WebSphere. software

IBM

Microsoft BizTalk to WebSphere Business Integration Server Express Migration

- Best migration process to WebSphere Business Integration
- Understand the similarities and differences between BizTalk and WBIS
- Learn the techniques to migrate to WBIS

Boaz Carmeli
Yardena Peres

Redbooks

ibm.com/redbooks

International Technical Support Organization

Microsoft BizTalk to WebSphere Business Integration Server Express Migration

March 2006

SG24-7142-00

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

First Edition (March 2006)

This edition applies to WebSphere Business Integration Server Foundation 5.1.

Contents

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

Trademarks

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

@server®	CICS®	Rational Rose®
@server®	DB2®	Rational®
Redbooks (logo) ™	IBM®	Redbooks™
AIX®	IMS™	WebSphere®
AS/400®	OS/390®	XDE™

The following terms are trademarks of other companies:

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

BizTalk, JScript, Microsoft, Visio, Visual Basic, Visual C++, Visual Studio, 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

The purpose of this IBM® Redbook is to help the reader understand what is involved in migrating a BizTalk® application into the WebSphere Business Integration (WBIS) environment. It deals specifically with migrating a BizTalk 2002 application into the WebSphere® Business Integration Server Express (WBI SE) environment.

It begins with a general discussion of business integration platforms, and migration drivers and issues relevant in the small to medium business environment. Several migration options are described and some basic migration techniques are presented.

A comprehensive description of the BizTalk and WBIS platforms is provided, highlighting the similarities and differences between the two environments. A migration path and a set of best practices are proposed to migrate a BizTalk application into the WebSphere Business Integration environment. Finally, the detailed steps for migrating the functional and architectural components of BizTalk to the WBIS environment are presented.

Readers are assumed to be familiar with the concepts of business integration and distributed applications, including multi-tiered architecture, synchronous and asynchronous messaging, transformations, content-based routing, and workflows. This Redbook also assumes knowledge of cross-platform open standards, such as eXtensible Markup Language (XML) syntax and methods, Web Services Description Language (WSDL), and Business Process Execution Language (BPEL).

This Redbook is intended for both internal IBM employees and external IT personnel who intend to migrate a BizTalk-based business integration application into the WebSphere Business Integration environment.

The team that wrote this redbook

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

Boaz Carmeli is a researcher in the area of Information Technology and Business Process. He works extensively with the IBM WebSphere product line and other IBM middleware. On his previous research projects Boaz worked on

computer network-related issues such as reliable multicast and personal area networks.

Yardena Peres is a research staff member at IBM Haifa Research Lab in Haifa, Israel. She joined IBM in 1992, and has since worked on business process integration, management applications, search technologies and user interface frameworks. She has a B.A. and M.Sc. in Computer Science from the Technion, Haifa, Israel.

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

Rufus P. Credle Jr., Jeanne Tucker, Diane O'Shea
International Technical Support Organization, Raleigh Center

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 team with IBM technical professionals, Business Partners and/or customers.

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 e-mail to:

  redbook@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

BizTalk to WBIS migration path

1

Executive summary

Business integration is currently one of the hottest competition points in the IT world. Integrating the IT infrastructure across a business is essential since companies can no longer assume that a single product will address all their business needs. The concept and architecture of a business integration platform was developed and introduced to address the flexibility and responsibility required in today's business world.

As part of IBM's effort to gain market share in the small and medium business environment (SMB), this Redbook looks into the definition of a set of methodologies required to support seamless and straightforward migration from BizTalk, a Microsoft® based business integration platform, into the open standard, IBM-based WebSphere Business Integration (WBIS) environment.

We begin this document with a general explanation of business integration platforms, and with a discussion of their various aspects and components. We then move on to discuss the different migration drivers and issues relevant in the SMB environment. By providing a comprehensive description of diverse migration aspects, we begin to understand the issues involved.

After providing a definition of migration and a description of the SMB characteristics related to business integration platforms, we analyze the possible directions for migration. We consider three possible options:

- Migration at the IDE level
- Migration at the runtime level

- Migration at the operating system level

We then proceed to explain why the first option was chosen.

Continuing the migration discussion we list manual and automatic migration techniques that are appropriate. Special attention is given to the following techniques:

- XML transformation that allows transformation of a source XML structure to a target one.
- Interface adjustment that suggests migrating just the interface of a custom code from the source to the target platform, allowing the custom code piece to run as is under the target platform.
- Component substitution that suggests replacing an entire, well-defined, source component with its target counterpart.
- Use of a ready made adapter from the target environment to communicate with an existing adapter from the source environment without changing it.

We provide a comprehensive description of the BizTalk platforms. Although the BizTalk product has evolved through several versions, in this Redbook we review BizTalk 2002 (note that BizTalk 2000 very much resembles BizTalk 2002).

After presenting the BizTalk platform, we describe the WebSphere Business Integration platform. We first look at the various WBIS server families that exist as part of the IBM portfolio, and the supporting transport they use. We include a comprehensive description of the various development and administration tools of the WBI Server Express (SE).

After describing the source and target platforms, we divide them into several functional and architectural parts:

- Messaging - BizTalk Messages and WBIS Business Objects
- Adapters and other application connectivity components - BizTalk Receive Function, Channel, Port, etc., and WBIS Adapters
- Data transformation - BizTalk Transformation and WBIS Mapping
- Process choreography - BizTalk Orchestration and WBIS Collaboration
- System configuration
- Custom software component execution

We then compare the BizTalk and WBIS approaches for each part and suggest a set of techniques to migrate them from the source to the target platform.

2

Introduction

As part of IBM's effort to gain market share in the small and medium business (SMB) environment, a set of tools is required to support seamless and straightforward migration from Microsoft-based information technology (IT) infrastructure to open standard IBM-based IT. Microsoft is gaining market share in the SMB environment due to its strength in development tools and its unified set of business integration software. IBM, which traditionally positions its IT solutions for high-end enterprise businesses, provides the business integration arena with scalable, robust, and mature products that are based on open standards.

Business integration is currently one of the hottest competition points in the IT world. Businesses can no longer assume that a single product will address all their business needs. Integrating the IT infrastructure across the business is essential when addressing the flexibility and responsibility required in today's business world. Business integration platforms like IBM WebSphere Business Integration (WBIS) and Microsoft BizTalk Server have materialized in the market to address the need for integration. Based on the worldwide Internet connectivity and on new communication standards like Web Services, even a small or a medium business can benefit from business integration platforms. Using these platforms allows businesses to efficiently communicate with partners, suppliers, and customers.

This document begins with a general explanation of business integration platforms, discussing their various aspects and components. We then move on

to discuss migration drivers and issues in light of the SMB environment. This discussion includes a list of migration options and a set of general migration techniques that can be utilized while migrating the various parts of a business integration platform. The subsequent sections provide a comprehensive description of the BizTalk and WBIS platforms. After gathering the required insights about BizTalk, WBIS, and the various migration aspects, we illustrate the component mapping between BizTalk and WBIS. This section suggests a set of migration techniques and a migration path for each component, to allow complete migration from the BizTalk environment to WBIS.

2.1 Business integration environment and platforms

Business integration (BI) involves taking a set of distinct, disconnected business applications, processes, people, and data, and combining them into a set of applications, processes, people, and data that are more productive, integrated, easy-to-maintain, and easy-to-control. This integration is usually implemented so that applications and processes can share data and state, primarily when the applications and processes deal with similar types of business data. The applications and processes that may require integration come from all possible sources: packaged applications, custom (in-house) applications, and applications belonging to other businesses (for business-to-business scenarios). This integration often involves some form of *enterprise application integration* (EAI) and *services-oriented architecture* (SOA).

2.1.1 What is WebSphere Business Integration?

At the highest level, a WebSphere Business Integration (WBIS) system consists of an *integration broker* and a set of *adapters* that allow different kinds of business applications to exchange data through the coordinated transfer of information, in the form of business objects. WBIS uses a distributed hub-and-spoke infrastructure, where the spokes of the WBIS architecture are represented by the WBIS adapters. These adapters enable application connectivity and use business objects as containers for the business data exchanged between applications.

The hub is currently implemented by one of the following:

- WBI Server Express (WBI SE), formerly known as WBI InterChange Server (WBI ICS)
- WBI Message Broker (WBI MB)
- WBI Server Foundation (WBI SF), which is based on WebSphere Application Server (WAS) and was formerly branded WAS Enterprise Edition (WASEE).

