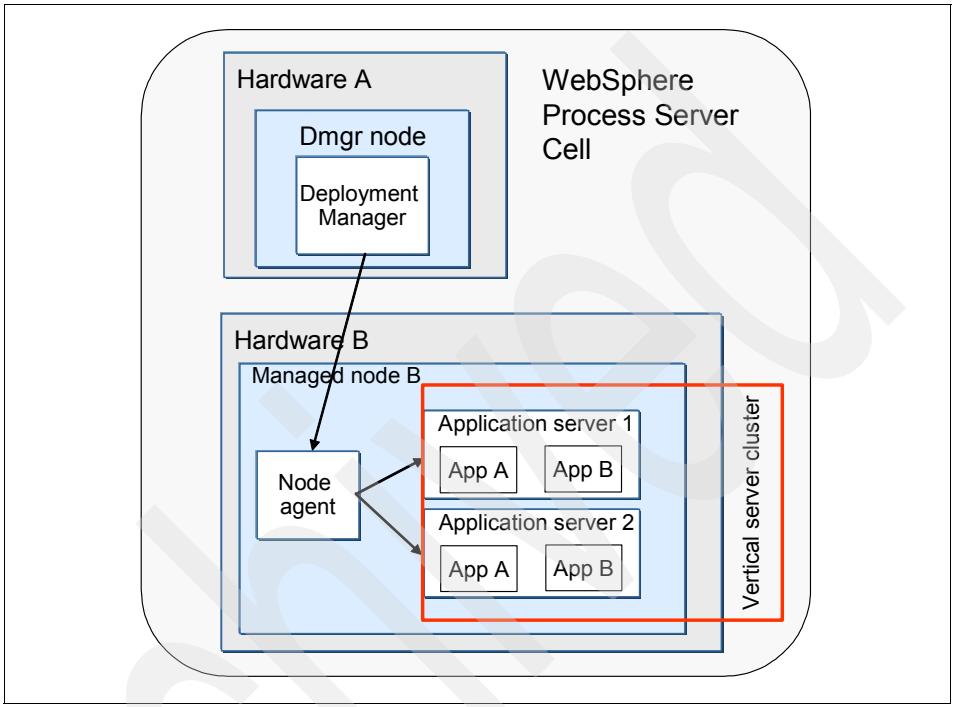


Figure 13-8 A vertical cluster

A vertical cluster can be used to better utilize the available resources. Vertical clusters do not provide for continuous availability in the event that the hardware hosting the members' node fails.

If the application servers in a server cluster are located on different machines, this is called *horizontal scaling*, as shown Figure 13-9.

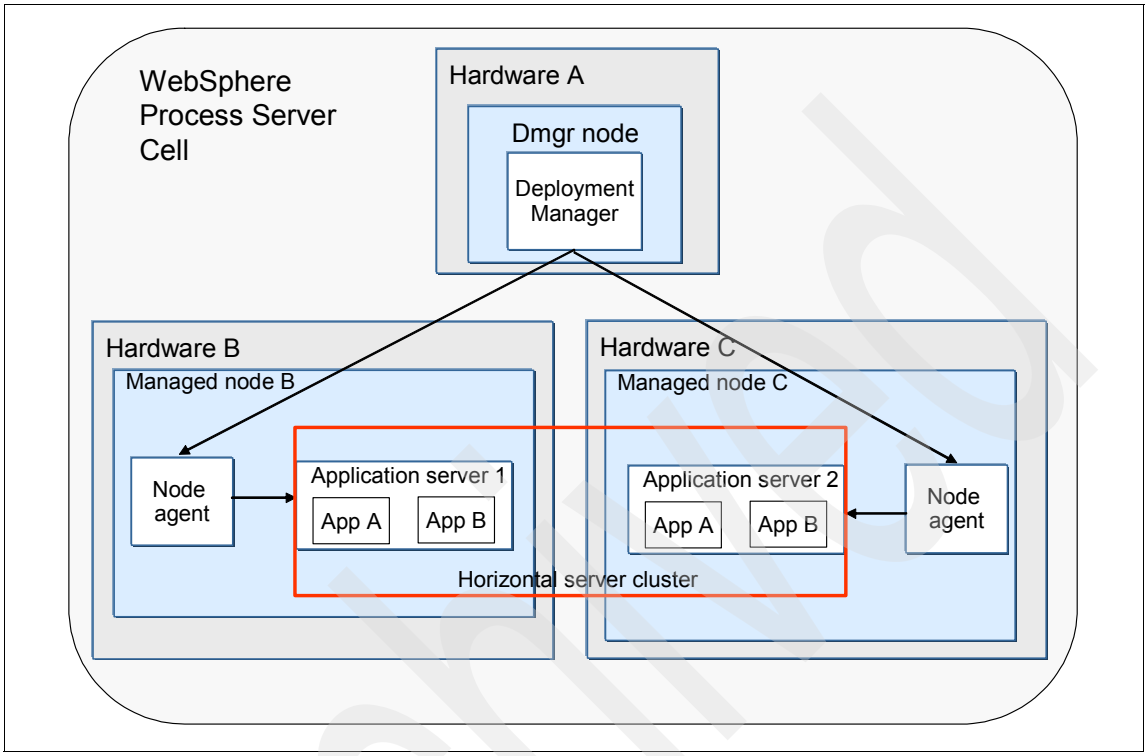


Figure 13-9 A horizontal cluster

In a horizontal cluster, multiple application servers are distributed across nodes on different machines in order to utilize more physical resource. Horizontal clusters can increase throughput and provide service availability in case a cluster member fails due to an application fault or if the hardware of this member's node fails.

To scale a clustered environment, one or more machines can be brought online, new nodes can be federated into the cell, and more servers can be added to the cluster.

13.2.4 WebSphere Process Server fundamentals

Although service integration buses and messaging are a part of WebSphere Application Server Network Deployment V6, they must be discussed briefly because they determine the topology options available for WebSphere Process Server.

A service integration bus (SI bus) is a form of managed communication that supports service integration through messaging. A bus consists of interconnecting messaging engines that manage bus resources. The members of a service integration bus are the application servers and clusters on which the messaging engines are defined.

A service integration bus comprises a SIB Service, which is available on each application server in the WebSphere Application Server environment. The members of a service integration bus are the application servers and the server clusters within which messaging engines for that bus can run.

A messaging engine is a server component that provides the core messaging functionality of a service integration bus. A messaging engine manages bus resources and provides a connection point for applications.

Each messaging engine is associated with a server or a server cluster that has been added as a member of a bus. When you add an application server or a server cluster as a bus member, a messaging engine is automatically created for this new member. If you add the same server as a member of multiple buses, the server is associated with multiple messaging engines (one messaging engine for each bus). If (and only if) this bus member is a cluster, then you can create additional messaging engines for that bus member. This means that a cluster as bus member can have one or multiple messaging engines that manage destinations localized to that bus member.

Each messaging engine has its own message store used, for example, to store persistent messages and maintain durable subscriptions. By default, a messaging engine associated with a server is configured with a file-based message store. In other cases, you are asked to provide the Java Naming and Directory Interface (JNDI) name of a Java Database Connectivity (JDBC) data source for use by the messaging engine.

WebSphere Process Server includes enterprise service bus capabilities. It uses messaging for the Common Event Infrastructure (CEI), for Business Process Choreography, and for asynchronous SCA invocations. To understand the topology options available with WebSphere Process Server, a thorough understanding of clustering and messaging is required.

13.2.5 Messaging fundamentals

A messaging engine is a relatively lightweight runtime object. This allows a single application server to host several messaging engines, as shown in Figure 13-10. When an application server is added as a member of multiple buses, that application server is associated with multiple messaging engines, one messaging engine for each bus of which it is a member.

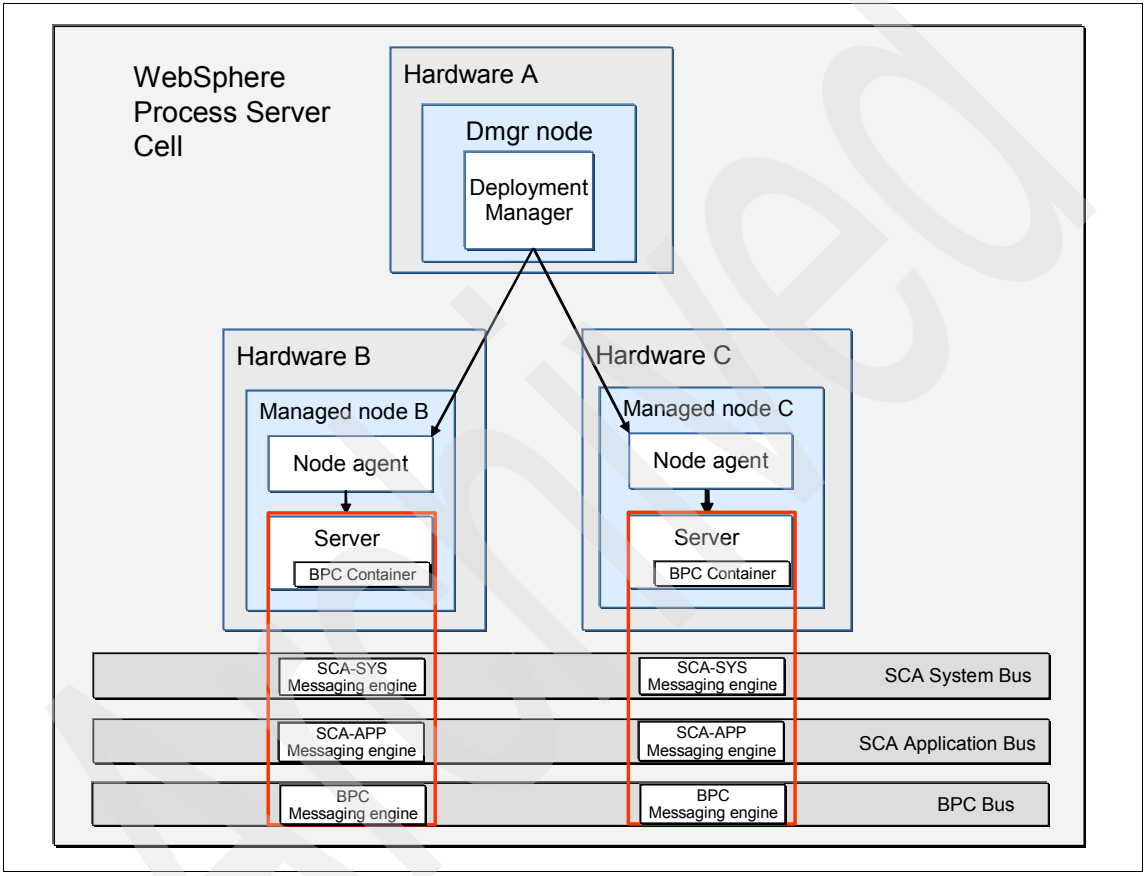


Figure 13-10 Messaging engines on a bus

When a cluster of application servers is added as a member of a bus, a single messaging engine is created and associated with the application server cluster automatically, regardless of the number of application servers that are defined as members of the cluster. At runtime, only one messaging engine is active. This messaging engine runs within a single application server within the cluster, as shown in Figure 13-10. The application server that is selected to host the active

messaging engine is the first cluster member to start. The remaining application servers host standby messaging engines.

13.2.6 WebSphere Process Server cluster patterns

Various topology patterns for production environments are available that offer varying levels of support for product components and quality of service. When designing a production topology for WebSphere Process Server, you can choose to build a topology that supports everything that this product has to offer, or you can build a smaller topology that supports just the product components and qualities of service that you need. This section briefly describes the three main production topology patterns available:

- Single cluster
- Remote messaging
- Remote messaging and remote support

For more information about these patterns see also:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.wps.610.doc/doc/cpln_topologypat.html

Single cluster pattern

Figure 13-11 shows a basic WebSphere Process Server topology. This topology consists of a single cluster into which the business process applications are deployed.

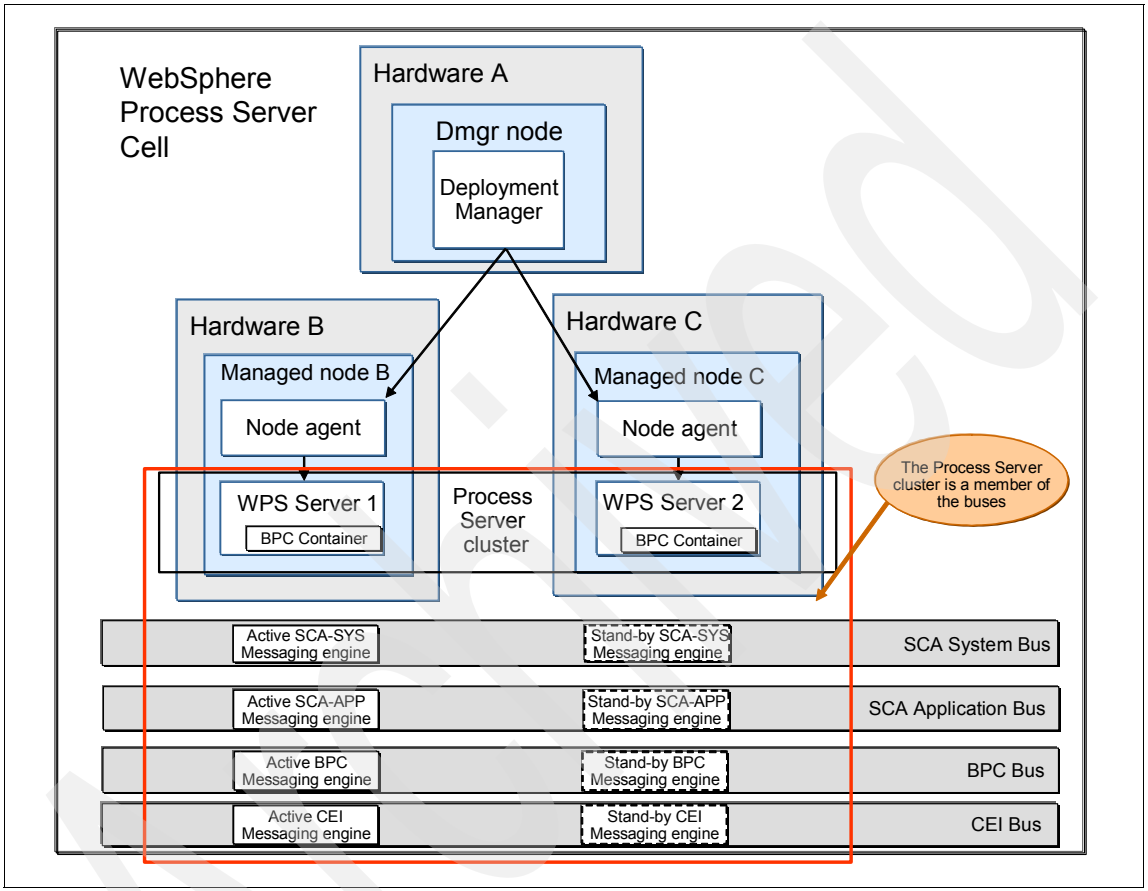


Figure 13-11 A single cluster pattern

The key architectural details of this topology are:

- The topology consists of one cluster in a cell. The cluster is the only member of the SCA-System, SCA-Application, BPC, and CEI buses.
- The Business Process Choreographer container is configured on this WebSphere Process Server cluster, and the cluster is a member of the SCA-System, SCA-Application, BPC, and CEI buses. Business applications and messaging engines are located in the same application server within the

cluster. Therefore, there is no clear distinction between the messaging layer and the application layer.