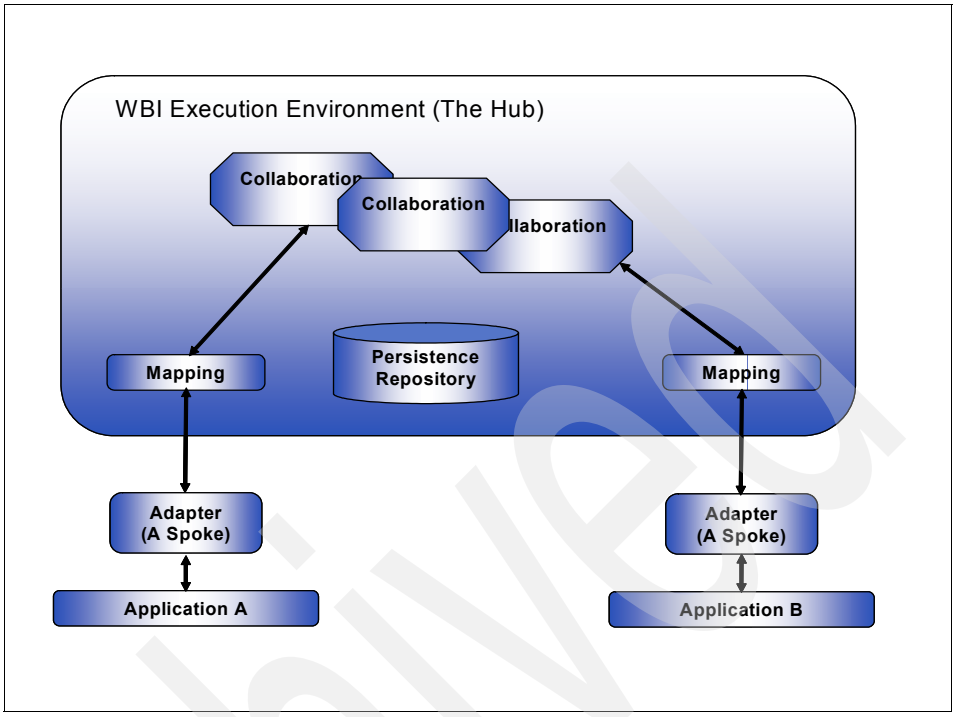

Figure 2-1 WBIS hub and spoke architecture

The small and medium business market is currently targeted by the WBI SE.

Each integration broker runtime is supplemented by a set of control, monitor, and administration tools. WBI SE uses the following tools:

- System Monitor for component activation, deactivation, and runtime monitoring
- Flow Manager for event monitoring and problem determination
- Log Viewer for viewing log files

A set of high-level graphical tools for process choreography configuration and other development activities is an integral part of the business process development environment. For the WBIS environment, these tools are supplied under the WebSphere Application Developer (WSAD), which is an Eclipse-based integrated development environment (IDE). The WBI SE tool set contains, among others, the following tools:

- Business Object Designer for creating business objects
- Object Discovery Agents for automatically generating business objects from a structural source

- Process Designer for creating business processes
- Map Designer for transforming business objects
- Relationship Designer for associating the data from two or more business objects
- Connector Configurator
- Test Connector for aiding in connector testing

2.1.2 What is BizTalk Server?

Microsoft BizTalk Server is an integration server product that enables the development, deployment, and management of integrated business processes and XML-based services. It helps create business processes that unite separate applications into a unified system.

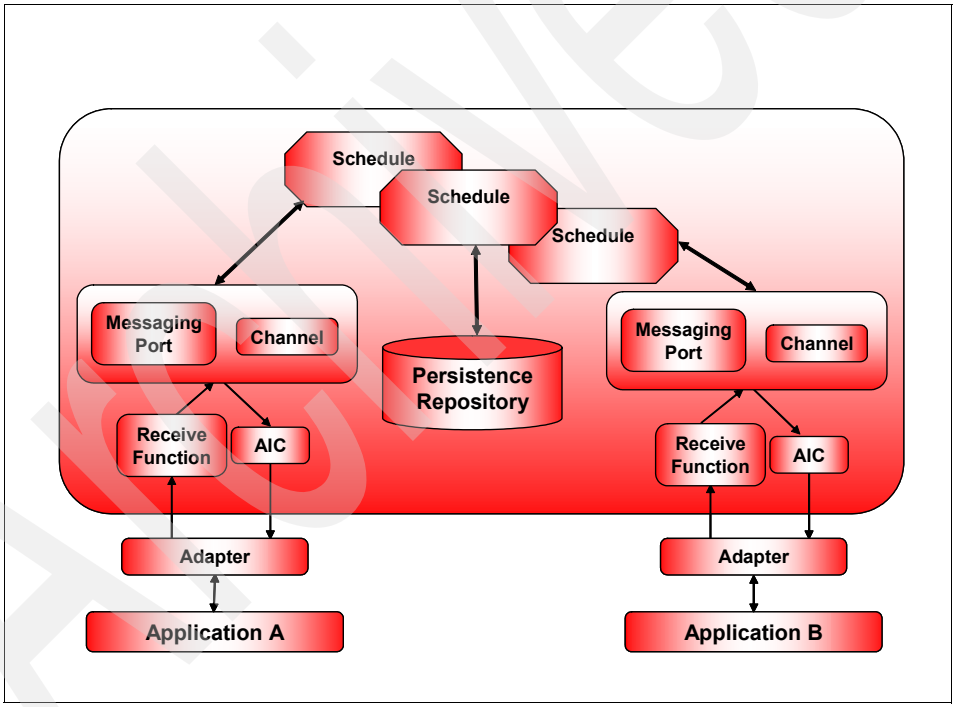

Figure 2-2 BizTalk 2002 high level architecture

BizTalk Server 2002 is COM-based and only partially integrated with Microsoft Visual Studio® .NET.

BizTalk Server enables users to orchestrate dynamic business processes within, and between, organizations. It supports interaction with legacy systems and automation of the daily business flow.

The main components of BizTalk Server 2002 are:

- Receive functions and messaging ports, which provide the communication mechanism through which BizTalk applications interact with other applications
- Channels, which process incoming and outgoing messages
- Orchestrations, which describe business processes
- The BizTalk messaging repository, which serves as a central repository for all user data, storing configuration and state information

The main graphical tools provided by the BizTalk Server 2002 are:

- BizTalk Editor for creating message schemas, based on Microsoft's proprietary XML Schema definition language (XDR)
- BizTalk Mapper for defining translation or transformation of messages
- Orchestration Designer for creating, modifying, and configuring orchestrations
- Management tools

2.2 The small and medium businesses market

The main item that characterizes small and medium businesses is the number of employees. Small businesses are usually defined as having fewer than 1000 employees. A medium business generally employs between 1000 and 5000 employees. A business that employs over 5000 is considered an "enterprise." The type of company can also be categorized by its annual revenue, where businesses with less than $10M annual revenue are considered small, and those that generate more than $100M in revenue are considered enterprises; this leaves the $10M to $100M range for the medium businesses.

The skill level of IT personnel at the SMB is usually limited. Although sufficient administration knowledge usually exists for managing the operating systems and applications, the SMB frequently suffers from lack of infrastructure experience. Most SMBs prefer to buy off-the-shelf solutions to address business needs—or find it convenient to outsource their IT—rather than develop the infrastructure on their own.

The small and medium business market, an IBM must-win, is a $300 billion opportunity. IBM is determined to increase market share by delivering open,

best-of-breed industry solutions and services, both locally and through partnership with Solution Providers (SPs) and other business partners. Business integration platforms, which until now addressed enterprise needs, are expected to play a major role in the growing SMB market.

2.3 Migration definition and needs

Migration is the process of taking software pieces and artifacts that run on one environment and modifying them to run on another environment. Migration can take place at each and every tier in a multi-tiered environment, and at every layer in the application software stack. Traditional migration projects deal primarily with migrating an application from one programming language to another (for example, from C to Java™), or from running on one operating system to another (for example, migrating a product from UNIX® to Windows®). Migration can also enable a business to move from one product to another product supplying basically the same functionality (for example, migrating from Oracle relational database to DB2®).

Recent years have brought a new abstraction layer into the application software stack. Advanced software development tools that are based on an integration development environment (IDE), such as the open code project Eclipse and Microsoft .NET, allow software developers to use XML-based abstraction languages like Unified Modeling Language (UML) and Business Process Execution Language (BPEL) to express business needs and requirements. Adding layers into the application software stack may complicate the migration task. Building this layer using XML-based open standards, on the other hand, opens an opportunity for easily migrating the software artifacts from one environment to another. Using standards-based concepts and architectures, especially those related to XML, enables the use of transformation tools for software migration, like those built on top of the Extensible Stylesheet Language Transformation (XSLT) standard.

2.3.1 Migration drivers

The need for migration in a business can be motivated by a number of different factors, including those discussed in this section.

Cost reduction: Businesses may choose to buy less expensive middleware and tools to reduce infrastructure costs. The WBI SE, targeted specifically for the SMB market, is less expensive than the BizTalk server. Additional cost reduction may come from the ability to run the WBI SE on the Linux® platform, since moving from a Windows-based infrastructure to Linux can reduce infrastructure costs.

Superior feature set: WBI SE has a superior feature set over BizTalk 2002. Written in Java, WBI SE can run over any relevant platform. Using the business objects approach for data description and manipulation allows WBI SE to introduce richer and more powerful capabilities, both at the transformation and mapping segments, and at the process choreography segment. Building on IBM market-leading technologies for enterprise businesses, WBI SE has the scalability, robustness, and high availability capabilities required to address all the SMB market needs.

Solution Providers' needs: Many companies in the SMB market use Solution Providers (SPs) for their IT development and maintenance. These SPs supply SMBs with custom, as well as ready-made, business integration solutions. Tools for migration from BizTalk to WBIS will assist the SP in reusing solutions developed for a BizTalk-based SMB, in a WBIS-based SMB. In this case, the migration needs come from the SP and not directly from the SMB itself.

Outsourcing the IT infrastructure: *Outsourcing* is a growing trend among companies that wish to focus on their core business and let IT experts manage their IT infrastructure. IBM, as a leading outsourcing company, may face the need to migrate Microsoft-based infrastructure to the IBM environment, for cost reduction and easier maintenance. This transformation may require migrating from BizTalk applications into the WBIS environment.

2.4 Summary

This chapter provides an introduction to the business integration environment and to the migration problem we are facing. It starts by introducing the business integration platforms concept in general, and the BizTalk and WBIS platforms in particular. It then covers the SMB environment and the need for migration from BizTalk to WBIS in this market segment.

3

Understanding business integration platforms

This chapter describes the components, features, and benefits of business integration platforms.

3.1 Overview

A well-integrated business process environment must enable users to model business processes, integrate the supply chain based on these models, monitor the implemented solution, and effectively optimize the business processes based on the monitoring inputs.

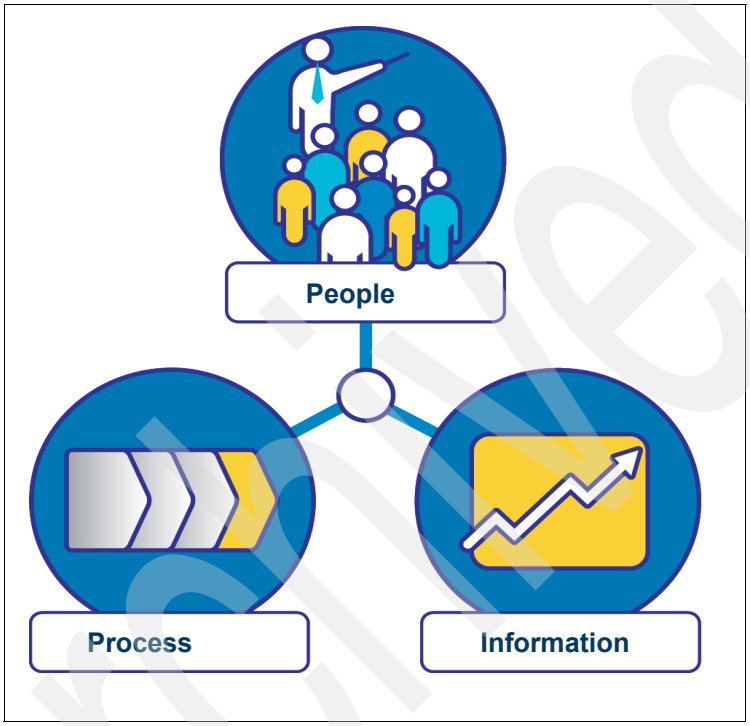

Figure 3-1 Business integration environment

To support the overall business strategies, it is essential to have a responsive IT infrastructure that mirrors the management decisions with corresponding actions. A well-integrated infrastructure allows more immediate and complete response to business initiatives from the appropriate IT systems. A business process platform helps achieve a closer fit between the business strategy and IT systems. One can model existing business processes, or design and implement new ones, quickly and cost-effectively — without expensive coding.

People, processes, applications, systems, and information can be integrated to link and improve end-to-end business operation. Customers and trading partners can be connected with the enterprise, and internal processes can be reused to reduce costs and increase productivity. Business process platforms allow users

to monitor the processes and leverage process metrics data to gain greater control and improve operational decisions. This allows more effective management of the business and the ability to forecast improvements by continual analysis of process performance. In short, business integration can help users maximize enterprise performance.

A business process development environment, which is a fundamental part of the business process platform, allows the business analyst and the software developer to collaborate in a common workspace. They use a visual model of the process that combines and correlates their respective contributions. The developer and analyst create the application by using a drag-and-drop graphical user interface to arrange high-level objects. These objects represent messages, messaging events, business rules and logic, information flows, activities, operations, transformations, and sub-processes.

The development environment also supplies the system administrator with tools and controls for administrating the solutions and applications developed by analysts and developers.

3.2 Business integration roles

The penetration of IT infrastructure into all levels of organizational activities, and the ever-growing need for both application and human collaboration, shape the requirements for a supporting integration environment. Such an environment must address the needs for collaborative development of business processes and activities. The integration system must also provide the ability to integrate existing applications, legacies, and other software components into an orchestration environment. Hence, at the software level, the fundamental principles of a business integration environment are: exposed functionality, document messaging, loose coupling, and platform independence.

Such an environment must also support a variety of human interactions from diverse organizational roles. Figure 3-2 identifies the organizational roles that collaborate to create the business integration platform.

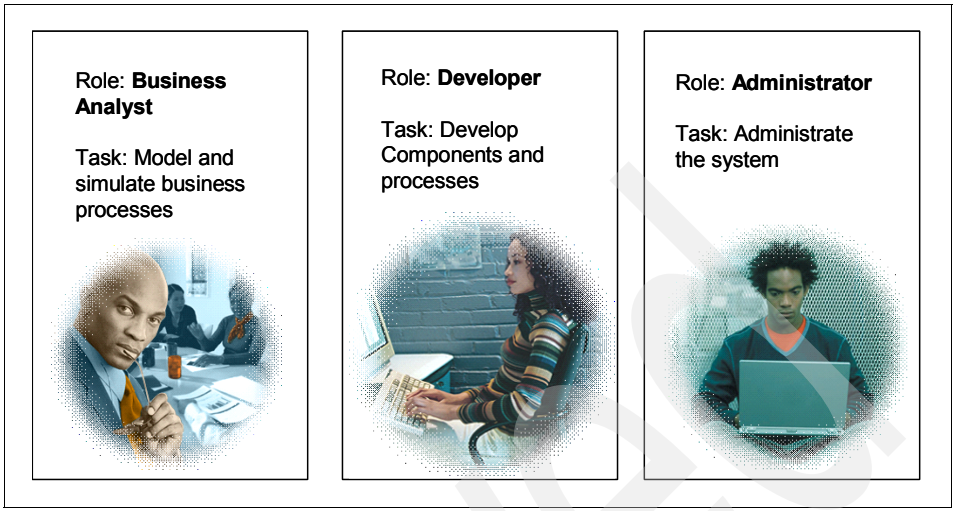

Figure 3-2 Roles in the business integration environment

- The **system administrator** is responsible for installing, maintaining, and managing the business integration platform. The administrator must also provide the access control to support multi-user development. During the runtime phase of the system, the administrator uses monitoring tools to verify the correct functioning of the system. The administrator also provides developers with the means to solve problems and exceptions, if such events occur.
- The **software developer** is responsible of all aspects of software development. The developer's main task is to implement business requirements into the business integration platform. The developer is also responsible for all activities in the software development cycle. After receiving the business requirements, the developer implements them within the business integration platform, then tests and deploys the new functions. Developers usually work in teams; therefore, business integration platforms must support a multi-user software development. The platform must also support all stages of the software development cycle, including debugging, testing, and deployment in the production environment.
- The **business analyst** is responsible for translating business requirements into business rules, and deploying these rules into the business integration platform. The business analyst serves as the bridge between line of business (LOB) personnel and the organization's IT people. The analyst also needs to monitor the execution of the business rules, maintain them, and supply reports and results to business management personnel. Business analysts are not expected to function as software developers. The business integration

platform must supply the analyst with a set of high-level tools for defining, maintaining, and monitoring the rule definitions.

In summary, business integration platforms must provide business personnel with a means to collaborate and share their expertise. The platform must combine the contributions from business workforces into a coherent software solution and supply tools and controls for maintaining and administering its runtime lifecycle.

3.3 Integrated development environment

Business integration platforms use an integrated development environment (IDE) to allow the various BI roles to collaborate on software development and maintenance. The IDE serves as a programming environment, which offers a set of tools tailored to the specific needs and views of each role involved in the business integration development process. Aside from supplying the tools necessary to develop software, the IDE serves as a central location for maintaining the developed software artifacts and tools, including debuggers and a testing environment. The IDE provides means for the deployment process, in which the developed artifacts are deployed into the BI runtime environments.

The IDE supplies developers and users with a variety of development tools to accomplish certain tasks that are fundamental to the business integration platform.

Message structure development: This enables the integration of applications, processes, and data that must be exchanged between participants. Business integration platforms use the notation of messages to describe the pieces of information that flow from one application or process to another. The IDE supplies the user with a tool to generate and edit message structures. Under BizTalk, the development environment offers the Schema Editor to generate and edit message structures. In WBIS, the WSAD environment includes the Business Object Designer for creating and editing message structures.

Message transformation: Different applications and processes use diverse message formats. The messages must be transformed from one format to the other in order to enable the applications and processes to understand each other's different languages. Business integration platforms use a transformation engine to transform messages from one format to the other. The IDE supplies the user with a tool to simplify the transformation task. Transformation may vary in complexity. It may be as simple as copying a field from the input message to the output message, or it may involve a complex task that requires custom code or additional external inputs. The BizTalk environment is based on XSLT transformation with the ability to add custom code or even call external

processes. The WBIS environment uses Java for transformation. While working with Business Objects as the transformation's input and output, the Mapping Tool supplied with the WSAD IDE offers users simple ready-made transformations, such as "move" and "join," and enables them to use Java code to generate any other custom transformations.

Message flow and process choreography: A *process flow* (also called *workflow*) is needed to allow more complex types of application-to-application and human-to-application interaction. A process flow usually represents a business process task and generally involves the cooperation of business analysts and software developers to correctly implement business rules and policies into the infrastructure. The area of process flow definitions and tools for process choreography is quite new, and no mature standards and architectures exist in today's market to address all aspects involved. The Business Process Execution Language (BPEL), which is currently the leading standard for business process description and interoperability, is relatively new, and has limited support both in the BizTalk and the WBIS environments. The most mature part of process choreography and workflows deals mainly with application-to-application interaction. The BizTalk environment supplies the Orchestration Designer tool for workflow creation. WBIS uses collaboration to support process workflow. Two other aspects that influence process choreography are human interaction and the business rule engine.