- There is only one active messaging engine on each of the buses. By default, this active messaging engine starts on one of the application servers inside the cluster. The active messaging engine has a stand-by messaging engine on each of the remaining application servers within the cluster. When the active messaging engine stops, the WebSphere HA manager activates one of the remaining stand-by messaging engines.
- Client applications that invoke business applications within the cluster can be either deployed into the same cell or into separate cells. In the sample topology, one WebSphere Process server hosts these client applications. These applications invoke the business applications within the cluster synchronously using SOAP/HTTP.

Note: The single cluster pattern is suitable for scenarios that are focused on running applications and on synchronous invocations. Any messaging requirements should be kept to a minimum with this pattern. Service Component Architecture internal asynchronous invocations, the Java Message Service, and MQ messaging bindings do not support multiple messaging engines in the same cluster. If your modules require any of these, choose one of the other patterns in which the messaging infrastructure is in a separate cluster from the application deployment target.

For business processes, the major drawback of this topology pattern is that MDBs are only driven on the server that hosts the active messaging engine. This has the consequence that long-running business processes are only navigated on one server in the cluster. We therefore recommend choosing one of the other patterns for business processes.

Remote messaging pattern

Figure 13-12 shows a remote messaging pattern for WebSphere Process Server. It provides a separate cluster for the messaging role.

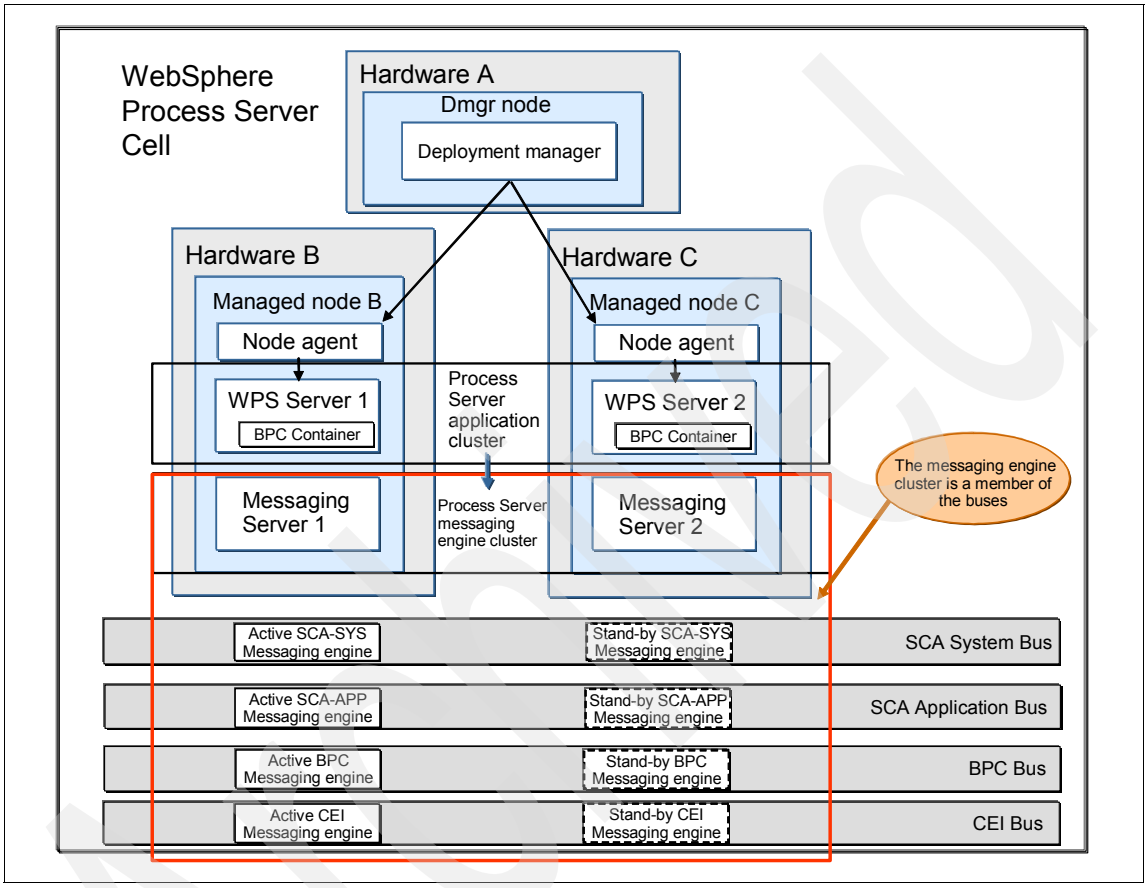


Figure 13-12 A sample remote messaging pattern

The key architectural details of this topology are:

- The topology consists of one messaging engine cluster and one or more application clusters.
- The messaging engine cluster is the only member of the buses.
- Business applications and messaging engines reside in separate application servers within the cluster. Therefore, there is a clear distinction between the messaging layer and the application layer.
- There is only one active messaging engine on each bus. This messaging engine is located within the messaging engine cluster. By default, an active

messaging engine starts on one of the application servers inside the messaging engine cluster.

- The active messaging engine has stand-by messaging engines on each of the other application servers within the messaging engine cluster. When an active messaging engine stops, the WebSphere HA manager activates one of the remaining stand-by messaging engines.

Note: This pattern is suitable for scenarios involving asynchronous invocations and for business processes.

Note that in Figure 13-12 the members of the WebSphere Process Server application server cluster and the members of the messaging engine cluster are co-located on the same hardware. For production environments that require higher scalability and availability, we recommend using separate hardware for the messaging cluster servers.

Remote messaging and remote support pattern

With this three-cluster pattern, resources are allocated to the cluster that handles the highest loads. This pattern is the most flexible and versatile, and is preferred by most users. The components are divided between the three clusters. See Figure 13-13.

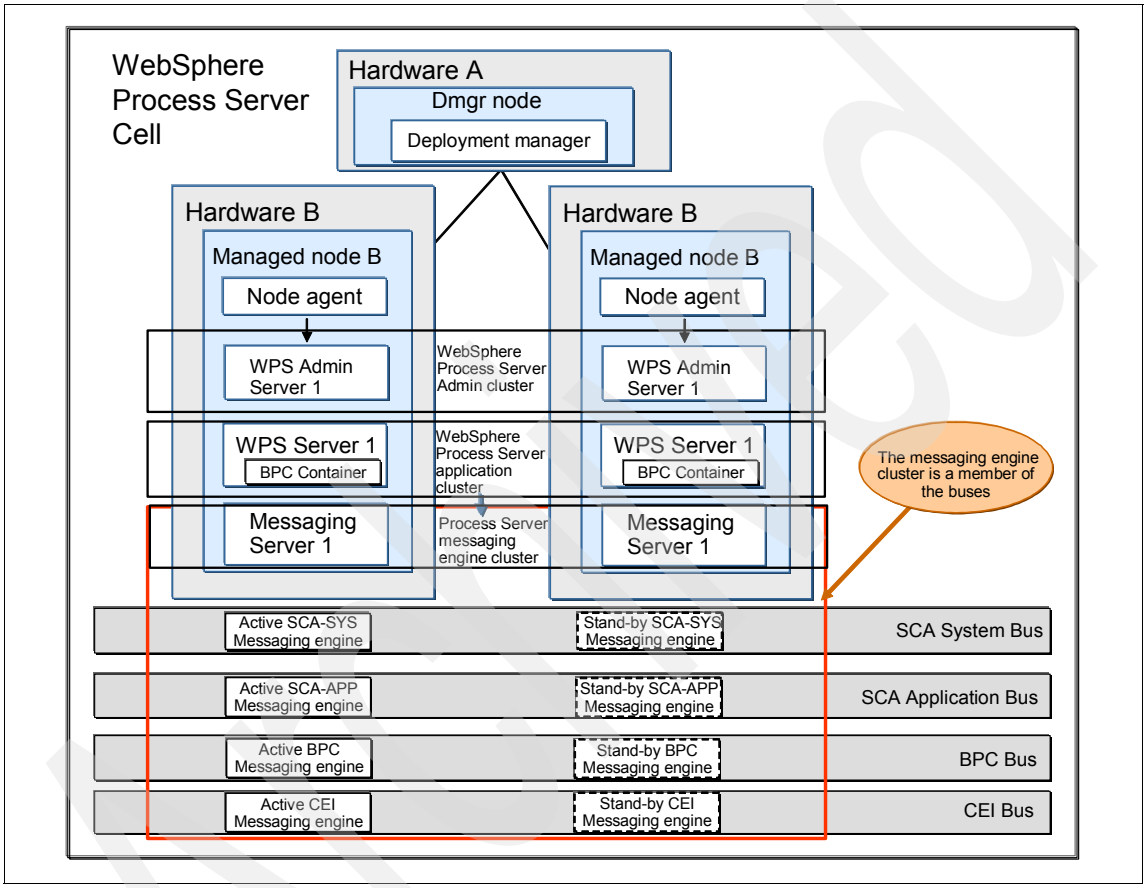


Figure 13-13 A remote messaging and remote support pattern

The key internal architectural details of the topology are:

- The topology consists of one messaging engine cluster, one support cluster, and one or more application clusters.
- The messaging engine cluster includes one or more WebSphere Application Server members. A single member hosts the active messaging engines for each of the buses. The messaging engine cluster is the only member of the buses. There is only one active messaging engine on each bus. This messaging engine is located within the messaging engine cluster:
  - By default, an active messaging engine starts on one of the application servers inside the messaging engine cluster.
  - The active messaging engine has stand-by messaging engines on each of the other application servers within the messaging engine cluster. When an active messaging engine stops, the WebSphere HA manager activates one of the remaining stand-by messaging engines.
- The support cluster includes the CEI-related and business rules manager-related applications, as well as Business Process Choreographer components such as the explorer and observer.
- Business applications and messaging engines reside in separate application servers within the cluster. In Figure 13-13 on page 383 there is one WebSphere Process Server application server cluster. The Business Process Choreographer and Human Task Manager (HTM) containers are configured within this cluster, and business processes are deployed within this cluster.

Note: This pattern is suitable for scenarios involving asynchronous invocations and for business processes.

Note that in Figure 13-13 the members of the WebSphere Process Server application server cluster, the members of the messaging engine cluster, and the members of the administration cluster are co-located on the same hardware. For production environments that require higher scalability and availability, we recommend using separate hardware for (at least) the messaging cluster servers.

13.2.7 Databases

A WebSphere Process Server topology requires as many as five types of database table sets, as shown in Table 13-2.

Table 13-2 WebSphere Process Server databases

Databases or tables	Default name	Usage
Common	WPRCSDB	► One per cell or standalone server. ► Used for relationships, failed events, business rules. ► Accessible by dmgr, all application deployment targets, business rules manager.
Event	EVENT, CEIDB	► One per CEI server installation. ► Used for storage of business events (may grow large quickly). ► Accessible by dmgr, CEI server.
Enterprise Service Bus Logging Mediation Primitives	EsbLogMedDB	► One per cell. ► Used for storing WebSphere Enterprise Service Bus mediation logging primitives. ► Accessible by all deployment targets of the mediation module.
Messaging	MEDB	► One per messaging engine (active). ► Used for storage of messages. ► Accessible by all potential messaging engines (active and standby).
Business Process Choreographer	BPEDB	► One per application deployment target. ► Used for macroflows and business processes and human tasks. ► Accessible by the local single server or the members of the local cluster.
Business Process Choreographer Observer	BPEDB	► For production, use a separate database. ► Same deployment target as CEI server.

Note: The following needs to be considered when planning the database setup for a production environment:

- The databases are used by WebSphere common functions, Business Process Choreographer, relationships, selectors, business rules, the Common Event Infrastructure, and possibly user applications.
- Databases can be remote or local. Local databases are suitable for small-scale single-server installations and development setups. For test and production environments, we recommend that databases be set up remotely on a database server. They must be created before starting the servers.
- Depending on the size and expected scaling of the environment, the number of databases may vary. Tables for several components may be created within the same database. For example, the EVENT and ESBLogMedDB could both be created in the WPRCSDB.
- Separation of concerns: A key element of WebSphere Process Server applications is messaging. For reliable messaging in production environments, separate the message engine cluster from the application cluster. This means using a separate messaging database for all messaging engines and needs to be defined at cluster setup time.

13.2.8 Business rules, selectors, and relationships

When setting up a WebSphere Process Server cell, the use of other SCA components may impact the design of the cell. You may need to consider the following:

- Business rules, selectors, and relationships make use of the common database WPRCSDB.
- SCA component modules are deployed as J2EE ear files.
- SCA modules can only be deployed once in a cell.
- For global entities, the following must be considered:
  - Uninstallation does not remove potentially shared objects from the database.
  - Unique names and namespaces are required across the cell.
- WebSphere Adapters that use inbound traffic can only be deployed once in a cell.

13.3 High availability, workload management, and scalability in WebSphere Process Server

Planning a WebSphere Process Server environment includes focusing on the operational properties of availability, scalability, and workload management. This section discusses some of the issues, trade-offs, and technologies involved in addressing each of these focus areas at planning time.

13.3.1 Availability

Availability covers the aspects of high availability and continuous availability. High availability covers unplanned events where components or an entire environment experience an outage.

In many cases, high availability is defined in these terms:

- MTBF: Mean time between failures.
- MTTR: Maximum time to recovery.
- Availability: MTBF/(MTBF+MTTR).
- The smaller the MTTR, the higher the availability.
- The larger the MTBF, the smaller the impact of MTTR.

However, in a WebSphere Process Server topology, there are many different components, and it is difficult to define a reliable value for MTBF. Another way of defining availability is using the *nines* rule:

- 99%: 3.7 days/year of down time (1h42' a week)
- 99.9%: 8h48'/year of down time (10' a day)
- 99.99%: 52'30"/year of down time (1' a day)
- 99.999%: 5'18" a year (half an hour every 5 years)

Continuous availability covers planned events such as application and infrastructure upgrades, hardware replacement or relocation, creation of backups, and so on. It is often referred to as 24x7 service and needs to cater for a continuous availability flow during planned events such as servicing a member:

- Take member off line (stop routing new work).
- Let in-flight work complete.
- Apply service to the member.
- Start the member.
- Put the member online (start routing new work).