Human interaction: Developers must be able to create workflows that support human input during process execution. Human interaction introduces an additional set of requirements to the runtime components of the business integration platform. These requirements are raised due to the asynchronous nature of the interaction, which includes long response times. A process usually represents business transactions that must be executed as a unit. Long response times demand support for *persistence* to allow rollback and compensation in case of failure. The WBIS environment uses MQ Workflow to support human interaction activities. BizTalk uses Human Workflow Services (HWS), which was introduced in BizTalk 2004. Previous versions (2000 and 2002) did not have tools for human workflow.

Business rule engine: This tool offers further support for business analysts, who need to add, edit, or delete business rules that are applied to the business processes. The engine is usually designed over the Unified Modeling Language (UML) to allow the business analysts to define business rules and policies at a higher level of abstraction. The software developer can then use these polices while developing the actual process workflow. The WBIS environment uses the Modeler tool as a business rule engine. The Modeler tool is not currently part of the WBI Server Express tool suite. BizTalk uses the Business Rule Composer to define rules and policies, and the business rule engine to run them. Both were

introduced in BizTalk 2004. Previous versions (2000 and 2002) did not have business rule support.

To summarize, the IDE is the software environment over which business personnel collaborate and share their expertise. IDE provides tools for software development, maintenance, and administration.

3.4 Business integration runtime components

The *runtime* components of the business integration platform allow the execution of the developed artifacts. The runtime can be roughly divided into four components: server, adapters, message-oriented middleware, and databases.

Server

The *server*, also referred to as the *broker* or the *hub*, is the heart of the business integration system and provides the runtime environment for the artifacts created by developers. Serving as the hub in the hub and spoke architecture, the server is the central point through which messages flow, transformation is made, and workflows are executed. The server uses message-oriented middleware to talk to adapters, which in turn, talk to applications and data sources. The server uses persistent storage (usually implemented over a relational database) to save messages, process states, and other runtime variables. The database also provides the transactional support required for business process execution.

The server supplies the scalability, high availability, and parallelism required by the nature of the business environment.

The IDE usually supplies a deployment process in which developed artifacts are made available to the runtime part of the business integration platform. Using the deployment process, users can add, develop, and remove runtime components from the server. The server also supplies monitoring feedback and logging for the system administrator.

Adapters

The business integration platform uses adapters to talk to applications, processes, and data repositories of all kinds. The adapter's main task is to adapt application protocols and data formats into formats and protocols understood by the server. Adapters are software components with two interfaces. The adapter uses one interface to interact with a specific application, and the other interface to interact with the server. Both BizTalk and WBIS supply numerous adapters to talk to a huge variety of applications, data storage repositories, and other processes.

During its runtime, the adapter can transform data structures that are recognized by the application to structures recognized by the server, and vice versa.

Message-oriented middleware

Message-oriented middleware (MOM) is a term used to describe the communication infrastructure between the server and the adapters. There currently exist many protocols and communication technologies that can serve this type of communication. These protocols must address a variety of communication characteristics such as performance, reliability, ordering, persistence, scalability, and so on.

WBIS has built-in support for three communication protocols: Common Object Request Broker Architecture (CORBA), Message Queuing (MQ), and Java Messages Service (JMS).

In BizTalk, the communication between the different components is done programmatically (COM in 2002 and 2000), or via Message Queuing (MSMQ), or via database (SQL Server).

Database

The business integration platform must use some kind of persistent storage for saving messages and state, as well as transactional support. The technology of choice is usually a relational database. The server uses the database to store all long-lived information. The server manages a set of database tables to control the start and stop of runtime processes as well as the deployment process. Business processes are usually long-lived, state-full and require reliability. The database's built-in transactional capability provides the needed support to fulfill these business process requirements.

In summary, the runtime part of the business integration platform is a software layer that sits between the application and the operating system, and supplies the runtime services required by the business integration application. This layer is composed of a set of mature software components supplied by the business integration platform provider.

3.5 Side benefits of business integration platforms

By using a business integration platform, companies gain insight, indicators, and new views into their business activity; these were often not envisioned prior to platform installation.

3.5.1 Business activity monitoring

Business activity monitoring (BAM) provides operational information for business processes. Generally used to display real-time information and operational metrics, BAM can provide key insight into how these processes impact business objectives. This enables analysts to adjust resources or processes to more effectively achieve business goals.

4

Migration considerations

In this chapter we provide a comprehensive discussion of diverse migration issues. We start with a definition of migration and a description of the SMB characteristics related to business integration platforms. We then analyze some possible migration scenarios and drivers. After defining the various migration technology options, we provide a set of possible manual and automatic migration techniques. We conclude with a short list of commercial migration technologies provided by other organizations.

4.1 Migration definition

Migration is the process of transferring software components that run under one hardware/software environment to another environment. The purpose of migration is to preserve the integrity of the software components and to retain the users' ability to retrieve, display, and otherwise use them the same way they did prior to the transfer.

4.2 Small and medium business environment

Although migration methodologies and techniques are somewhat orthogonal to the market environment in which they are being used, it is still important to understand SMB market characteristics and trends. This understanding will help us better analyze migration forces and drivers, and adjust the migration tools for those who will use them.

The size of a business can be categorized by either the number of employees or its financial strength. A business is usually referred to as small if it employs fewer than 1000 workers or has less than $10M annual revenue. Medium businesses are those that employ up to 5000 workers and have less than $100M annual revenue. Businesses larger than that are categorized as enterprises. It should be noted that these classifications are given only as a point of reference, and there is no clear definition for categorizing businesses by their sizes.

This discussion is mainly concerned with the IT characteristics of the SMB. Most SMBs have more than one location, but only about 20% have an international location. Almost all SMBs have a LAN environment and a Web site. About two-thirds host their Web site externally. The applications essential to the SMB segment are backup and recovery, database management, accounting, integrated e-mail, and security. Content and document management, together with customer relation management (CRM), data mining, and Web site activity analysis, are expected to be the strongest considerations for the near future.

In terms of operating systems, Microsoft has a significantly higher penetration than all other platforms. Nevertheless, about 25% of SMBs use the IBM AS/400® and an additional 25% use AIX® and Linux.

Problem determination and software administration are the two most time-consuming activities for the IT personnel of the SMB. Software installation and upgrade consume the least amount of time. Visual Basic® is by far the language of choice for the SMB market; followed by C++, Java, and XML. It is important to note here that Microsoft recently decided to end its support for Visual Basic and is pushing the market into the .NET and C# world.

About 65% of businesses in the SMB market use infrastructure software that includes an integration platform; about 35% use more than one infrastructure platform in their business.

IT personnel in the SMB market acknowledge a lack of experience mainly at the infrastructure software level, and are more confident in their knowledge of operating systems and application software. This lack of experience is probably the reason why SMBs are so open to outsourcing work pertaining to their IT infrastructures. Solutions for the SMB market may come from three major sources: software manufacturers, solution providers (SPs), or in-house development. In most cases, the solutions comprise components purchased from, or developed by, multiple sources. Migration tools can be of assistance for the solution providers and in-house developers, as well as outsourcing companies.

About half of the SMB market is sitting on the fence with regard to changes in software infrastructure. For many companies, alternative software capabilities based on open standards usually stem from financial considerations, and could result in an infrastructure change.

In general, SMB IT managers tend to feel a close relationship with business management and perceive IT as central to the company's business.

To summarize, although companies in the SMB market can be treated as scaled-down versions of larger businesses, they have a somewhat different set of issues and emphases. This book takes a special interest in the role played by SPs in the SMB market, and the willingness of SMBs to outsource their infrastructure software. An additional important issue is the time spent by IT personnel on problem determination and software administration, which implies the need for easy-to-use, self-managed software tools.

4.3 Migration scenarios and drivers

To design and build a migration methodology and tools, it is important to understand the scenario under which an SMB may decide to migrate from its BizTalk environment to WBIS. It is also important to designate the target users under each scenario and address their expected requirements and concerns. Three types of migration scenarios may come into play when considering migration projects:

1. A solution provider who needs to support SMBs with an IBM-based infrastructure may wish to develop a WBIS solution based on its existing BizTalk solution.

2. The SMB itself decides to switch its infrastructure and to move to IBM-based solutions.
3. The SMB decides to outsource its infrastructure to IBM.

Each scenario can emerge from different drivers and will involve different users.

4.3.1 Solution providers

Solution providers (SPs) are software development companies that develop solutions for other businesses. SMBs rely heavily on SPs for business solutions. The SPs try to reuse as many software assets as possible for solutions supplied to different SMBs. Supplying migration tools for SPs will allow them to more easily migrate their current BizTalk solutions to the WBIS environment. By doing so, SPs increase their chances for cooperation with IBM, and becoming an IBM partner, a status that can bring great benefits. In addition, reusing BizTalk solutions for WBIS will allow the SPs to increases their market share and profit.

Users in this scenario are the SP staff. SP personnel are assumed to have high-level software development skills and a good understanding of the specific solution from which they want to migrate. It is also expected that for the SP, a migration process will not be a one-time process but an ongoing one. A good migration toolset will allow the SP to continuously maintain solutions for the two environments.

4.3.2 In-house development

Small or medium businesses may decide to change their business infrastructure from a Microsoft-based environment to IBM. This decision may stem from a variety of reasons, such as:

- Cost reduction opportunity presented by using the Linux operating system.
- Superior feature set of the IBM infrastructure. This case is likely to occur when a medium business grows to the point where a real enterprise infrastructure is needed.
- The Microsoft-based infrastructure no longer addresses the business's needs. This is especially true for a business that uses BizTalk 2002. The limited support for this BizTalk version, and the lack of backward compatibility of newly developed adapters with the old BizTalk server, may force the business to migrate from its BizTalk environment.
- The need of the business to align its infrastructure with an enterprise business partner, with which it has very close and intimate business activities, may drive the need for infrastructure change.

Once the decision is made, the business needs to migrate its applications and software assets from the BizTalk environment to WBIS. The company's internal resources and personnel will work with the migration methodology and toolset to successfully accomplish the migration task. In this scenario, it is expected that SMB personnel will have limited development skills and knowledge of the WBIS infrastructure and toolset. It is also expected that the migration in this case will be a one-time process, and that users will not be willing to invest a lot of time and effort in the migration toolset. The migration tools, therefore, need to be comprehensive and mature.

4.3.3 Outsourcing

Outsourcing is the process by which the management and maintenance of a business infrastructure is transferred to a third party. Market research shows that 25% of SMBs are likely to outsource their infrastructure, with an additional 50% who will consider the option.

After winning an outsourcing deal, IBM may wish to migrate the SMB's Microsoft environment, including the BizTalk applications, into an IBM and WBIS-based environment. The main driver in this migration scenario is cost reduction for IBM. Migration also has the advantage of better addressing business needs and enabling better control over the IBM environment and products.

The main users in this scenario are IBM personnel. It is expected that the IBM team will be assisted by the business's personnel during the outsourcing project. The IBM outsourcing team is expected to have WBIS development experience and be familiar with the migration methodology and toolset. The SMB's staff, on the other hand, are expected to have knowledge of, and experience with, the BizTalk environment in general, and with the specific application to be migrated. They will probably be less familiar with the WBIS environment and the migration tools.

4.4 Listing migration options

There are a number of levels at which users can migrate a BizTalk-based application to the WBIS environment. Each has particular advantages and disadvantages, at both the technical and business levels. Migration is possible at the following levels:

- IDE level
- Runtime level
- Operating system level

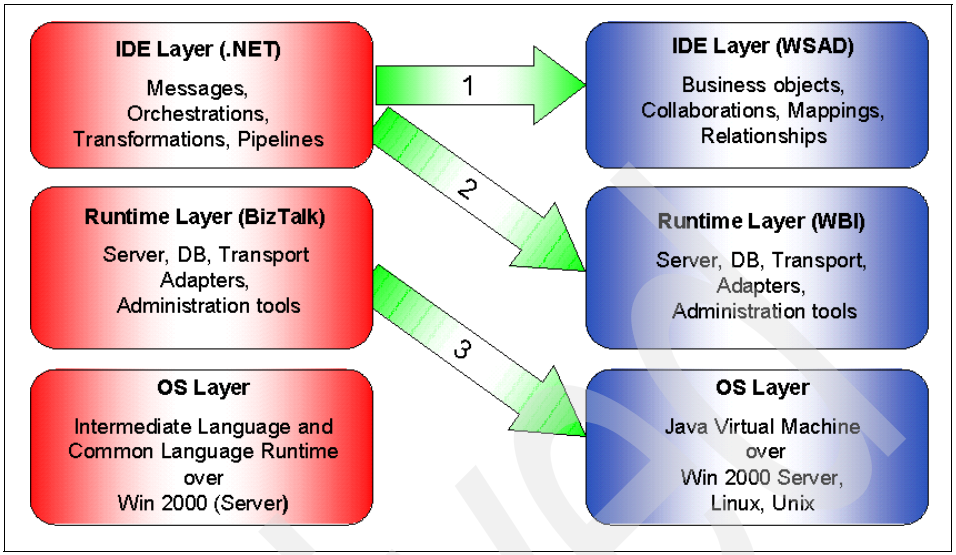

Figure 4-1 Migration options

4.4.1 Migration at the IDE level

Migration at the IDE level means that the set of resources and components that were created in the BizTalk development environment are migrated to their counterparts in the WBIS development environment. Resources and artifacts from the BizTalk environment are exported to an intermediary format and then imported into the WBIS IDE. Users can then continue to use the application by utilizing the WSAD-based tools from the WBIS environment.

The main advantage of migration at the IDE level is that the whole application stack is migrated. Once migration is complete, the user is free to work in the WBIS environment, while the BizTalk server and the tools are no longer needed. Users can use development tools, as well as control and administration tools, from the WSAD-based IDE.

The main disadvantage when migrating at the IDE level is that it requires some manual work by the user. It is not currently anticipated that the migration toolset will provide a fully automated migration process. However, it is realistic to expect that manual work will be reduced to a minimum.

4.4.2 Migration at the runtime level

Migration at the runtime level means that a set of plug-ins in the BizTalk development environment allows users to design and develop the application

under that environment and then deploy it into the WBIS runtime environment. This approach presents a major technical challenge. Deep access to Microsoft's development tool internals is required in order to seamlessly integrate the deployment plug-ins into the environment.

The main advantage of migration at the runtime level is that migration becomes transparent from the user's perspective. The user can continue working with existing assets and is not required to change anything in the development process and tools. However, although the development tools remain familiar to the user, the user still needs to work with a new set of tools for controlling and administering the runtime part of the platform.

Migration at the runtime level has other significant drawbacks:

- Microsoft currently bundles its software for business process development with the BizTalk package and not with .NET. This means the customer must purchase both BizTalk Server and .NET packages for the development part, and also purchase WBI SE for the runtime part of the integration platform.
- Allowing the customer to continue using a Microsoft development environment and tools reduces the chances that IBM software will gain market share in the SMB segment.
- One of the SMB's major characteristics is limited knowledge in complex IT infrastructure. It is unlikely that such a hybrid solution will by easily adopted by the SMB customer.

4.4.3 Migration at the operating system level

Migration at the operating system level is theoretically possible:

- By converting the .NET bytecode represented by the Intermediate Language into Java bytecode understood by the JVM™
- By developing a .NET environment on platforms other than Windows

4.4.4 Migration versus interoperability

There may be cases where a business decides to add WBIS products into a working BizTalk environment. This may be due to unique features that exist only in the WBIS suite, or due to a larger infrastructure switch that requires a period of coexistence for the two products. Coexistence can be achieved by having the WBIS talk to the BizTalk server using a special adapter. A decision has to be made regarding whether the two servers would share the same MOM and database.

Although interoperability raises many interesting questions, it does not actually require a migration from one environment to the other, and thus is beyond the scope of this redbook.

4.5 Project assessment and manual migration

Migrating a BizTalk application to the WBIS environment is not expected to be a fully automated process; some human resources will definitely be needed. The ability to correctly analyze the project scope and provide a rough estimate of the needed resources is crucial for making the right migration decisions and for the success of the migration project.

As the first step in the migration process, the *assessment* phase should provide some insights for the migration project plan and scope. Understanding the application's architecture and the complexity of the various migration techniques at hand is important for project sizing and scoping. In addition, you must define the project plan, including a proof of concept, migration order, and checkpoints.

The first assessment phase should offer a clear understanding of the source application's high-level architecture and a list of components used by the application. The list should include the various adapters, messages, transformations, and orchestrations used by the application. The second assessment phase can then be executed on the artifact list to supply a rough sizing of the migration efforts needed. The sizing is based on analyzing the artifacts' code and consulting a knowledge base to estimate the amount of work required.

It is sometimes important to assess and analyze the runtime performance of the source system in order to select hardware for the target platform. A knowledge base can help in estimating the hardware needed to address the performance requirements, based on experience from previous projects.

4.6 Building and using the knowledge database

The knowledge database stores rules, matrixes, past experience, best practices, and preferences. It also stores other information relevant to the migration process, from the analysis and assessment phase, through the migration planning, and on to the actual migration process. Users can retrieve this knowledge during the current migration process through a set of application program interfaces (APIs) or using a browsing tool.

The information in the knowledge database is divided into several sections, according to the information's role in the migration process. The section list may include the following:

- **Migration best practices** - A list of past experiences and best practices learned from previous migration projects that can be helpful for this migration project and for future ones.
- **Migration issues** - A list of issues found to be problematic when migrating to the WBIS environment, such as use of external resources.
- **Migration techniques** - Available migration techniques that can be used in certain situations, for example, to transform a well-known BizTalk interface to its WBIS counterpart.
- **Runtime knowledge** - Information gained during migration of one artifact that can be helpful while migrating another artifact. For example, while migrating from BizTalk message to WBIS Business Object, the field mapping must be saved so that it can be used for the migration from the BizTalk transformation to the WBIS mapping.

4.7 Migration techniques and automatic migration

Various migration techniques and concepts must be used to address the complexity of the operational environments over which the BizTalk application is expected to run, and the variety of artifacts and other software components from which it is expected to be composed. There may be cases where a simple translation of an artifact to the other platform will be enough; there will probably be other cases where architectural changes will be required. The following sections list the various migration cases, along with the migration techniques and concepts required for each case.

4.7.1 Export-import and XML translations

Export-import is the most straightforward technique for migration and relies on the fact that most state-of-the-art IDEs save their artifacts in XML format in the file system. A standards-based platform exports its artifacts in XML format compatible with the relevant standard, while other platforms may use proprietary structures. If migration is needed from one standards-based platform to the other, it is possible in an ideal world to export the artifact from the source application and import it to the target application. For example, the Business Process Execution Language (BPEL) is an XML-based standard for describing business processes and workflow. When this standard reaches maturity and business process platforms have full support and compliance with it, it will be possible to take business process artifacts from one environment (for example,

BizTalk Orchestration) and import them into the other environment (for example, WBIS Collaboration).