Clients of the applications are not impacted by the actions, and there is no perception of operational failures.

To ensure overall availability, a number of WebSphere Process Server components must be made highly available:

- Application server Java Virtual Machines, which run:
  - WebSphere Process Server containers (SCA runtime, BPEL engine, human tasks, Common Event Infrastructure, and so on)
  - User applications (BPEL processes, integration logic, and so on)
- Messaging infrastructure, needed by SCA, Business Process Choreographer, and human tasks
- Databases, needed by SCA, Business Process Choreographer, human tasks, messaging engines, Common Event Infrastructure, and so on
- Other components: LDAP servers, Web Servers, networking software and hardware, security infrastructure, and so on

Techniques for achieving high availability

An overall concept of high availability must be considered at all levels, from the operating system to the application environment. On an operating level, IP takeover redundancy must be ensured.

Solutions such as High Availability Cluster Multi-Processing (HACMP) for AIX, Tivoli System Automation, Veritas Cluster Server, and Sun Cluster are available for IP takeover. Typically, these solutions adopt an active/stand-by approach.

Failover requires anywhere from one to several minutes. These techniques are not covered in this book.

At the application server level, high availability is ensured with the following:

- WebSphere clustering and WebSphere Application Server HA Manager: Allows for active/active and active/stand-by approaches.
- Transaction manager: Can use an active/active policy.
- Messaging engines: Use an active/stand-by policy.
- WebSphere Process Server databases: Data replication.

Failover is very fast, usually only a few seconds. These topics are briefly discussed in the following sections.

13.3.2 WebSphere Process Server high availability using clustering

At the application level, a WebSphere Process Server clustered setup can ensure high service availability. It provides increased workload capacity, improved resource utilization, and workload management. Refer to the sections

on clustering WebSphere Process Server earlier in this chapter for information about the clustering options available.

13.3.3 High availability of transactions

WebSphere Process Server uses the two-phase commit protocol for transactions. If a failure occurs between the first and the second phase of a two-phase commit protocol, the transaction is moved to the in-doubt state:

- The transaction is recovered when the transaction manager and resource manager come back up (re-synchronization).
- Re-synchronization requires an active transaction manager and the WebSphere Application Server transaction log file.
- Resource managers maintain any locks on their resources during that time.
- It is essential for a highly available environment that this recovery occurs as quickly as possible.

Note: By default, server transaction logs are stored locally and are not shared across cluster members. In a production environment that must be highly available, this is not a good idea. If one of the cluster members fails, in-doubt transaction recovery only occurs when (and if) the member comes back up.

Sharing the transaction log file is therefore critical to ensure a quick recovery of in-doubt transactions. In-doubt transactions may retain lock on the databases. Those locks may quickly reduce the system concurrency and availability. The shared file system must be capable of handling and quickly releasing file locks. Network File System (NFS from Version 4 on), IBM General Parallel File System (GPFS™), and Windows Common Internet File System (CIFS) are valid options.

For more information about how to set up an environment that caters to transactional high availability, refer to the following information:

- IBM WebSphere Developer Technical Journal: Transactional high availability and deployment considerations in WebSphere Application Server V6:

  `http://www.ibm.com/developerworks/websphere/techjournal/0504_beaven/0504_beaven.html`

- Best Practice: WebSphere HTTP Plugin Failover for High Volume Web Sites:

  `http://www.ibm.com/developerworks/websphere/library/bestpractices/plugin_failover_hvws.html`

13.3.4 Messaging engine high availability

When a cluster of application servers is added as a member of bus, a single messaging engine is created, regardless of the number of application servers that are defined as members of the cluster. At runtime, only one messaging engine is active and runs on the first cluster member (server) to start. The remaining application servers host standby messaging engines.

If the messaging engine or the application server within which it is running should fail, one of the remaining standby messaging engines is activated. Therefore, adding an application server cluster as a member of a bus enables failover for messaging engines that are associated with that cluster.

Messaging engine clustering

To cater for high availability of the messaging engine, you must consider how many messaging engines to set up, and where to locate them.

Single versus multiple messaging engines:

- A *single* messaging engine, which is created automatically when you add the cluster as a member of the bus. It uses a One-of-N policy for high availability, resulting in a single instance of the messaging engine being active.

  In this case, there is only one physical repository for the messages associated with the destination. This scenario ensures availability. However, scalability can only be achieved by providing additional computing power to the server (essentially, by configuring the application server on more powerful hardware).

- *Multiple messaging engines* are active at the same time. After you have added the cluster to the bus, which results in the creation of the first messaging engine, you can also manually create additional messaging engines on that cluster for that bus. Each messaging engine operates with a One-of-N policy, but since there are multiple engines, you can now have multiple active instances. You can create your own high-availability policies to define where each active instance should run by default, and thus evenly distribute the active engines across the various cluster members.

The second option, however, implies that the destinations on the SI bus are partitioned. In other words, each instance of the ME controls and works with a portion of the entire queue. There is no longer a single physical repository of messages for a certain destination. Such a topology does ensure scalability and some degree of availability. However, there are three considerations that must be considered before deploying this topology:

- Since there is no longer a single physical repository of messages, these topologies are not always adequate when requirements such as preserving the sequencing of messages are to be enforced.
- Message consumers may be statically bound to one particular partition of the queue. If this is the case, you may end up with stranded messages that nobody is ready to consume if that particular consumer partition dies.
- These topologies are more complex to set up and administer than topologies where there is a single active messaging at one time (non-partitioned).

We therefore recommend using this topology only when it is proven that the messaging infrastructure is indeed the bottleneck of the solution.

Note: For WebSphere Process Server production environments that must provide for scalability, availability, and workload distribution, we recommend setting up the *remote messaging pattern* or the remote messaging and support pattern described in 13.2.6, "WebSphere Process Server cluster patterns" on page 378.

13.3.5 Process server databases high availability

Several features of WebSphere Process Server use databases:

- Business Process Choreographer and human tasks
- Relationship service
- Failed event manager
- Selectors
- Business rules
- Common Event Infrastructure
- Messaging engines
- Mediation log

Consistency is essential for transaction recovery. The transaction log, the messaging engine databases, and the BPEDB must be kept synchronized in order to fully recover individual state transitions of long-running processes.

The WebSphere Process Server database topology that needs to be considered for transaction recovery is:

- One WebSphere Process Server *common* database per cell (WPRCSDB).
- One and only one Business Process Engine database (BPEDB) per instance of the Business Process Choreographer.
- One database schema per messaging engine. The database contains a unique identifier, which is tied to the specific messaging engine.

For transactional recovery in WebSphere Process Server to work, consider the following:

- The databases and the transaction log must be kept in perfect synchronization for transaction recovery.
- If you plan to recover a database on a backup cell, only an exact copy of the original cell can be pointed to the recovered database. By exact copy, we mean a cell that has the same topological information (system names, node names, cell name, server names, IP addresses, and so on).

13.4 Monitoring and auditing

This section describes monitoring and auditing functions in WebSphere MQ Workflow, corresponding functionality in WebSphere Process Server, and conceptual information for transitioning from one to the other.

13.4.1 Process monitoring and auditing in WebSphere MQ Workflow

The audit trail is used for monitoring and auditing of processes in WebSphere MQ Workflow. Events can be configured to be saved in the audit trail. There are many event types that can be set to be written. You can also define whether to save the data in a database or a WebSphere MQ queue.

With SupportPac WA61 monitoring in WebSphere MQ Workflow V3.6 is introduced. WebSphere Business Monitor V6.1 is supported.

With this SupportPac it is possible to use WebSphere Business Monitor V6.1 for both WebSphere Process Server and WebSphere MQ Workflow. That is, the infrastructure for the WebSphere Business Monitor V6 has to be set up only once.

13.4.2 Process monitoring and auditing in WebSphere Process Server

Use either administration programs provided with WebSphere Process Server (Common Base Event Browser or Business Process Choreographer Observer) for basic monitoring or WebSphere Business Monitor for advanced process monitoring. WebSphere Business Monitor provides graphs, cost/impact analysis figures, business-oriented dashboards, and many other features.

Monitoring and auditing is based on Common Base Events (CBEs) carried by the Common Event Infrastructure (CEI). During development of WebSphere Process Server solutions (including services, business processes, and human tasks) events that WebSphere Process Server sends to the Common Event Infrastructure are modeled or defined. The WebSphere Business Monitor tooling Monitor Model Editor then reads the information, consolidates it, and provides its output to the user.

13.4.3 Transitioning to WebSphere Process Server

The FDL2BPEL conversion tool does not transition audit events settings. It is important to analyze events that are currently monitored and configure the corresponding events for the process server. Note that in addition to using the FDL2BPEL conversion tool, the user must define and configure monitoring with WebSphere Integration Developer or WebSphere Business Modeler.

13.4.4 Advice/best practice for WebSphere MQ Workflow/Monitor WA61

SupportPac WA61 introduces WebSphere MQ Workflow monitoring using WebSphere Business Monitor V6.1. It offers an end-to-end solution for modeling, runtime, and monitoring with WebSphere Business Modeler V4, WebSphere MQ Workflow V3.6, and WebSphere Business Monitor V6.1.

SupportPac WA61 does not offer an end-to-end solution but only enables WebSphere MQ Workflow to do so. For this solution WebSphere MQ Workflow must be customized to AUDIT_TO_MQ and *not* to AUDIT_TO_DB.

Figure 13-14 shows the overall system design, which illustrates the main components and their interactions.

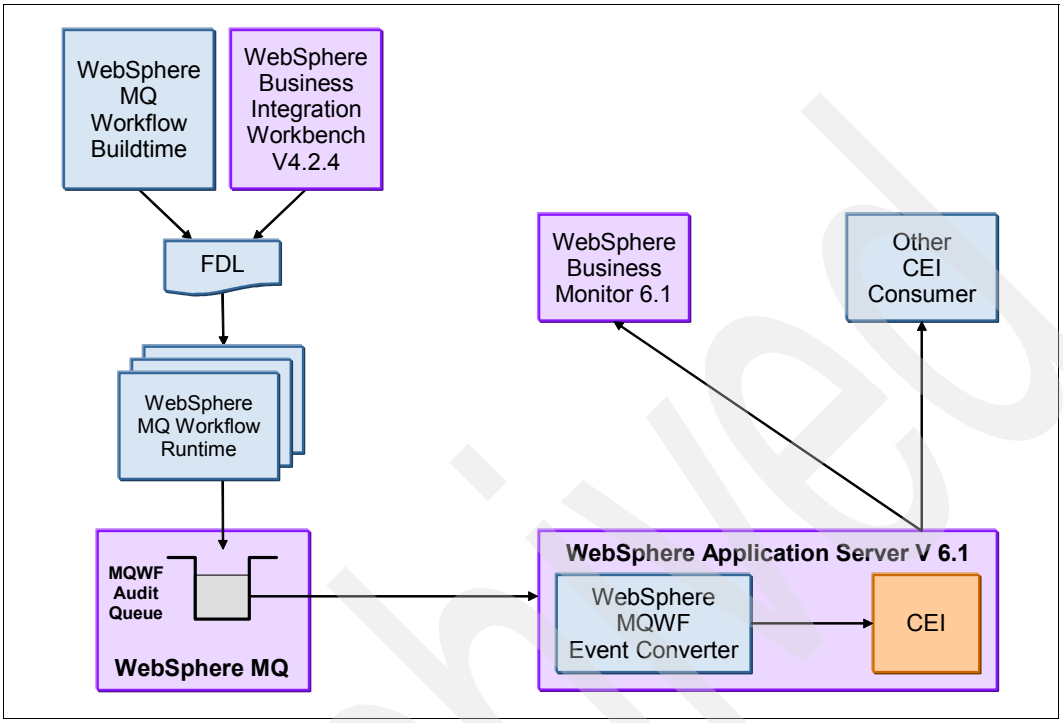


Figure 13-14 WebSphere MQ Workflow monitoring components

SupportPac WA61 converts WebSphere MQ Workflow audit events into Common Base Events and sends them to the Common Event Infrastructure. A J2EE application (WebSphere MQ Workflow Event Converter) is provided that receives WebSphere MQ Workflow audit events coming from the WebSphere MQ Workflow audit queue. It converts them into Common Base Events. These Common Base Events are transmitted to the Common Event Infrastructure, which is part of the WebSphere Application Server V6.1. These Common Base Events are in a format suitable for WebSphere Business Monitor V6.1.

13.4.5 Benefits

The main benefit of the new functionality released in WebSphere Business Monitor V6.1 is that it extends the reach of business application monitoring. The capability to monitor WebSphere MQ Workflow processes in WebSphere Business Monitor V6.1is important for the transition from WebSphere MQ Workflow to WebSphere Process Server as well. A common monitoring solution

simplifies completing already running process in WebSphere MQ Workflow while already starting new ones in WebSphere Process Server.

13.5 Cleanup of processes and tasks

During its life cycle a business process goes through several intermediate states until it finally reaches an end state. Depending on modeling options on processes and tasks, related objects are cleaned up automatically (physical deletion) or remain in the Runtime database for a specified or unlimited duration.

However, stored process instance data and related data, especially that with unlimited duration, not only impacts disk space but also performance. Therefore, you should regularly delete process-instance-related data from the database.

The following sections explain:

- Process and task-related cleanup modeling options
- Different cleanup possibilities

13.5.1 Modeling options

In this section we discuss modeling options.

WebSphere MQ Workflow

The values of the FDL options IMMEDIATE_CLEANUP and KEEP_PROCESSES determine how process instances and work items are deleted. For a transition from WebSphere MQ Workflow to WebSphere Process Server you may need to analyze the WebSphere MQ Workflow-based process model for the above-mentions settings. More details about these modeling settings and the cleanup mechanism can be found in the “Cleanup server,” section of the *IBM WebSphere MQ Workflow Administration Guide Version 3.6.*

WebSphere Process Server

Cleanup policies should be modeled on a process level and a task level.

To do the modeling cleanup at the process level:

1. In the WebSphere Integration Developer business process editor on a process level, in the Properties view select the **Details** tab.
2. Select **Automatically delete the process after completion**.

This setting is only relevant for long-running processes. It determines how the Runtime environment deals with the process instance that is used by the process once it has run its course. You have the following options:

- Yes

  Choose this to delete the data associated with this instance of the process once it has completed. This setting removes the process instance, whether or not the process executed successfully.

- On successful completion

  In this case the data remains in the database when the process fails so that the problem can be traced and the process may be restarted by the process administrator, if required.

- No

  Choose this to not delete the data associated with this process once it has completed.