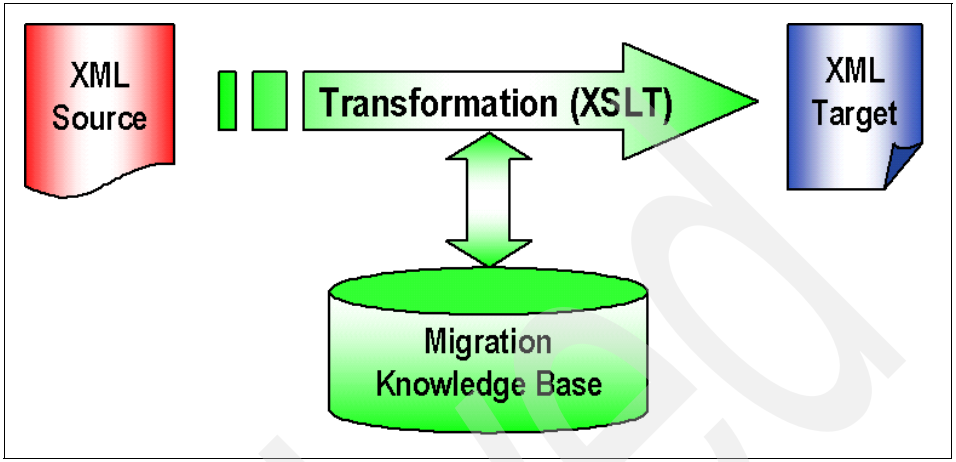

Figure 4-2 XML translation technique

In a less than ideal world, where standards exist but are not yet fully supported by platforms, it is still possible to build on the XML structure of the artifacts and on the XML Stylesheet Transformation Language (XSLT) to translate the proprietary and non-standard parts of the artifact from the source platform into their counterparts on the target platform (Figure 4-2).

4.7.2 Interface adjustments

In some cases, business integration platforms supply developers with the means to add custom code to the component behavior. Custom code usually has a known, well-defined programming interface that can be used by the calling process. By emulating this interface on the target platform, it can execute the custom code developed for the source platform.

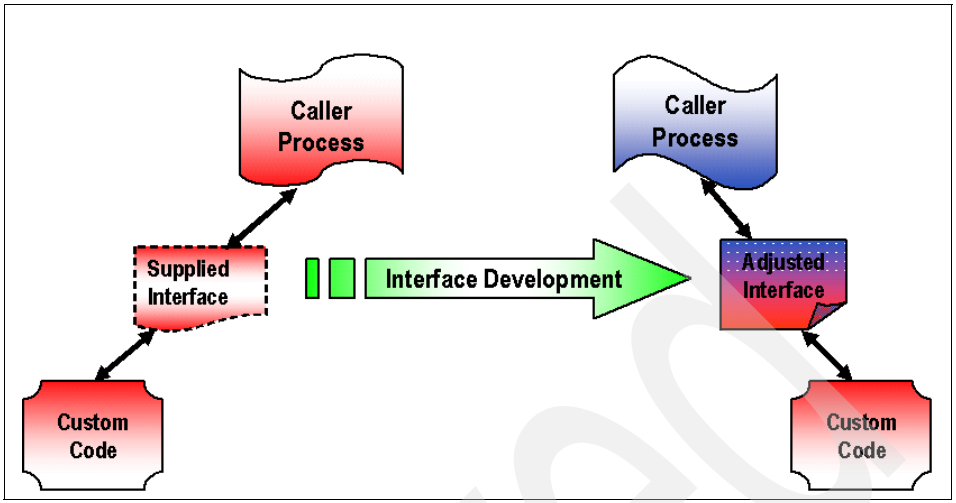

Figure 4-3 Interface adjustment technique

For example, when developing a transformation from one BizTalk message to the other, users can add *functoids*, which are pieces of code that implement sophisticated transformations. A functoid has a well-known, well-defined interface to the caller transformation process.

Whereas BizTalk uses functoids for transformation, WBIS uses mapping to transform from one business object to the other. WBIS also supplies users with an interface for custom code execution. By implementing an adapting interface that adjusts the supplied WBIS interface to the interface implemented by the BizTalk functoid, users can continue to use the custom transformation code without changes. It is important to note here that the technique shown in Figure 4-3 only applies to applications running on the same operating system. Users may need to build the custom code for the target application if the source operating platform differs from the target; for example, if the custom code is written in C for Windows and needs to be run under Linux. There may even be cases where users will have to rewrite the custom code if the target application does not support the programming language of the custom code. This may be the case if code is written in one of the programming languages supported by Windows but not by Linux.

4.7.3 Black-box technique

The *black-box* technique is similar in nature to interface adjustment, but different in scale. There are cases where business integration platforms provide interfaces to run external processes as sub-processes for special tasks. In cases where the source application uses this kind of interface to execute specific sub-processes and tasks, a similar interface may be added to the target

application to allow invocation of the same tasks. For example, the BizTalk application uses an XSLT transformation to transform messages from one XML format to the other. You can add an interface to the WBIS collaboration that will allow you to run an XSLT engine. You would then be able to send the original XML message through the interface and get back the converted message. This step can be accomplished by using the BizTalk XSLT transformation, with no changes whatsoever.

4.7.4 Using adapters

The main purpose of the business integration platform is to integrate stand-alone applications into a comprehensive infrastructure. BI platforms use adapters to adjust connectivity to the various applications. There may be cases where new adapters will be needed on the target platform in order to communicate with adapters on the source platform. This may be practical when communicating with the adapter on the source platform is easier than rewriting or changing it.

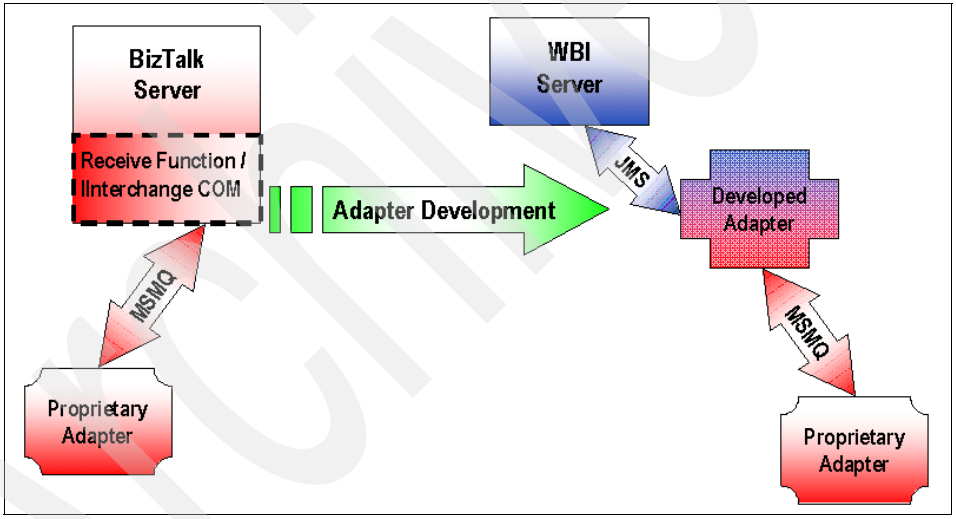

Figure 4-4 Using adapters for migration

Say, for example, an SMB has a custom adapter for accessing a legacy system. This adapter consists of application-specific code that interacts with the legacy system and submits messages to the BizTalk Server. When migrating the custom adapter to the WBIS platform (Figure 4-4), one alternative could be to rewrite the adapter components that interact with BizTalk, and replace them with new code that interacts with WBIS. Alternatively, one could leave the custom adapter “as-is” and write a brand new WBI adapter that will appear as a BizTalk Server to the custom adapter (for example, it implements the expected interface), and as a regular WBI adapter to WBIS.

4.7.5 Component substitution and configuration

Component substitution is a technique in which a component from the source platform can be substituted with its counterpart from the target platform. This technique may apply to small, well-defined components, such as BizTalk functoids, as well as to large components, like the BizTalk SAP adapter. There may be cases where a new WBIS component is developed to match existing BizTalk functionality not currently available in WBIS. Once this component is developed, it can be used in further migration projects.

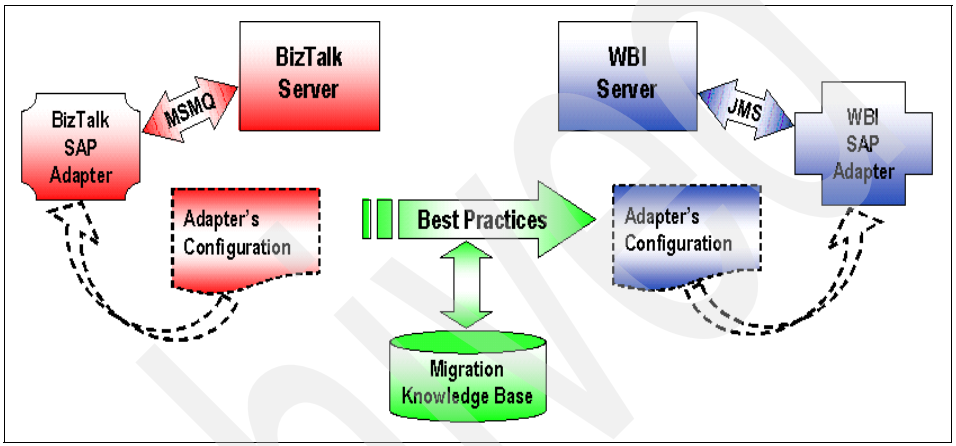

Figure 4-5 Component substitution

In these cases, supplemental component configuration may be needed to adjust the new WBIS component for the runtime environment. For example, if we substitute the BizTalk HL7 adapter with its WBIS counterpart (Figure 4-5), a configuration may be needed to adjust the new adapter behavior for its working environment.

4.8 Other migration techniques

We started this section by describing three migration options. Aside from the IDE-level migration, the options discussed here do not migrate the entire application software stack, and consequently necessitate a hybrid development and runtime environment. But to fill in the complete picture, this section describes additional migration techniques that are suitable for the other two migration options.

4.8.1 .NET plug-in development

Ideally, BizTalk developers would like to migrate from BizTalk to WBIS without having to change the development environment. Migration would then be a transparent process that requires no changes. Users could continue developing BizTalk applications, while WBIS runtime artifacts would somehow be generated behind the scenes. This option presents a major drawback from an IBM business perspective since it will allow the developer to continue working with a Microsoft-based development environment.

This concept, illustrated in Figure 4-6, is based on the development of a .NET plug-in that is responsible for generating the WBIS runtime artifacts. Developing this .NET plug-in is expected to be a very challenging task. First, it requires a deep understanding of both .NET and WBIS internals. Second, more than simply generating the equivalent WBIS components, BI application migration must mimic the way these components will be invoked. Third, users would still need to manage and monitor the runtime application using WBIS tools.

Although .NET is a proprietary technology, there is an open source effort (known as the Mono Project) that provides an implementation of the .NET development environment for free platforms. The insights provided by this open source effort would be helpful when developing the .NET plug-in.

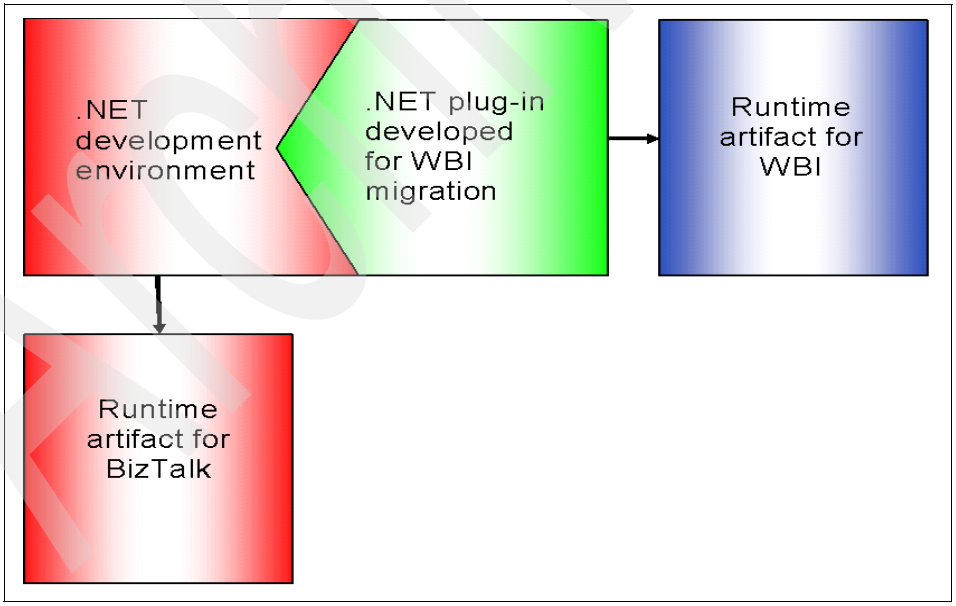

Figure 4-6 Plug-in development

4.8.2 Intermediate language to JVM migration

One option that would allow migration from the Microsoft-based .NET environment into the Java world is to translate the Intermediate Language (IL) created by the .NET back into Java source code. This code can then be compiled with a Java compiler and run on a Java Virtual Machine (JVM). The big advantage of this technique is that it allows migration of any .NET code to the Java world. In fact, there are several companies that actually build such solutions (for example, Mainsoft and Stryon). The main disadvantage of this approach, on the other hand, is that it requires much broader, high risk, development efforts. In addition, this does not provide a migration from BizTalk to WBIS, but rather a way to run BizTalk on operating systems other than Windows.

4.9 Existing migration technologies

Some of the techniques discussed here involve migration technologies developed either as free, open source efforts or as company products. This section provides a short description of these technologies and their corresponding Web sites.

4.9.1 Mainsoft

Mainsoft is a cross-platform development company that extends the Visual Studio environment to multiple platforms (J2EE™, UNIX, and Linux). Visual MainWin introduces a compiler that compiles Microsoft Intermediate Language (MSIL) into Java bytecode. Developers can write programs in Visual Basic .NET or C# and compile their source code directly into Java bytecode.

More information on Mainsoft is available from http://www.mainsoft.com.

4.9.2 Stryon

Stryon provides software transformation solutions. The company offers migration products, such as iNET, that enable developers to migrate applications developed in Microsoft-based environments so they execute in Java-enabled environments. iNET is a Java-based implementation of the Microsoft .NET framework.

More information on Stryon is available from http://www.stryon.com.

4.9.3 The Mono project

The Mono project is an open source effort, sponsored by Novell, to create a free implementation of the .NET development framework for UNIX/Linux platforms. Mono provides a C# compiler, a Virtual Execution System that has a Just-In-Time compiler, garbage collector, loader and threading engine, an implementation of the .NET class libraries, and so forth.

More information on the Mono project is available from http://www.go-mono.com.

4.10 Summary

This chapter defined possible migration techniques and options. The next two chapters describe the BizTalk and WBIS fundamentals. In the following chapters, we use the technique definitions described here to discuss the actual migration of various artifacts and components developed for the BizTalk architecture into the WBIS environment.

5

BizTalk fundamentals

This chapter describes the BizTalk 2002 product. (The first version, BizTalk 2000 is very similar to the second one, BizTalk 2002). Since this redbook concentrates on migrating BizTalk 2002 artifacts into WBIS, the present discussion focusses on the 2002 version.

5.1 BizTalk Server 2002

BizTalk Server 2002 provides integration technology for both enterprise application integration (EAI) and business-to-business (B2B) integration. It helps users integrate applications and automate business processes without excessive coding. BizTalk 2002 is built entirely from COM components, so it can be programmed and extended. It provides a set of independent tools and some support for standards such as XML and XSLT; however, it does not support BPEL or XSD.

5.2 BizTalk 2002 architecture

A BizTalk application can be divided into two independent processes (Figure 5-1):

- **Messaging** - BizTalk's runtime support for message routing from source to destination
- **Orchestration** - BizTalk's runtime support for business workflow

Messages from the messaging process can be routed into the orchestration process and back. Messages can also be routed by the messaging process directly from source to destination, bypassing the orchestration process.

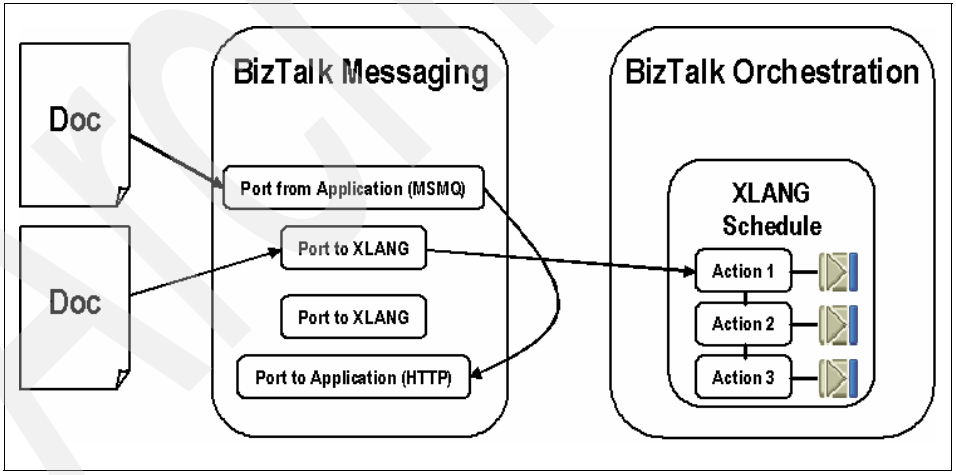

Figure 5-1 BizTalk high level architecture

5.3 BizTalk messaging and application connectivity

The messaging part of the BizTalk solution is used primarily for routing messages from one or more sources to one or more destinations (Figure 5-2). BizTalk messaging has support for rich routing features like many-to-one, one-to-many, content-based routing, and so on.

BizTalk messaging also deals with building the messages that actually carry the business data, filtering them if needed, and transforming messages from one format to another.