To do modeling at task level:

1. In the WebSphere Integration Developer Human task editor, in the Properties view select the **Duration** tab.
2. The cleanup policy is specified in a combination of the Duration until task is deleted and Auto deletion mode fields.
   - Duration until task is deleted

     Use this field to determine when the task is removed from the system once it is completed. You can select either of the following:

     - Immediate

       In this context, the task is deleted immediately after it is completed.

     - Never

       In this context, the task is never removed from the system and never expires.

– Auto deletion mode

This selection is only available when you choose to delete the task upon immediate completion of the task. Use this to decide whether to delete the task based on its outcome. You have the following options:

- On completion

  Choose this option to delete the task from the system once it is finished, whether or not it was successfully completed.

- On successful completion

  Choose this option to only delete the task when it has been successfully finished.

Note: The cleanup of stand-alone task instances is independent of the cleanup of process instances. Maintain both to ensure that both are cleaned up.

13.5.2 Further cleanup options

Business Process Choreographer offers different cleanup options. Because the cleanup of processes and tasks, especially for large amount of data, can affect system performance, it is best performed at times when the system utilization is low.

API-based deletion

Business Process Choreographer offers APIs to delete process instances and task instances. For further information see:

- Deleting process instances

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_admdfe.html

- Deleting task instances

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6task_admdfe.html

Administrative script-based deletion

Use an administrative script to selectively delete from the Business Process Choreographer database any top-level process instances that have reached an end state of finished, terminated, or failed and all their associated data (such as activity instances, child process instances, and inline task instances).

Note: This script only cleans up inline task instances. Stand-alone task instances are not covered.

For further information see the WebSphere Process Server information center topic on deleting completed process instances:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/tadmin_deleting_completed_process_instances.html

Process Cleanup Service for Business Process Choreographer

The process cleanup service for Business Process Choreographer takes advantage of the scheduler service and other extensions that come with WebSphere Process Server to regularly trigger cleanup actions for completed business processes.

For further information see the article Process Cleanup Service for Business Process Choreographer:

http://www.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&q1=process+cleanup+service&uid=swg27007816&loc=en_US&cs=utf-8&lang=en

Note: This cleanup service is published for WebSphere Process Server Version 6.0. However, the described concepts are also applicable for WebSphere Process Server Version 6.1.

Additional cleanup maintenance topics

In addition to the process instance and task instance cleanup it may be necessary to do additional cleanup maintenance. The topics are described at the WebSphere Process Server information center in “Using scripts to administer Business Process Choreographer:”

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/tadmin_scripts.html

13.6 Best practices and recommendations

Sorting through all of the deployment options that are available with WebSphere Process Server and WebSphere Enterprise Service Bus can be a daunting task. If a topology is selected that accommodates any possible application, one can end up with more configuration than required. However, if the environment is

customized so that it contains exactly what is needed, one may need to spend more effort to gain knowledge than can be afforded.

The following set of recommendation and best practices is provided to help with this task.

13.6.1 Planning for future considerations

Finding the topology that covers the current and future needs requires an analysis of application properties and of non-functional requirements. When deciding on the appropriate target server infrastructure in WebSphere Process Server, the following questions should be addressed:

- Is the same number of applications hosted on the new infrastructure or will new, additional applications be deployed now or in the future?
- Does the new architecture start small, but need to cater for future enhanced scalability, high availability, or workload balancing?
- Do processes in the new environment need to be reached by new external partners?
- Will the level of service be the same as that in the old environment, or is a higher one required?
- What type and granularity of monitoring and auditing must the new infrastructure support?
- What types of and how many clients will connect to the new infrastructure? Will new clients be added later (for example, a portal client)?
- In addition to the production environment, will corresponding test and staging environments be set up?

13.6.2 Considering application architecture specifics

Some topology decisions must be made early and are based on the architecture specifics of the new environment, such as:

- Module design: The deployment environment is impacted by the modules that will be deployed because specialized support is needed for each component type (in the modules) and for each interaction type (between components and between modules).
- Application types: A single WebSphere Process Server cell can support J2EE applications, Web applications, WebSphere Process Server applications, WebSphere Enterprise Service Bus applications, and WebSphere Portal applications. You may need to decide early on how many cells are required and how you will distribute the applications among the cells. Their distribution

should be based on the isolation needs of the applications, particularly if multi-tenancy needs to be ensured. You may also need to consider whether groups of applications must be managed together or are independent of each other.

- Impacts of deployment requirements on the application design: There may be some non-functional requirements that have an impact on the development of the application:
  - Late binding generally requires a level of indirection in the application (for example, looking up a JNDI name that is set up during deployment).
  - If the deployment target of an application is a cluster, this generally requires static data to be stored someplace besides in memory. For example, when the next order number is updated, every cluster member needs to see the same value.
  - Applications deployed in a cluster generally require stateless behavior. For example, there is no affinity requirement between a client and a specific cluster member.
- Impacts of licensing on the deployment: A separation of the deployment targets or other WebSphere Process Server or WebSphere Enterprise Service Bus components (CEI servers, messaging engines, and so forth) impacts the distribution and failover properties of the applications. When you select a pattern that separates these resources, you can either deploy them on the same machine or on different machines. However, the standard trade-off must be made between memory and the number of server processes (including deployment manager, cluster members, and node agents) that are running on that machine. In addition, you may need to consider the number of licenses required for a distributed deployment pattern.

13.6.3 Application deployment pattern considerations

The design and setup of a deployment environment is dependant on the content and interactions of the business module being deployed. The make-up of a business module impacts the deployment environment design. A mapping can be made between a specific deployment pattern and the business module properties of the following:

- Export types (entry points of the module)
- Connections to the exports and the interactions with the exports (the way the module is used)
- Component types and the interactions between components (the pieces that make up the module)
- Import types (the partners of the module)

- Connections to the imports and the interactions with the imports (the way the partners are used)
- Resources needed by the components, such as databases or JMS resources that are used directly in the application
- Production and consumption of business events, for example, whether business events are collected, if they are collected asynchronously or synchronously, and so forth
- Interactions that business administrators have with the business modules (need for accessing the business rules manager, must access the BPC explorer, and so forth)

Refer to the "WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern" developerWorks article for a detailed discussion about the influence that the deployment pattern can have on topology decisions:

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

13.6.4 Scalability considerations: From single server to cluster

You may be starting with a single-server environment. However, if this environment potentially needs to grow in the future, some considerations concerning the topology must be made right from the start. WebSphere Process Server offers various topologies, and each has its benefits and trade-offs.

For a single-server environment that needs to scale to a clustered environment at a later date, consider the following:

- Instead of configuring a stand-alone profile, configure a managed profile in a network deployment environment to provide the Runtime environment for the server. This ensures that the infrastructure does not have to be modified when moving to a clustered environment.
- Do not use Derby for any of the databases. Cloudscape cannot be used as a remote database, which is most likely going to be a requirement for a clustered environment.

13.6.5 Messaging engine considerations

The SI bus infrastructure caters for the following:

- An SI bus can have one or many members. The members can be servers or clusters, or both.
- If the bus member is a cluster, it can host one or many messaging engines on an SI bus. As soon as a member is assigned to a bus, a messaging engine is automatically created on this member.

When planning a clustered topology, the messaging engine setup must be evaluated for scalability, high availability, and workload management. The following must be considered:

- Can a single messaging engine satisfy all requests of the planned number of cluster members?

  For high-volume messaging, consider the remote messaging and remote support pattern, and place the servers that host the messaging engine on appropriate (possibly separate) hardware.
- What happens when a single messaging engine fails?

  Is the messaging engine hosted by a cluster with at least two members so that it can fail over to another server? If yes, is this server on the same hardware or separate hardware?
- Since the SI bus and the SCA and Business Process Choreographer runtimes are loosely coupled, what happens if additional messaging engines are created on the bus?

Answers to these questions depend on the WebSphere HA manager, the WebSphere Workload Management, and the specific WebSphere Process Server setup. For more information about the options available, refer to *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413:

`http://www.redbooks.ibm.com/abstracts/sg247413.html`

13.6.6 Defining and verifying the deployment topology

To define the deployment topology of the applications to be run in a WebSphere Process Server environment (cell and clusters):

1. List all applications to be deployed.
2. If requirements exist that applications must be installed in separate clusters because they must not share data for security or auditing reasons, group them accordingly.

3. Map them to the appropriate component of the deployment environment. The following components are available:
   - Support cluster for CEI and business rules
   - One or more application deployment target clusters
   - Messaging engine cluster or clusters
   - Managed server in the cell.
4. Define the database setup accordingly:
   - All data in one database: This option facilitates tasks such as database backup and maintenance. However, it may, for example, cause performance bottlenecks during peak times, and is generally not recommended for production.
   - Different databases for the different components:
     - Common database
     - Event database
     - Messaging database
     - One or more application databases

     For environments that are expected to scale or that experience peak performance times, we recommend this setup.
5. Map the environment components to physical hardware (one or more machines or LPARs). Each machine hosts one or more components.
6. Verify that the deployment topology can scale:
   - If a large number of new applications must be deployed in the future, does the topology cater for this easily?
   - Does the topology cater for a large increase in the number of users?
   - Is the database setup appropriate for considerably higher load in the future?
   - Can the messaging engine setup scale? Discuss the remote messaging pattern versus each server in a cluster hosting a local messaging engine.
   - Set up one cell or more? If applications must be completely separate in terms of databases, messaging, and infrastructure consider setting up more than one cell.
7. Verify that the topology is size-tolerant. For a topology to be tolerant of changes to the cluster size, client applications should require no notice or modification that the cluster has changed. A production environment that requires client processes to re-establish their connections to the cluster is said to be cluster size-intolerant.
8. Verify that the topology is location-tolerant. For a topology to be tolerant of changes to the location of cluster members, client applications should require

no notice or modification that the cluster has changed. A production environment that requires client processes to re-establish their connections to the cluster is said to be cluster location-intolerant.

Figure 13-15 from the IBM Redbooks publication *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413, provides a flow chart for selecting the appropriate topology.

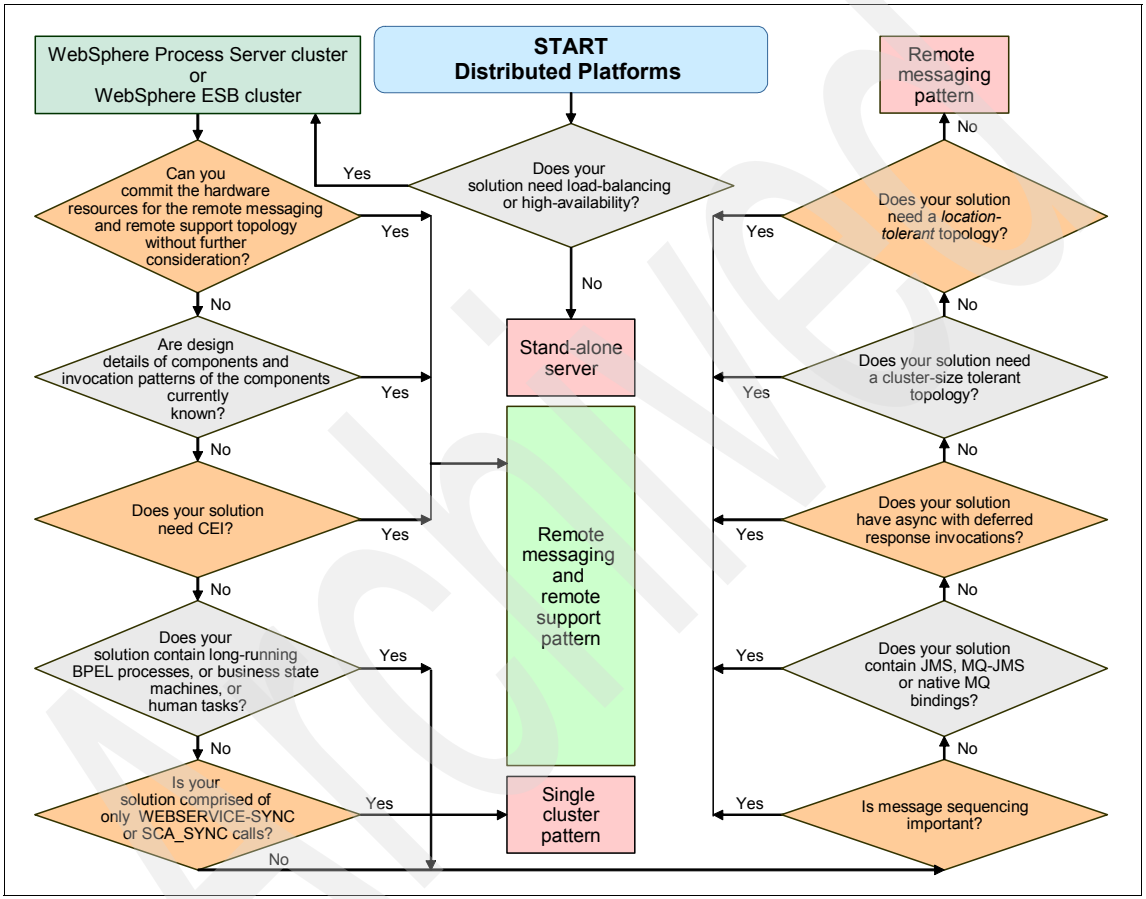


Figure 13-15 Flow chart for selecting the appropriate production topology for a business solution

For a detailed discussion of the topology options and on how to select the one most suited for your requirements, refer to the following information:

- IBM Redbooks publication *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413, available for download here:

  http://www.redbooks.ibm.com/abstracts/sg247413.html

- WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern:

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

- WebSphere Process Server Version 6.1 information center, section “Planning your deployment environment:”

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/doc/tins_inst_topologies.html

13.6.7 Cell administration considerations

A single WebSphere Process Server cell can support at least J2EE applications, Web applications, WebSphere Process Server applications, WebSphere Enterprise Service Bus applications, and WebSphere Portal applications. Consider the following when making the decision about how many cells you need and how to distribute applications among those cells:

- Your decision should be based on the isolation needs of the applications or their management, or based on geography.
- Multi-tenancy is required (the chosen topology is serving multiple client organizations (tenants) or applications while ensuring data segregation of the individual applications).
- Avoid co-locating test and production environments on the same system or cell.
- Carefully plan for software and application maintenance:
  - The topology should allow for staging software maintenance.
  - Decide early what down-time window is reasonable.

- Avoid concentrating multiple different environments in the same WebSphere Process Server cell. For example, preferably put WebSphere Portal and WebSphere Process Server each in its own cell. Advantages of this are:
  - Independent maintenance
  - Tuning
  - Administration
  - Greater independence of the level of the underlying software stack (WebSphere Application Server, Java version (JDK™), JDBC drivers)

13.6.8 Security considerations

Application security must be considered on multiple layers, not only for the application.

In order to secure your application on WebSphere Process Server consider and implement security on these levels:

- Hardware/software infrastructure and networking: Use the same principles as when securing your WebSphere MQ Workflow environment.
- Database: Use the same principles as when securing your WebSphere MQ Workflow environment.
- WebSphere Process Server: process choreography features (BPEL runtime, staff service, SCA infrastructure, and others) are implemented as WebSphere applications and thus adopt security mechanisms of WebSphere Application Server. WebSphere Application Server in general implements and extends Java 2 and J2EE security standards. Consider and configure security of these artifacts (all can be configured using the administration console):
  - User registry: WebSphere Process Server supports various user registries such as local operating system, Lightweight Directory Access Protocol (LDAP), custom registry, and federated repositories. You must transition the user registry from proprietary WebSphere MQ Workflow (recommended) or implement your own custom registry interface to connect to your legacy registry.
  - Transport layer security (Secure Socket Layer, SSL).
  - Java Authentication and Authorization Service (JAAS) for authentication and role-based authorization.
  - Application security: Securing application resources, mapping security roles to users and groups, and others. Securing of both Web and EJB applications.
  - Client security.

– Service Integration Bus security: JMS or WebSphere MQ security.

WebSphere Process Server provides these levels of security:

- Java security

  Java 2 security guards access to system resources such as file I/O, sockets, and properties. J2EE security guards access to Web resources such as servlets, JavaServerPages (JSP) files, and Enterprise JavaBeans™ (EJB) methods.

  WebSphere Process Server global security includes J2EE role-based authorization, the Common Secure Interoperability Version 2 (CSIv2) authentication protocol, and Secure Sockets Layer configuration.

- Global security

  This applies to all applications running in an environment and determines whether security is used at all, the type of registry against which authentication takes place, and other values, many of which act as defaults. The configuration of global security for a security domain involves configuring the common user registry and the authentication mechanism.

- Application security

  Secures various types of WebSphere applications, including Web applications, Web services, and many other types.

- Service integration bus security

  Asynchronous invocation in Service Component Architecture (SCA) is implemented using messages that are sent and received over the SI bus. The SI bus supports authentication for connecting to the bus and role-based access control for accessing the destinations, sending, receiving, and browsing messages.

  – WebSphere Process Server uses container-managed aliases to authenticate with the bus. These aliases are set up during installation. Default access control grants permissions to all authenticated users.

  – Business Process Choreographer uses application-managed resource authorization. This implies that you must set the component-managed authentication alias (not the container-managed) for authentication against the bus.

The following security requirements should be considered for a business integration solution:

- Authentication

  A mechanism that ensures the identity of users. Using this mechanism, the system can distinguish between valid users and invalid users of the system.

- Access control

  This ensures that authenticated users have necessary permissions to access only those resources or perform only those operations for which the system has granted the user's permission.

- Integrity

  This ensures that data that is sent over the network is not modified in transit.

- Privacy and confidentiality

  This ensures that data that is sent over the network cannot be viewed by unauthorized parties and that it remains confidential.

- Single sign-on

  When a client request needs to flow through multiple systems within the enterprise, the client should not have to authenticate several times. The client should be authenticated only once. The authenticated context is propagated to downstream systems that can apply access control.

For more information about WebSphere Process Server security, refer to the following resources:

- WebSphere Process Server V6.1 information center: Securing applications and their environment:

  `http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/doc/welcome_wps_sec.html`

- IBM redbook *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413, available for download at:

  `http://www.redbooks.ibm.com/abstracts/sg247413.html`

For more information about Java 2 security with WebSphere Process Server, refer to:

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/doc/csec_j2sec.html`

13.6.9 Performance considerations

There are no direct formulas that allow you to compare performance of an application running on WebSphere MQ Workflow to its performance on WebSphere Process Server.

This section provides hints and tips that help assess and prepare your WebSphere Process Server environment to run your applications with desired throughput and response times.

- The safest approach to size your WebSphere Process Server environment is to prepare and run performance tests on a proof of concept and derive your sizing based on its result. A proof of concept should include the following:
  - It should cover typical transactions provided by your application.
  - Testing environment and transaction distribution should be as close to the production environment as possible. For example, do not set up a stand-alone server test environment if your production environment will be a network deployment environment.
  - Your test will probably not simulate production to 100% since you usually cannot simulate all possible user interactions.
- A WebSphere Process Server configuration provides a complex number of configuration parameters that can affect performance, most of them not directly comparable to those of WebSphere MQ Workflow. You are therefore best advised to tune your environment according to performance tuning guidelines and configuration of WebSphere Process Server optimized during performance tests.
- Your workflow process can be modified for better performance if it is in accordance with business requirements.
- The following qualitative statements can be used to get indicative sizing:
  - Some artifacts perform better in WebSphere MQ Workflow, others in WebSphere Process Server, but always consider scenarios that are being compared.
  - Automated workflows:
    - Migrating to message-based macroflows with JMS bindings: WebSphere Process Server has less throughput.
    - Migrating to message-based macroflows with MQ bindings: WebSphere Process Server has about the same throughput.
  - Migrating to macroflows with synchronous invocations: WebSphere Process Server has better throughput.
  - Migrating to microflows: WebSphere Process Server has much higher throughput (factor 5 to 50).

  - Scalability of WebSphere Process Server is nearly linear.
- Use these hints when migrating:
  - Tune transaction boundaries in macroflows.
  - Avoid SOAP because it is a quite *heavy processing* protocol.
  - Do not carry too much data through the flow.
  - Avoid subprocess if possible.

13.6.10 Further reading

Sorting through all of the deployment options that are available with WebSphere Process Server and WebSphere Enterprise Service Bus can be a daunting task. If a topology is selected that accommodates any possible application, one can end up with more configuration than required. However, if the environment is customized so that it contains exactly what is needed, one may need to spend more effort to gain knowledge than can be afforded.

It is beyond the scope of this book to provide the detail of information required to set up any of the environments. A number of excellent publications is available that will help do this. We recommend the following resources:

- WebSphere Process Server Version 6.1 Infocenter: Planning your deployment environment

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/doc/tins_inst_topologies.html
- WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern

  http://www.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html
- WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 2: My first WebSphere Process Server cluster

  http://www.ibm.com/developerworks/websphere/library/techarticles/0701_carlson-neumann/0701_carlson-neumann.html
- IBM redbook *WebSphere Application Server Network Deployment V6: High Availability Solutions*

  http://www.redbooks.ibm.com/abstracts/sg246688.html
- IBM redbook *Production Topologies for WebSphere Process Server and WebSphere ESB V6*, SG24-7413

  http://www.redbooks.ibm.com/abstracts/sg247413.html

A

Transition planning worksheets

This appendix contains planning worksheets for assessment of:

- Business requirements
- General assessment of the WebSphere MQ Workflow environment
- Hardware
- Machines in the WebSphere MQ Workflow setup
- Security
- Administration
- Performance and availability
- Resources
- WebSphere Business Integration Adapters
- WebSphere Process Server topology selection flow chart

These worksheets provide a comprehensive list of questions that, when answered, lead to a good understanding of the current WebSphere MQ Workflow environment. This helps in determining the level of effort required to transition to WebSphere Process Server.

Note: The worksheets listed below as tables are available for download from Appendix B, “Additional material” on page 433.

Business requirements

This section discusses the business requirements of a transition project.

Table A-1 can be used to gather answers to the main questions concerning the business aspects of an overall transition project.

Table A-1 Business requirements

Question	Answer	Comments
1. What is motivating the business to the transition from WebSphere MQ Workflow to WebSphere Process Server? Rate the priority to transition to WebSphere Process Server (1 to 5, 1 being the highest priority).		
2. Do you have a transition plan?		
3. By what date must the transition be complete?		
4. Has a training and education plan been defined for each role?		Roles may include: Business analyst, integration developer, system administrator, and so on
5. What is the risk to the business if the transition cannot be completed by that date?		
6. Who are the users or consumers of the interfaces?		
7. What are the critical success factors for a successful transition?		
8. What are the important milestones for the transition?		
9. Are there any hardware upgrades that are considered part of this transition?		
10. Are there additional software upgrades that are considered part of this transition (for example, Siebel upgrade, DB2 upgrade, and so on)?		

Question	Answer	Comments
11. Can the transition be partitioned? (Do you plan to upgrade some of the interfaces, all at once, coexist in both WebSphere Process Server and WebSphere MQ Workflow, and so on?)		
12. What national languages are supported by the interfaces?		
13. Rate the priority in which to consolidate runtimes (1 to 5, 1 being the highest priority).		
14. Is the current system design documented? If yes, provide it.		
15. Is the current integration architecture documented? If yes, provide it.		
16. Are the current functional and non-functional requirements documented? If yes, provide it.		
17. Are the current qualities of service requirements documented? If yes, provide them.		
18. Are there interdependencies shared across WebSphere MQ Workflow processes? If yes, provide documentation.		In other words, do processes invoke other processes, or do processes depend on previous successful or unsuccessful completion of processes?

General assessment of the WebSphere MQ Workflow environment

This section discusses the information to gather for the environment currently in place in WebSphere MQ Workflow.

Table A-2 can be used to gather answers to the main questions about the existing implementation details.

Table A-2 General assessment of the WebSphere MQ Workflow environment

Question	Answer	Comments
1. How many WebSphere MQ Workflow applications exist today? How many individual processes exist for each?		
2. What version of WebSphere MQ Workflow is in production, test, staging, and development?		
3. Has there been a recent capacity assessment for current WebSphere MQ Workflow performance and server throughput?		SupportPac available WP01: WebSphere MQ Workflow - Performance estimates and capacity assessments: http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24006573&loc=en_US&cs=utf-8&lang=en
4. What are the key performance characteristics of theWebSphere MQ Workflow application?		
5. Describe user interface applications for human interaction with workflow. Is this a Web client (thin client) or custom *thick* client application?		
6. What dependencies exist among interfaces that would cause them to be transitioned together?		
7. Which interface patterns are used (UPES invocations, MQ, backend integration methods)?		
8. Identify client application programs.		

Question	Answer	Comments
9. Describe the characteristics of WebSphere MQ Workflow client applications. Is it a thin/Web client or thick/custom user interface?		
10. What are the durations of processes? Are long-lived business processes used? What are the longest and average durations of workflow processes?		
11. Which APIs are used to interact with WebSphere MQ Workflow runtime and in what language (Java, XML, COBOL, C++, and so on)?		
12. List any external Web services that your interfaces access.		
13. List any internal Web services that your interfaces access.		
14. How many interfaces are in production, test, staging, and development?		
15. How many WebSphere MQ Workflow processes are currently in production? In test? In staging? In development?		
16. What is the complexity of the processes identified above? (Low, medium, or high complexity?)		
17. Is WebSphere Business Monitor currently being used?		Follow-on question: If no, are monitoring capabilities desired in a future WebSphere Process Server architecture?
18. Are monitoring tools used based on the WebSphere MQ Workflow audit trail?		

Question	Answer	Comments
19. What are the average volumes for each workflow process?		
20. Based on the above information, can you estimate or provide the number of API interfaces for each?		
21. Based on the above information, can you estimate or provide the number of human tasks for each WebSphere MQ Workflow process?		
22. Are you using PEA constructs in your processes?		
23. Identify the number of activity implementation programs used.		
24. Is PES/UPES functionality used in your processes? If so, identify the implementations.		
25. How many instances per day for each process?		
26. What are the peak times for system usage (certain day/time/week /season)?		
27. What is the number of concurrent users per WebSphere MQ Workflow application?		
28. What is the average depth of user worklists?		
29. How is process maintenance (transferring, alerts, processes in errors, and so on) and updates handled?		
30. How are work items assigned to staff?		
31. Is LDAP being used for staffing?		

Question	Answer	Comments
32. Dynamic staffing usage: Are you using *absent* functionality for staffing model?		
33. Identify core WebSphere MQ Workflow processes.		
34. Development of WebSphere MQ Workflow processes: Are they performed in WebSphere MQ Workflow Buildtime or WBI Modeler v4.2.4?		Is the development environment a WebSphere MQ Workflow/Modeler setup that takes advantage of the round-tripping feature?

Hardware

This section lists the information to gather for the hardware currently in place in WebSphere MQ Workflow. Table A-3 can be used to gather answers to the main questions on the existing hardware details.

Table A-3 Hardware

Question	Answer	Comments
1. Which hardware is used in production, test, staging, and development?		
2. Which operating systems and release levels are in use in production, test, staging, and development?		Especially for z/OS customers
3. What are your release levels for: ► WebSphere MQ Workflow ► WebSphere Business Integration Server ► DB2 ► WebSphere MQ ► WebSphere Application Server?		
4. How many CPUs/LPARs are used for production, test, staging, and development? What speed are these CPUs?		

Question	Answer	Comments
5. What is the current CPU utilization? Specify MIPS/MSUs as relevant.		
6. How much RAM is available in production, test, staging, and development? or How much memory is anticipated to be allocated per LPAR?		
7. Does the infrastructure include a dedicated pre-production staging environment?		
8. Does the infrastructure include a dedicated performance testing environment?		
9. Does the current infrastructure currently support the WebSphere Process Server prerequisites?		

Machines in the WebSphere MQ Workflow setup

This section lists the information to gather for the machines currently in place in the WebSphere MQ Workflow setup. For each system group, make a copy of Table A-4.

Table A-4 Machines in the WebSphere MQ Workflow setup

MQ Workflow machine types in system group:	Host name			
Buildtime hosts[1]				
Buildtime database hosts[2]				
Runtime database host	[3]			
Workflow systems (servers[4] [9])		[5]	[5]	[5]
Web servers[6]				
Application servers with Web client[6]				
Portal server with portal client[7]				
Client concentrators[8] [9]				
Clients				
...				
...				
...				

Security

This section lists the information to gather for the security setup currently in place in WebSphere MQ Workflow. Table A-5 can be used to record the users, authentication, and security setup.

Table A-5 Security

Question	Answer	Comments
1. Describe the security requirements for WebSphere MQ Workflow process access, if any.		
2. How are users authenticated?		

3. How are user authorizations managed?		
4. Which mechanism is used for user authentication and authorization?		
5. Which third-party security mechanisms are used (if any)?		
6. Which network security measures are in place (for example, firewalls)?		

Administration

This section lists the information to gather for the overall administrative setup currently in place in WebSphere MQ Workflow. Table A-6 can be used to make note of the administrative items to consider for a transition project.

Table A-6 Administration

Question	Answer	Comments
1. Describe how WebSphere MQ Workflow processes are tested and approved for production release.		
2. Is Web-based System Monitor used?		
3. What custom code/frameworks are used for error handling?		
4. What custom code/frameworks are used for auditing?		
5. What custom code/frameworks are used for monitoring?		
6. What custom code/frameworks are used for deployment?		
7. What custom code/frameworks are used for promotion from one environment to another?		

Question	Answer	Comments
8. What custom code/frameworks are used for logging/tracing?		
9. How is log rotation and file system maintenance handled?		
10. What is the process for the production release of an interface?		
11. What if any source control is used (CVS, SourceSafe, and so on)?		
12. Can source control export all artifacts of an interface into one jar, for example, one jar file for one interface?		
13. Which development methodology is used (for example, iterative, waterfall, and so on)?		
14. How long is a typical development cycle?		
15. How frequently are releases of the interfaces delivered?		
16. Do existing interfaces need to coexist with newly released interfaces?		
17. Are any custom scripts used for administering the interfaces?		
18. How are suspended processpes (processes that are in error) administered?		
19. Describe the setup, usage, and configuration of the Workflow Scheduling Server.		
20. Describe the setup, usage, and configuration of the WebSphere MQ Workflow Cleanup Server.		

Question	Answer	Comments
21. Describe the setup, usage, and configuration of the WebSphere MQ WorkflowAdministration Server.		

Performance and availability

This section lists the information to gather for performance and availability currently in place in WebSphere MQ Workflow. Table A-7 can be used to record the implementation details.

Table A-7 Performance and availability

Question	Answer	Comments
1. Is high availability used? If so, describe the topology in detail, includingWebSphere MQClusters, client concentrators, or WebSphere Application Server clusters, if used.		
2. Are multiple Workflow engines used concurrently? If so, describe the configuration.		
3. What are the availability requirements/SLAs for the interfaces (24x7, 99.999%, and so on)?		
4.Describe the above in detail.		

5. How is failover provided?		
6. What is the current backup strategy when components fail (databases, WebSphere MQ Workflow, hardware, and so on)?		

Resources

This section lists the information to gather on the skill and experience level of the people currently working with WebSphere MQ Workflow and those who will be working with WebSphere Process Server. Use Table A-8 to gather this information.

Table A-8 Resources

Question	Answer	Comments
1. What is the experience level of the staff in WebSphere Application Server (1 to 5, 1 being the most experienced)?		
2. What is the experience level of the staff in J2EE (1 to 5, 1 being the most experienced)?		
3. What is the experience level of the staff in WebSphere Integration Developer (WID) (1 to 5, 1 being the most experienced)?		
4. What is the experience level of the staff in WebSphere Process Server (1 to 5, 1 being the most experienced)?		

5. What is the skill level of the administrators for the existing WebSphere MQ Workflow production environment (1 to 5, 1 being the most experienced)?		
6. Are business partners involved in the transition efforts?		
7. Are there dedicated administrators for the existing WebSphere MQ Workflow production environment?		
8. How many administrators support the existingWebSphere MQ Workflow production environment?		

WebSphere Business Integration Adapters

This section lists the information to gather for adapters, UPES implementations, and other integration components currently in place in WebSphere MQ Workflow. Use Table A-9 to note the integration components in place.

Table A-9 WebSphere Business Integration Adapters

Question	Answer	Comments
1. Which custom adapters are used?		
2. Which UPES/PES functionality is used?		
3. Which UPES programs have you implemented to which backend systems?		
4. Are you using WebSphere Message Broker for integration?		
5. Are you using WebSphere Interchange Server for integration?		
6. Are you using any MQ-based bridges or adapters?		

7. Which client-side application integrations must be considered for transition?		

WebSphere Process Server topology selection flow chart

This section describes how to select the most appropriate topology for the required business and technical needs. Figure A-1 can be used to make the decisions required to decide on the topology best suited for the customer's requirements.

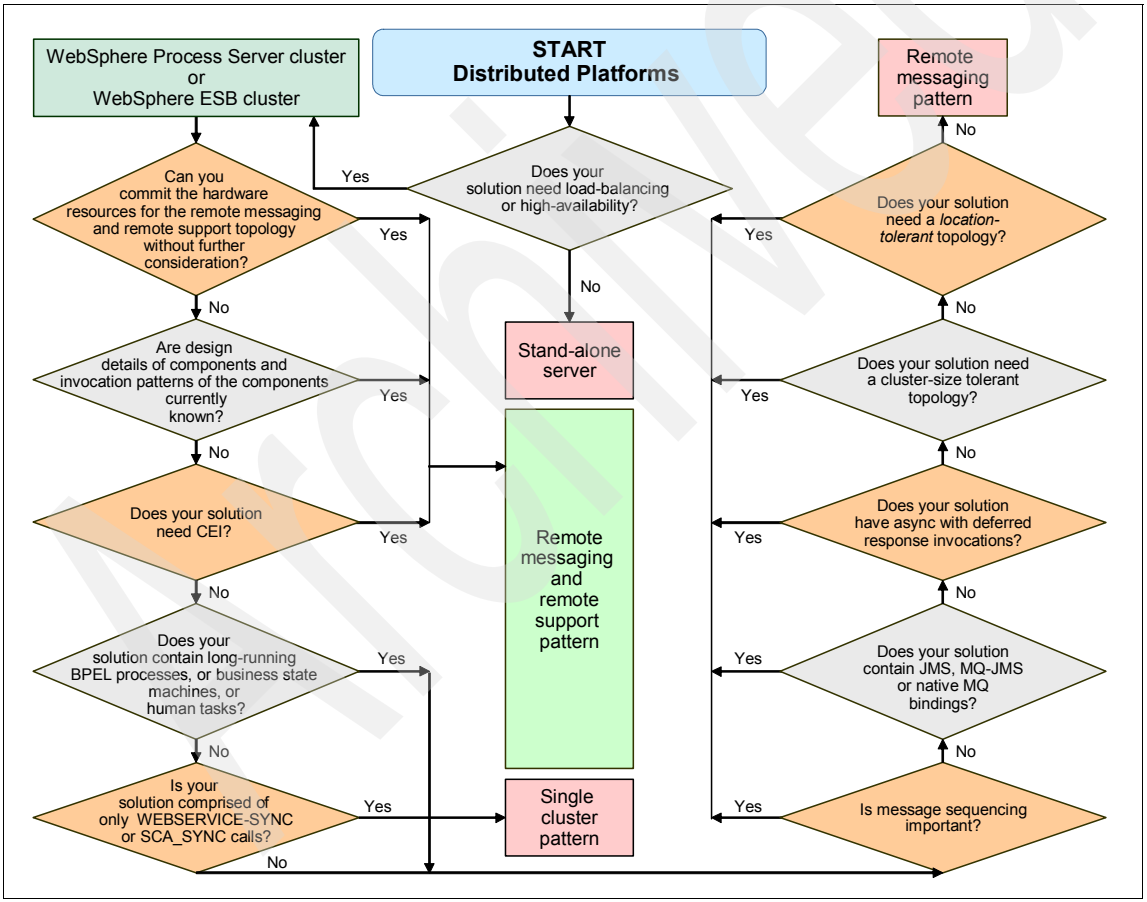


Figure A-1 WebSphere Process Server topology selection flow chart

UPES questionnaire

This section contains a check-list that can be used for transitioning user-defined program execution server (UPES) functionality in WebSphere MQ Workflow to WebSphere Process Server. It provides a comprehensive list of questions that, when answered, lead to a good understanding of the current WebSphere MQ Workflow UPES environment. This helps in determining the level of effort required to migrate to WebSphere Process Server.

Modeling aspects

This section lists the modeling aspects of the UPES implementation. Use Table A-10 to record the UPES modeling specifics in place.

Table A-10 Modeling aspects

Modeling	Customer input
UPES name selection mode (static or dynamic via container).	
UPES name qualifier (fully qualified name versus partly qualified name). Could give indication for clustering.	
Invocation mode: ► Synchronous: Request/response invocation (two-way operation) ► Asynchronous: Request-only invocation (one-way operation)	
Data mapping via: ► Data connector ► Data default connector ► Data loop connector	
UPES specifics: ► Description ► Version (for example, 3.3.0) ► System ► Queue name ► Queue Manager name ► Message format (XML versus JMS-compliant XML)	

Staff assignment aspects

The modeled staff assignment in WebSphere MQ Workflow in a UPES Program activity has two major aspects of assignment for administrative purposes (activity repair) and optionally, when modeled, a manual start and manual complete of the UPES Program activity. Use Table A-11 to record what is currently implemented.

Table A-11 Staff assignment aspects

Staff assignment	Customer input
Staff assignment criteria: ► In Buildtime: Program activity properties tab Staff1 and Staff2 ► In FDL: DONE_BY statement	
Start and complete criteria: ► Start manual or automatic ► Exit manual or automatic	

Data and context aspects

Which inbound and outbound data parts or other message elements are used in the invoked service/application? Use Table A-12 to record them.

Table A-12 Data and context information aspects

Data/context information	Customer input
ProgramInputData ► Custom data members only ► Predefined data members – Fixed data members (_ACTIVITY, _PROCESS, PROCESS_MODEL) – Process and activity data members (for example, _PROCESS_INFO.Role or _ACTIVITY_INFO.Duration)	
ProgramOutputDataDefaults ► Custom data members only ► Predefined data members – Fixed data members (_ACTIVITY, _PROCESS, PROCESS_MODEL) – Process and activity data members (for example, _PROCESS_INFO.Role or _ACTIVITY_INFO.Duration)	

Data/context information	Customer input
ProgramOutputData ► Custom data members only ► Predefined data members – Fixed data members (_ACTIVITY, _PROCESS, PROCESS_MODEL, _RC) – Process and activity data members (for example, _PROCESS_INFO.Role or _ACTIVITY_INFO.Duration)	
Starter	
ProgramID	
ExternalProcessContext	
ImplementationData	

UPES messages aspects

Which message types are evaluated? Use Table A-13 to record them.

Table A-13 UPES message aspects

UPES messages	Customer input
ActivityExpired	
TerminateProgram	
Encoding; Which encoding (code page) is used in the ActivityImplInvokeResponse message?	

UPES infrastructure aspects

Which infrastructure is used to read and write the UPES messages and to invoke the services? Use Table A-14 to record them.

Table A-14 UPES infrastructure aspects

UPES infrastructure	Customer input
UPES Framework as part of the WebSphere MQ Workflow product: Version, release, and service level	
Custom UPES framework	
WebSphere MQ Workflow ReplyToQ/ReplyToQMgr settings: ► Hardcoded settings ► Handled via MQMD fields	
Is the WebSphere MQ Workflow-recommended MessageID to CorrelID mapping within an UPES environment implemented? Note: It was not required for WebSphere MQ Workflow, but is required for WebSphere Process Server.	

Application purpose aspects

What is the invoked service/application doing? Use Table A-15 to record this information.

Table A-15 Application purpose aspects

Application purpose	Customer input
Semantic of the application (for example): ► Transactional back-end invocation ► Small calculation based on inbound data ► NoOp function ► Using WebSphere MQ Workflow API methods for process interaction ► And so on	

Error handling aspects

A service that was invoked by a UPES is able to return an ActivityImplInvokeResponse message with an exception, as shown in Example A-1.

Example: A-1 ActivityImplInvokeResponse

```
<ActivityImplInvokeResponse>
...
    <Exception>
     <Rc>1113</Rc>
     <Parameters>
       <Parameter>My problem was ...</Parameter>
       <Parameter><null/></Parameter>
     </Parameters>
     <MessageText></MessageText>
     <Origin>File(299)</Origin>
    </Exception>
...
```

To find out if such an exception pattern is used, check whether any of the error handling implementations described below are used. Use Table A-16 to record them.

Table A-16 Error handling aspects

Error handling	Customer input
In case of a custom UPES framework, check the code that generates the ActivityImplInvokeResponse message to see whether an <Exception> message part could be generated.	
In case of the WebSphere MQ Workflow UPES framework, check the custom code layer above the UPES framework to see whether that code raises an InvocationTargetException in case of a back-end error. Note that an InvocationTargetException forces the UPES framework to generate an ActivityImplInvokeResponse message with an Exception message part.	

Error handling	Customer input

In addition, check the WebSphere MQ Workflow system log for entries like:
`FMC12210E Could not execute program of activity UPES Activity ('RUEAAAABAD+ABQAAAAA . . .') in process UPES Test.`
`The error was: 1113`	

Customer requirement aspects

If available, check the customer requirements for a UPES framework and invoked services. Use Table A-17 to record the technical and business requirements and to verify that they match the implementation currently in place.

Table A-17 Customer requirement aspects

Customer requirement	Customer input

B

Additional material

This IBM Redbooks publication refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG247282

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG247282.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
All.zip	All zipped samples
Chapter 9.zip	Zipped samples used in Chapter 9, “Business Process conversion” on page 187
Appendix A.zip	Zipped worksheets used in Appendix A, “Transition planning worksheets” on page 411

How to use the Web material

Create a subdirectory (folder) on your workstation and unzip the contents of the Web material zip file into this folder.

Glossary

Administration console The IBM administrative console is the Web interface to the Application Server, Process Server or Enterprise Service Bus. It offers powerful options for configuration and deployment.

Administration server The administration server manages a WebSphere MQ Workflow system. It communicates with all other components in a system or system group. It is the working center of the administration component. It is responsible for the availability, operation, and error recovery of all server components. A command interface is used to send requests to the server's administration utility.

Administration utility The Administration Utility is the administrator's user interface to request services from the Administration Server. Using this utility, a WebSphere MQ Workflow system is started and stopped and the current state of any server can be listed. Furthermore, error logs can be accessed to check if any problems exist within the system.

API Application programming interface is a source code interface that a computer system or program library provides in order to support requests for services to be made of it by a computer program. The software that provides the functionality described by an API is said to be an implementation of the API. The API itself is abstract, in that it specifies an interface and does not get involved with implementation details.

Audit trail The audit trail is used for monitoring and auditing of processes in WebSphere MQ Workflow. Events can be configured to be saved in the audit trail. There are many event types that can be set to be written. You can specify whether to save the audit trail data in a database or a WebSphere MQ queue.

Binding Defines the kind of interaction as well as the caller and sender. It can be used to connect different SCA Modules together.

Binding Defines the kind of interaction as well as the caller and sender. It can be used to connect different SCA Modules together.

BO A Business Objects are representations of data structure used in business processes.

BPC Business Process Choreographer (BPC) is component of WebSphere Process Server runtime architecture and provides support for business processes and human tasks. It offers a way to model your business process based on the WS-BPEL specification, and to model interactions that involve humans, such as human-to-human, human-to-machine, and machine-to-human interactions. Both business processes and human tasks are exposed as services in a service-oriented architecture.

BPE Business Process Engine**.**

BPEL The Business Process Execution Language is a open standard to store process definitions in a file.

BPEL4WS Business Process Execution Language for Web Services.

Breakpoint Used as a point to stop the flow of a message in a message flow when the flow debugger is attached.

Broker A broker is a set of execution processes that host and run message flows.

Buildtime A WebSphere MQ Workflow component with a graphical user interface for creating and maintaining workflow models, administering resources, and the system network definitions.

Bus A bus provides the possibility to transfer data between different nodes connected to this bus.

Business Process A business Process defines a flow of actions to solve a special task. It can be stored in a BPEL file.

CBE Common Base Event specification defines a new mechanism for managing events in business enterprise applications and how to communicate self-healing events in the Autonomic computing model.
A small event can change things far beyond the seeming initial circumstance. Nowhere is this more true than in today's complex world of e-business where multitudes of interconnected systems must work together to perform many of the simple housekeeping activities which are necessary to keep a computing system healthy. In this world, small things can have wide-reaching implications and few things are as small, yet pervasive, in a computing infrastructure than an event. The event, which encapsulates message data sent as the result of an occurrence, or situation, represents the very foundation on which these complex systems communicate. Basic aspects of enterprise management, such as performance monitoring, security and reliability, as well as fundamental portions of e-business communications, such as order tracking, are grounded in the viability and fidelity of these events, in that quality data lends to accurate, deterministic and proper management of the enterprise.

CEI Common Event Infrastructure are part of the SOA core. CEI can be used to capture events and monitor applications, for example, in IBM WebSphere Business Monitor or IBM Tivoli products. WebSphere Process Server builds on and leverages CEI to emit a specific set of events for each SCA service component.

CICS Customer Information Control System (CICS) is a family of application servers and connectors that provides industrial-strength, online transaction management and connectivity for mission-critical applications.

Cleanup Server Cleanup Server is a runtime component of WebSphere MQ Workflow architecture. The Cleanup Server is responsible for physically deleting process instances that are finished. Depending on the definitions you set for your system, finished processes are deleted immediately or later in the day when the system is idle.

Debug perspective This is the perspective in the WebSphere Integration Developer used for debugging business process and the Java, or anything associated with them.

Domain A domain represents all or parts of your organization. There can only be one domain per setup. Any properties defined at this level, such as staff, data structures, programs, IT resources, are valid for all systems and are inherited by all lower levels.

EJB Enterprise JavaBeans.

Event messages Messages produced by software on a machine indicating a specific event or error.

Execution Server Execution Server is a runtime component of WebSphere MQ Workflow architecture. The Execution Server is responsible for the execution of processes. If a process contains activities for automatic execution, the Execution Server performs the right activities at the right time without human intervention. If staff is involved in a process, the right work item is moved to the right person at the right time. It maintains the information about the states of all activities in the Runtime database. It acts as a database client to communicate with the database server.

FDL WebSphere MQ Workflow Definition Language is the language used to exchange WebSphere MQ Workflow information between WebSphere MQ Workflow system groups. The language is used by the import and export function of WebSphere MQ Workflow and contains the workflow definitions for staff, programs, data structures, and topology. his allows non-WebSphere MQ Workflow components to interact with WebSphere MQ Workflow.

Fmcibie The runtime import/export utility of WebSphere MQ Workflow.

Foreign bus Connect different buses together by configuration. Extending a bus network in this way enables all the buses in the network to exchange messages.

GUI Graphical User Interface.

HHTP Hypertext Transfer Protocol.

HTM Human Task Manager is component of WebSphere Process Server runtime architecture and supports the ad hoc creation and tracking of tasks. You can use existing LDAP directories (as well as operating system repositories and the WebSphere user registry) to access staff information.

HTML Hypertext Markup Language

IMS Information Management System (IMS) is IBM's premier transaction & hierarchical database management system. The latest capabilities enable SOA exploitation, secure your investment and enable new application development.

J2EE Java Platform, Enterprise Edition or Java EE (formerly known as Java 2 Platform, Enterprise Edition or J2EE up to version 1.5), is a programming platform—part of the Java Platform—for developing and running distributed multi-tier architecture Java applications, based largely on modular software components running on an application server. The Java EE platform is defined by a specification. Similar to other Java Community Process specifications, Java EE is also considered informally to be a standard because providers must agree to certain conformance requirements in order to declare their products as Java EE compliant; albeit with no ISO or ECMA standard.
Java EE includes several API specifications, such as JDBC, RMI, e-mail, JMS, Web services, XML, so on, and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise Java Beans, servlets, portlets (following the Java Portlet specification), JavaServer Pages and several Web service technologies. This allows the developer to create an enterprise application that is portable between platforms and scalable, while integrating with legacy technologies. Other added bonuses are, for example, that the application server can handle the transactions, security, scalability, concurrency and management of the components that are deployed to it, meaning that the developers can concentrate more on the business logic of the components rather than the lower level maintenance tasks.

JAR Java Archive.

Java Snippet A small part of a program code written in JAVA.

JCA The J2EE Connector Architecture is a Java based technology solution for connecting application servers and enterprise information systems.

JDBC Java Database Connection.

JMS Java Message Service, part of the J2EE (Java 2 Enterprise Edition) suite, provides standard APIs that Java developers can use to access the common features of enterprise message systems. JMS supports the publish/subscribe and point-to-point models and allows the creation of message types consisting of arbitrary Java objects.

JNDI Java Naming and Directory Interface (JNDI) is an API specified in Java technology that provides naming and directory functionality to applications written in the Java programming language. It is designed especially for the Java platform using Java's object model. Using JNDI, applications based on Java technology can store and retrieve named Java objects of any type. In addition, JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.

JSP JavaServer Pages.

LDAP Lightweight Directory Access Protocol is a networking protocol for querying and modifying directory services running over TCP/IP.
A directory is a set of information with similar attributes organized in a logical and hierarchical manner. The most common example is the telephone directory, which consists of a series of names (either of a person or organization) organized alphabetically, with an address and phone number attached.

An LDAP directory often reflects various political, geographic, and/or organizational boundaries, depending on the model chosen. LDAP deployments today tend to use Domain Name System (DNS) names for structuring the topmost levels of the hierarchy. Deeper inside the directory might appear entries representing people, organizational units, printers, documents, groups of people or anything else which represents a given tree entry (or multiple entries).

LDIF LDAP Data Interchange Format is a standard data interchange format for representing (LDAP) directory content as well as directory update (Add, Modify, Delete, Rename) requests. It represents update requests as a set of records, one record for each update request. In both cases, the data is presented in a plain text form.

MDB message driven bean is an Enterprise JavaBean (EJB) similar to a session bean, except it responds to a JMS message rather than an RMI event. MDBs were introduced in the EJB 2.0 specification which is supported by Java 2 Platform, Enterprise Edition 1.3 or higher. The message bean represents the integration of JMS (Java Message Service) with EJB to create an entirely new type of bean designed to handle asynchronous JMS messages.

MTBF Mean Time Between Failure is the average time between failures, the reciprocal of the failure rate in the special case when failure rate is constant. Calculations of MTBF assume that a system is "renewed", that is, fixed, after each failure, and then returned to service immediately after failure.

MTTR Mean Time To Recovery is the average time that a device takes to recover from a non-terminal failure. Examples of such devices range from self-resetting fuses (where the MTTR would be very short, probably seconds), up to whole systems which have to be replaced.

MyTask portlet The My Task portlet displays the human task instances on the portal server.

PEA Program Execution Agent is a component of WebSphere MQ Workflow that manages the implementations of program activities, such as .EXE and .DLL files.

PES Program Execution Server invokes application programs that you define in theWebSphere MQ Workflow model. These programs run in subsystem environments of z/OS, for example, IMS or CICS.

Portal client WebSphere MQ Workflow Portal client.

Queue manager A queue manager is a system program that provides queuing services to applications. It is used to enable communication between the WebSphere Message Broker components, each component requires access to a Queue Manager.

Rational® Agent Controller The Rational Agent Controller is used for message flow debugging in the Message Brokers Toolkit. It must be installed on the same machine as the broker being debugged.

RPC Remote Procedure Call is a protocol that provides the high-level communications paradigm used in the operating system.

SCA The Service Component Architecture is the base for a service-oriented architecture because the different services can be easily connected to solutions.

SCA component A part of a SCA module which has a specific functionality. They can be assembled to a meaningful service.

SCA Export An possibility to provide a interface which can be called by other SCA modules via an import.

SCA Import An possibility to call a interface of an other SCA module which is provided via an export.

SCA Module A logical unit of different SCA components building a service.

Scheduling Server Scheduling Server is a runtime component of WebSphere MQ Workflow architecture. The Scheduling Server controls and manages notification for activities that must be performed within a certain time frame. For example, if activities or work items are overdue for a process, the Scheduling Server changes the activity state to expired. For work items, the Scheduling Server sends notifications to the worklists of the relevant persons.

SDDS An internal (proprietary) format of WebSphere MQ Workflow messages that is generated by the client API functions.

SDO The Service Data Object is a data representation mostly used to connect to a enterprise information system. It caches the data so that the consumer do not have to care about interacting with the EIS directly.

SI Bus A logical entity based on the physical implementation of a Message Driven Bean, there is no inherent high availability or workload management functionality.

SMO A service message object provides an abstraction layer for processing and manipulating messages exchanged between services.

SOA Service-oriented architecture is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services. SOA helps users build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes.

System group Within a system group, all systems share the same database. By installing more than one system in a system group, the workload for process execution can be distributed while still sharing the same data and the same workflow model. Databases cannot be shared by several system groups.

TEL Task Execution Language is XML based language that defines and models predefined tasks at Human Task Manager Component (HTM).

UDDI Universal Description, Discovery and Integration.

UPES User-defined program execution server (UPES) must be implemented as a custom program using the documented message formats that apply to it. This means that the user-defined program execution server must be able to handle the messages. The message format is XML, and the protocol is restricted to the message types that are needed to invoke and finish a program. With the user-defined program execution server you can integrate any type of application and use these applications to extend the management capabilities of WebSphere MQ Workflow.

WebSphere MQ Workflow program execution server: To invoke application programs within your workflow, WebSphere MQ Workflow uses a program execution agent for the execution of executable programs (EXE or DLL) on a client machine and a program execution server for the automatic and unattended execution of backend programs on the server. The program execution server is available for z/OS only and supports the invocation of IMS and CICS transactions.

UPES Framework To help implement a UPES application, WebSphere MQ Workflow provides a UPES Framework library. It is best practice to use this UPES Framework to develop a new UPES. The UPES Framework is a fully integrated and fully functional server framework.

WAR Web Archive.

Web client WebSphere MQ Workflow Web client.

WebSphere Application Server A very powerful application server offered from IBM. It is also included as a powerful foundation in other products like WebSphere Portal, WebSphere Process Server.

WebSphere MQ A messaging application which enables the Message Brokers Toolkit, Configuration Manager, and brokers to communicate. WebSphere MQ provides many of the available transport protocols between business applications and message flows.

WebSphere MQ Explorer A graphical user interface for WebSphere MQ for administering WebSphere MQ components such as queue managers, channels and queue.

WebSphere Process Server A state of the art Business Process execution environment.

WSDL Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services.
WSDL is an XML-based service description on how to communicate using Web services. The WSDL defines services as collections of network endpoints, or ports. WSDL specification provides an XML format for documents for this purpose.

XML Extended Markup Language.

XSD XML Schema, published as a W3C Recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve Recommendation status by the W3C.
Like all XML schema languages, XML Schema can be used to express a schema: a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. However, unlike most other schema languages, XML Schema was also designed with the intent of validation resulting in a collection of information adhering to specific datatypes, which can be useful in the development of XML document processing software.
An XML Schema instance is an XML Schema Definition (XSD) and typically has the filename extension ".xsd". The language itself is sometimes informally referenced as XSD.

XSL The eXtensible Stylesheet Language (XSL) is a family of transformation languages which allows one to describe how files encoded in the XML standard are to be formatted or transformed. There are three languages in the family:

* XSL Transformations (XSLT): an XML language for transforming XML documents

* XSL Formatting Objects (XSL-FO): an XML language for specifying the visual formatting of an XML document

* the XML Path Language (XPath): a non-XML language used by XSLT, and also available for use in non-XSLT contexts, for addressing the parts of an XML document.

Abbreviations and acronyms

API	Application Program Interface
ARM	Application response monitor
ASF	Application Server Facility
BFM	Business Flow Manager
BOs	Business Objects
BPC	Business Process Choreographer
BPE	Business Process Engine
BPEDB	Business Process Engine database
BPEL	Business Process Execution Language
BPM	Business Process Management
CBE	Common Base Event
CCDT	Client channel definition table
CEI	Common Event Infrastructure
CIF	Common Internet File System
CSIv2	Common Secure Interoperability Version 2
CSS	Cascading style sheet
CTG	CICS Transaction Gateway
EAR	Enterprise archive
EIS	Enterprise Information System
EJB	Enterprise JavaBeans
ESB	Enterprise Service Bus
EXCI	External CICS interface
FDL2BPEL	FDL to BPEL Conversion Tool Version 6.0.2
FFST™	First-failure support technology
HTM	Human Task Manager
IBM	International Business Machines Corporation
IT	Information Technology
ITSO	International Technical Support Organization
J2C	J2EE Connector Architecture
J2EE	Java 2 Enterprise Edition
JAAS	Java Authentication and Authorization Service
JDBC	Java Database Connectivity
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JNI™	Java Native Interface
JSF	JavaServer Faces
JSP	JavaServer Pages
JVM	Java Virtual Machine
KPI	Key performance indicator
LDAP	Lightweight Directory Access Protocol
MDB	Message driven bean
MQMD	Message descriptor
MsgId	Message identifier
NCName	No-colon-name
ND	Network Deployment
ORB	Object Request Broker
OS	Operating System
PEA	Program execution agent
PES	Program execution server
PMI	Performance monitoring infrastructure
POJO	Plain Old Java Object
QP	Query properties

RAD	Rapid Application Development
RDBMS	Relational Database Management System
SCA	Service Component Architecture
SCDL	Service Component Definition Language
SDO	Service Data Object
SLA	Service Level Agreement
SOA	Service-oriented architecture
SSL	Secure Sockets Layer
TEL	Task Execution Language
UI	User interfaces
UPES	User-Defined Program Execution Server
VMM	Virtual Member Manager
WfMC	Workflow Management Coalition
WID	WebSphere Integration Developer
WSDL	Web Services Description Language
WSRR	WebSphere Service Registry and Repository
XSD	XML Schema Definition

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this IBM Redbooks publication.

WebSphere MQ Workflow terms

Abbreviation	Source
PG	*WebSphere MQ 3.6 Programming Guide,* SH12-6291
AG	*WebSphere MQ Workflow 3.6 Administration Guide,* SH12-6289
CA	*WebSphere MQ Workflow 3.6 Concepts and Architecture,* GH12-6285
IG	*WebSphere MQ Workflow 3.6 Installation Guide,* SH12-6288
BT	*WebSphere MQ Workflow 3.6 Getting Started with Buildtime,* SH12-6286
RT	*WebSphere MQ Workflow 3.6 Getting Started with Runtime, SH12-6287*
AL	*WebSphere MQ Workflow 3.6 Announcement letter*

These resources are available at:

`http://www.ibm.com/software/integration/wmqwf/library/index.html`

Term	Source	Definition
Administration server	RT	The administration server manages a WebSphere MQ Workflow system. It communicates with all other components in a system or system group. It is the working center of the administration component. It is responsible for the availability, operation, and error recovery of all server components. A command interface is used to send requests to the server's administration utility.

Term	Source	Definition
Administration utility		The administration utility is the administrator's user interface to request services from the administration server. Using this utility, a WebSphere MQ Workflow system is started and stopped and the current state of any server can be listed. Furthermore, error logs can be accessed to check whether any problems exist within the system.
Audit trail		The audit trail is used for monitoring and auditing of processes in WebSphere MQ Workflow. Events can be configured to be saved in the audit trail. There are many event types that can be set to be written. You can specify whether to save the audit trail data in a database or a WebSphere MQ queue.
Buildtime	RT	A WebSphere MQ Workflow component with a graphical user interface for creating and maintaining workflow models, administering resources, and the system network definitions.
CICS		Customer Information Control System (CICS) is a family of application servers and connectors that provides industrial-strength, online transaction management and connectivity for mission-critical applications.
Cleanup server	AG, RT	The cleanup server is responsible for physically deleting process instances that are finished. Depending on the definitions set for the system, finished processes are deleted immediately or later in the day when the system is idle.
Domain	RT	A domain represents all or parts of your organization. There can only be one domain per setup. Any properties defined at this level, such as staff, data structures, programs, and IT resources, are valid for all systems and are inherited by all lower levels.
Execution server	PG	The execution server is a runtime component of WebSphere MQ Workflow architecture. The execution server is responsible for the execution of processes. If a process contains activities for automatic execution, the execution server performs the correct activities at the correct time without human intervention. If staff is involved in a process, the correct work item is moved to the correct person at the correct time. It maintains the information about the states of all activities in the Runtime database. It acts as a database client to communicate with the database server.

Term	Source	Definition
FDL	CA	WebSphere MQ Workflow Definition Language is the language used to exchange WebSphere MQ Workflow information between WebSphere MQ Workflow system groups. The language is used by the import and export function of WebSphere MQ Workflow and contains the workflow definitions for staff, programs, data structures, and topology. This allows non-WebSphere MQ Workflow components to interact with WebSphere MQ Workflow.
fmcibie		The runtime import/export utility of WebSphere MQ Workflow.
IMS		Information Management System (IMS) is the premier IBM transaction and hierarchical database management system. The latest capabilities enable SOA exploitation, secure your investment, and enable new application development.
LDIF file		LDAP Data Interchange Format.
MyTask portlet		The My Task portlet displays the human task instances on the portal server.
PEA	PG	WebSphere MQ Workflow program execution agent: The WebSphere MQ Workflow component that manages the implementations of program activities, such as .EXE and .DLL files.
Portal client	PG	WebSphere MQ Workflow Portal client.
RPC		Remote Procedure Call (RPC) is a protocol that provides the high-level communications paradigm used in the operating system.
Runtime/ Runtime database	RT, IG	
Scheduling server	RT	The scheduling server controls and manages notification for activities that must be performed within a certain time frame. For example, if activities or work items are overdue for a process, the scheduling server changes the state of the activity to expired. For work items, the scheduling server sends notifications to the worklists of the relevant persons.
SDDS		An internal (proprietary) format of WebSphere MQ Workflow messages that is generated by the client API functions.

Term	Source	Definition
System group	AG	Within a system group, all systems share the same database. By installing more than one system in a system group, the workload for process execution can be distributed while still sharing the same data and the same workflow model. Databases cannot be shared by several system groups.
UPES	PG, CA	WebSphere MQ Workflow program execution server: To invoke application programs within your workflow, WebSphere MQ Workflow uses a program execution agent for the execution of executable programs (EXE or DLL) on a client machine and a program execution server for the automatic and unattended execution of backend programs on the server. The program execution server is available for z/OS only and supports the invocation of IMS and CICS transactions.
UPES Framework		To help implement a UPES application, WebSphere MQ Workflow provides a UPES Framework library. It is a best practice to use this UPES Framework to develop a new UPES. The UPES Framework is a fully integrated and fully functional server framework.
Web client	PG	WebSphere MQ Workflow Web client.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks” on page 457. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere for z/OS V6 Connectivity Handbook*, SG24-7064

  http://www.redbooks.ibm.com/abstracts/sg247064.html

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

  http://www.redbooks.ibm.com/abstracts/sg247413.html

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

  http://www.redbooks.ibm.com/abstracts/sg246688.html

- *WebSphere MQ Application Programming Guide*, SC34-6595

  http://www.ibm.com/support/docview.wss?uid=pub1sc34659500

Other publications

WebSphere MQ System Administration Guide, SC34-6584, is also relevant as a further information source:

http://www.ibm.com/software/integration/wmq/library/library6x.html

Online resources

These Web sites are also relevant as further information sources:

- WebSphere Process Server and WebSphere Process Server products installation and system requirements documentation

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp

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

- Description of the mapping constructs, SupportPac WA73

  http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en

- WebSphere Process Server V6.1 information center

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm

- Continue On Error setting

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/tasks/cconerr.html

- Fault handling and compensation handling

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm

- Human Task Manager, WebSphere Process Server information center

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/ctaskassign.html

- Manage people directory configurations, WebSphere Process Server V6.1

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/ctaskassign.html

- Business Process Choreographer authorization, WebSphere Process Server V6.1 information center

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/cstaffdefaultusers.html

- People resolution mechanism, WebSphere Process Server V6.1 information center

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/cstaffdefaultusers.html

- Service Component Architecture

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.prodovr.doc/topics/csrvcomparch.html

- Enterprise information systems (EIS) with IBM WebSphere Adapters

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.basics.doc/topics/aaccess1.html

- Bindings and on selecting the most appropriate one

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.basics.doc/topics/cbindings.html

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.prodovr.doc/topics/cappbnd.html

- Overview of adapters

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wbpm.scenarios.esb1.610.doc/concepts/cwesb_scenario_adapters.html

- Configuring and using adapters

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/tcreatecmps.html

- Developing services with adapters

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/coverview.html

- IMS TM resource adapter

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/ims_overview.html

- CICS ECI resource adapter

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.adapter.emd.ui.doc/topics/cicsoverview.html

- How to connect to CICS

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpm.scenarios.esb1.610.doc/concepts/cwesb_scenario_cics.html

- Invocation scenarios for business processes

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/cinvocation.html

- Implementing a human-centric business process application using WebSphere Portlet Factory: Part 1: Solution overview

  http://www.ibm.com/developerworks/websphere/library/techarticles/0804_ng/0804_ng.html

- Implementing a human-centric business process application using WebSphere Portlet Factory: Part 2: Developing a human-centric business process

  http://www.ibm.com/developerworks/websphere/library/techarticles/0804_kapadia/0804_kapadia.html

- Developing client applications for business processes and tasks, WebSphere Process Server V6.1 information center

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6ovr.html

- JSF-based client sample

  http://publib.boulder.ibm.com/bpcsamp/

- Setting up a user interface for your human task

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.tel.ui.doc/topics/tnclient.html

- About Business Process Choreographer Explorer

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/c7webclt.html

- Getting started with Business Process Choreographer Explorer

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t7explorer_using.html

- Business Process Choreographer Explorer user interface

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/c7explorer_areas.html

- Planning for the Business Process Choreographer Explorer

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t1pl_explorer.html

- Developing client applications for business processes and tasks

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6ovr.html

- Scripts to administratively clean up processes and tasks

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?

- DB2 Universal Database™ V8.2

  http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp

- WebSphere Process Server V6

  http://www.ibm.com/software/integration/wps/library/

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm

- WebSphere Process Server courses and self-guided training materials

  http://www.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_list&subChapter=2079&subChapterInd=S&subChapterName=WebSphere+Process+Serve

- Self-study information about *in the spotlight* documentation, planned maintenance, education, and general self-help resources

  http://www.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&dc=DA400&uid=swg27008332&loc=en_US&cs=UTF-8&lang=en&rss=ct2307websphere

- The IBM Education Assistant training on WebSphere Process Server and WebSphere Integration Developer

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wpi_v6/wpi6_coverpage.html

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wpi_v6/wpswid/6.0.2/Migration.html

- A wide range of reference and education materials on WebSphere Process Server and Business Process Choreographer developerWorks

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

- A model from WebSphere Business Integration Workbench V4.2.4 can be imported into WebSphere Business Modeler V6.1

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.btools.help.modeler602.doc/doc/tasks/migrating/importingadf.html

- Using Loops in WebSphere Business Modeler v6 to improve simulations and export to BPEL

  http://www.ibm.com/developerworks/websphere/library/techarticles/0703_fasbinder/0703_fasbinder.html

- FDL2BPEL Conversion tool built into WebSphere Integration Developer

  http://www.ibm.com/support/docview.wss?rs=795&uid=swg24008362

- SupportPac WA73: FDL2BPEL Conversion Tool

  http://www.ibm.com/support/docview.wss?rs=171&uid=swg24008362&loc=en_US&cs=utf-8&lang=en>>

- Authorization and staff resolution in Business Process Choreographer: Part 3: Customization options for staff resolution

  http://www.ibm.com/developerworks/websphere/techjournal/0712_lind/0712_lind.html

- SupportPac WA05: WebSphere MQ Workflow

  WebSphere MQ Workflow, Java UPES (toolkit)

  http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000664&loc=en_US&cs=utf-8&lang=en

- WebSphere MQ Workflow SupportPac WA07

  WebSphere MQ Workflow, Web Services Process Management Toolkit

  http://www.ibm.com/support/docview.wss?rs=171&uid=swg24000662&loc=en_US&cs=utf-8&lang=en

- Using the report options in XML clients and UPES servers

  http://www.ibm.com/support/docview.wss?rs=795&context=SSVLA5&q1=report&uid=swg21171263&loc=en_US&cs=utf-8&lang=en

- WebSphere MQ Workflow-invocation style

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_macste.html

- Explicitly identify the starting activity

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_macste_nonu.html

- Synchronous interface and asynchronous interface

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6task_startsynch.html

- WebSphere Process Server select clause for a process instance query

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6query.html

- Repairing activities

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6activity_repair.html

- Handling exceptions and faults

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_errors.html

- Enhanced continue-on-error behavior of activities and business processes

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.612.doc/doc/bpc/cfaulthandling_continueonerror.html

- Queries on business-process and task-related objects

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6query.html

- Processing human task activities

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_staff.html

- Processing a single person workflow

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_singleworkflow.html

- Developing applications for human tasks

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6tasks.html

- Javadoc of Package com.ibm.task.api

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/task/api/package-summary.html

- WebSphere Process Server Business Flow Manager interface

  com.ibm.bpe.api interface BusinessFlowManagerService

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/package-summary.html

- WebSphere Process Server Human Task Manager interface

  com.ibm.task.api interface HumanTaskManagerService

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/task/api/HumanTaskManagerService.html

- QUERY_PROPERTY view

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/r6bpel_viewqueryprops.html

- Filtering data using variables in queries

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_queryvariable.html

- JavaDoc

  Package com.ibm.bpe.api

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/package-summary.html

  Package com.ibm.bpe.api Interface BusinessFlowManager

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/BusinessFlowManager.html

- Interface QueryProperty

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.610.doc/web/apidocs/com/ibm/bpe/api/QueryProperty.html

- Query properties tab; business process editor

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/ref/rquery.html

- Declaring a query property for a variable

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.610.help.bpel.ui.doc/tasks/tdecqprp.html

- Reference Guide for the Java 2 SDK, Standard Edition, v 1.4

  http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html

- Customizing application login with Java Authentication and Authorization Service

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_j2clogin.html

- Developing programmatic logins with the Java Authentication and Authorization Service

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_pacs.html

- Programmatic logins

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/xsec_jaas.html

- Customizing Web application login

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tsec_pofolo.html

- Form login

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/xsec_formlogin.html

- JAAS exit; IBM WebSphere Developer Technical Journal, Advanced authentication in WebSphere Application Server, Managing user authenticity and privileges in a distributed application server environment

  http://www-128.ibm.com/developerworks/websphere/techjournal/0508_benantar/0508_benantar.html

- Integrating portal and business processes

  http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp?topic=/com.ibm.wp.ent.doc/wps/bizproc_6012.html

- Three main production topology patterns

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.wps.610.doc/doc/cpln_topologypat.html

- IBM WebSphere Developer Technical Journal, Transactional high availability and deployment considerations in WebSphere Application Server V6

  http://www.ibm.com/developerworks/websphere/techjournal/0504_beaven/0504_beaven.html

- Best Practice; WebSphere HTTP Plugin Failover for High Volume Web Sites

  http://www.ibm.com/developerworks/websphere/library/bestpractices/plugin_failover_hvws.html

- Deleting process instances and deleting task instances

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/t6bpel_admdfe.html

- Securing applications and their environment

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/doc/welcome_wps_sec.html

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

C

D

E

F

G

H

I

J

K

L

M

S

T

U

V

W

X

WebSphere MQ Workflow Transition to WebSphere Process Server

WebSphere MQ Workflow Transition to WebSphere Process Server

Transition concepts and planning

Transition support and guidance

Transition examples

This IBM® Redbooks® publication provides guidance on how to transition from a WebSphere® MQ Workflow 3.6 environment to WebSphere Process Server V 6.1. It provides a conceptual overview of WebSphere MQ Workflow and WebSphere Process Server, describes new features provided by the WebSphere Process Server, and discusses benefits of a transition to the new environment.

The book discusses the transition concepts available for converting a business process from WebSphere MQ Workflow to WebSphere Process Server.

It compares human interaction in business processes in WebSphere MQ Workflow to people assignment in WebSphere Process Server and how to map from one to the other. Integration of back-end applications is described for both environments, together with information about how to transition from one to the other. This book compares how clients were implemented in WebSphere MQ Workflow and how the corresponding implementation is performed in WebSphere Process Server. It helps assess the topology in place in WebSphere MQ Workflow and define a corresponding one in WebSphere Process Server, as well as best practices for target topology high availability, scalability, deployment of applications, and administration.

This book has two sections:

- Part 1 provides planning information required to assess the current environment, define the target environment, and plan for a transition from WebSphere MQ Workflow 3.6 to WebSphere Process Server V6.1.
- Part 2 provides detailed information about transition techniques, areas for transition, tools available, artifacts involved, and best practices.

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

SG24-7282-00 ISBN 0738432121