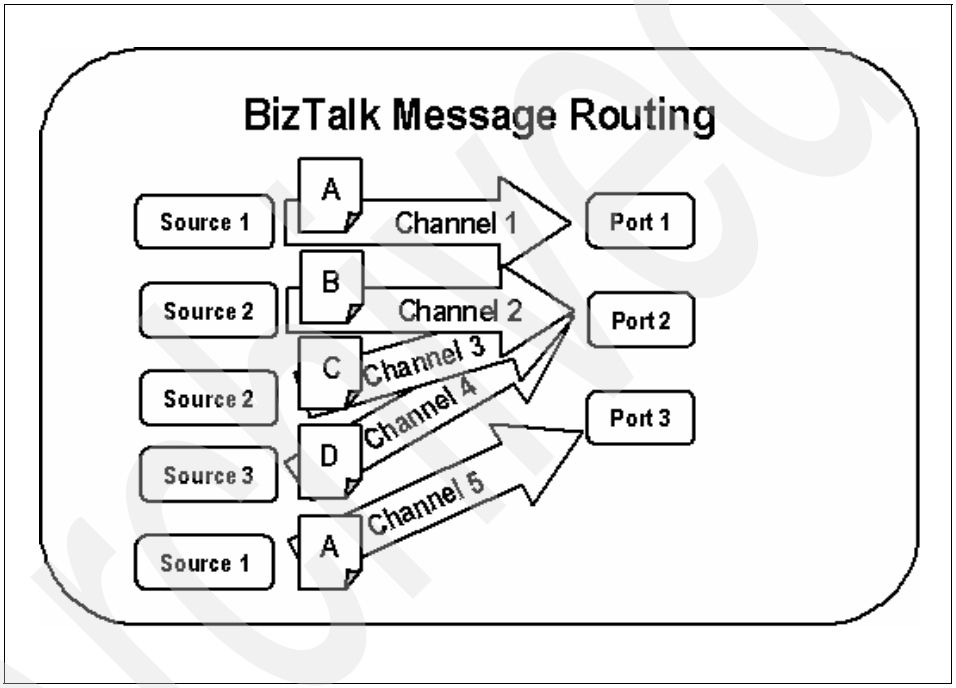

Figure 5-2 BizTalk message routing

5.3.1 BizTalk transformation

BizTalk transformation lets you transform one message format to another. The BizTalk transformation artifact is based on XSLT and supports custom transformations. The BizTalk channel is responsible for executing the BizTalk transformation on an input message and generating the output message. This document discusses message transformation in Chapter 11, “Migrating transformations” on page 121.

5.3.2 BizTalk orchestration

The BizTalk orchestration process is used for creating business workflows. BizTalk orchestration comprises two stages:

- Specifying the process flow between processing units
- Specifying the data input and output for each processing unit

The internal processing capabilities of the BizTalk orchestration tool are fairly limited. Any required logic must be implemented using external components.

BizTalk supports four types of interfaces to external components:

- **COM interface**: BizTalk orchestration has a proprietary interface by which it interacts with external COM components.
- **WSC interface**: BizTalk orchestration has a proprietary interface by which it interacts with Windows Scripting Components (WSC).
- **Message Queuing interface**: BizTalk orchestration is able to read and write messages into the MS Message Queuing (MSMQ) component.
- **BizTalk messaging interface**: BizTalk orchestration is able to read and write messages to and from the messaging process of the BizTalk server.

In a typical BizTalk orchestration, orchestration receives data through one of its ports and passes it to one or more processing units, using one of these interfaces. It then receives the result and sends it out to its destination (either directly or through the BizTalk messaging tool).

5.4 BizTalk 2002 components

The BizTalk Server 2002 engine provides two main services: BizTalk Orchestration, through which you specify and execute your business processes, and BizTalk Messaging, through which you transform, route, and send messages between business processes. These services are supported by a set of data stores for configuration information, message tracking, and so forth.

The BizTalk Server Engine, as shown in Figure 2-2 on page 8, is composed of the following main components:

- XLANG schedules
- Database
- Receive functions and pre-processing
- Channels
- Ports

- Messaging ports and distribution lists
- Custom adapters
- Custom components

Details about these components are presented in the following sections.

5.4.1 XLANG schedules

A business process is implemented as one or more orchestrations (called XLANG schedules in BizTalk 2002). Orchestrations represent the logic behind a business process, and serve to direct the flow of information and specify the set of actions to be taken. Orchestrations are created using a graphical tool (the Orchestration Designer, which is Microsoft Visio®-based), which generates the corresponding executable code. Orchestration Designer for BizTalk 2002 supports the creation of moderately complex business processes. Instead of using BPEL to represent the orchestration, BizTalk 2002 uses XLANG, Microsoft's proprietary XML mark-up for business process integration design.

All messages processed by orchestrations must be XML messages. Therefore, incoming messages, regardless of the original format, will be converted to XML before being transferred to the corresponding orchestrations.

You should understand that an XLANG schedule is submitted by the developer to the XLANG Scheduler Engine. The XLANG schedule is a COM+ application that is part of the BizTalk Server. The XLANG Scheduler Engine directs the instantiation, execution, dehydration, and rehydration of running XLANG schedule instances. When an XLANG schedule instance becomes dehydrated, the current state of that instance is carried through to an SQL Server persistence database (known as XLANG). Once a schedule has been instantiated, the XLANG Scheduler Engine continues the business process sequence until it becomes necessary to dehydrate the XLANG schedule. Dehydrating an XLANG schedule occurs when the XLANG Scheduler Engine detects that the XLANG schedule is blocked and waiting for a message to arrive. Upon arrival of the message, the XLANG Scheduler Engine rehydrates the XLANG schedule instance. It then continues to perform the actions in the business process sequence until the schedule either completes or the XLANG Scheduler Engine has to wait for another message.

5.4.2 Database

A number of SQL databases are used to store configuration and status information:

- The **BizTalk Messaging Management (BTM) database** stores the configuration for BizTalk Messaging and all BizTalk Server machines in the same BizTalk Server group. This database has low activity, since configuration information is cached.
- The **Document Tracking and Activity (DTA) database** stores message tracking information. Messages can be tracked by type or by individual instances. This database grows rapidly and is optimized for writing.
- The **Shared Queue (SQ) database** is the core database within the BizTalk Server. All documents are persisted to this database while being processed. It consists of the following logical queues:
  - Work - Messages currently in process
  - Scheduled - Messages processed and awaiting delivery to a distribution list
  - Retry - Failed messages awaiting a retransmission attempt
  - Suspended - Messages that could not be transmitted

  This database is extremely active, both for reading and writing.
- The **Orchestration persistence (XLANG) database** stores dehydrated schedules and is used to persist each step of a running XLANG schedule. The completed XLANG schedules are removed via scripts. As a result, this database has heavy read and write activity.

5.4.3 Receive functions and pre-processing

Receive functions are COM-based components that support the submission of messages from external systems to BizTalk. Receive functions are the starting point for documents submitted to BizTalk Server. They can monitor for any type of document and pass it to any channel.

BizTalk 2002 comes with three built-in receive functions:

- **File receive function** - Monitors events from the file system and is configured to detect documents arriving from other applications. The documents can be transported into the file system by any mechanism that can write a file to the directory (for example, FTP). The file receive function reads an input file and creates a corresponding BizTalk message. The message is in turn processed by the channel, stored in the work queue (SQ database), and finally the input file is deleted.

- **HTTP receive function** - A configurable Internet Server Application Programming Interface (ISAPI) extension that supports receiving documents via HTTP protocol. Documents are submitted either to the file system or to BizTalk through a COM call. Both synchronous and asynchronous calls are supported, but the default is asynchronous.
- **Message queuing receive function** - Supports receiving messages by using the proprietary Microsoft Message Queuing (MSMQ).

COM-aware applications may submit documents directly to BizTalk via the IInterchange COM interface. This mechanism enables applications to block on submission (SubmitSync method) or submit without waiting for delivery (Submit method). The status of queues can be checked via this interface, but no retrieval method is provided.

BizTalk Server provides interfaces that allow developers to create custom pre-processing components. These components process the message picked up by the receive function prior to submitting the data to BizTalk Server for processing. Custom pre-processors for receive functions must implement the interfaces IBTSCustomProcess and IBTSCustomProcessContext.

5.4.4 Ports

Ports are an abstraction of an application that communicates via a specific protocol, on a specific network location, and using a specific format. The port is a directed connection between BizTalk Server and an application. Ports may have pre-configured routing information, or may be left open, in which case routing information is collected from the message header.

In an XLANG schedule, ports facilitate synchronous and asynchronous communication and are used to pass messages into or out of the schedule. They are, in essence, named locations that use a specific implementation (for example, message queue, COM invocation, and so forth).

5.4.5 Channels

Channels are the combination of a port with transformation rules. They describe the document source and the processing steps to be performed before the document is delivered to the associated messaging port or distribution list. When BizTalk Server 2002 receives a document, it locates the appropriate channel to process the document. After the channel processes the document, it points the server to its associated messaging port or distribution list, which directs the server through the sequence of steps necessary to envelope, secure, and transport the document to the specified destination.

The routing process, by which BizTalk selects the channel or channels that will handle an incoming message, can be call-based or self-routing. Call-based routing means that the source, target, and document are all explicitly declared as parameters when invoking the IInterchange interface. Self-routing means the routing information can be found in the content of the document itself.

5.4.6 Messaging ports and distribution lists

Messaging ports send documents to XLANG schedules or to target applications, which can be either a trading partner or an application within the same organization. A messaging port to an XLANG schedule can be configured to activate a new instance of a specified XLANG schedule, and then deliver the document to a specified port of that schedule. In this case, the specified schedule must contain a port that is bound to BizTalk Messaging Services. When users specify the schedule, they also name the port. Users can configure a messaging port to deliver a document to a running instance of an XLANG schedule. In this case, the document must contain a queue name to which the document should be delivered and that the targeted schedule is monitoring.

An open messaging port does not have a specified destination. The missing information must be provided in the message.

A distribution list is a group of messaging ports with which the same document (submitted only once to BizTalk) can be simultaneously sent to several target applications. To achieve this, message ports must be created for each target application and these message ports must be added to the distribution list.

5.4.7 Custom adapters

Hundreds of BizTalk 2002 custom adapters are available for a varied range of systems. These custom adapters do not follow a standard API, since the Adapter Framework was only introduced with BizTalk Server 2004.

Custom adapters submit documents to BizTalk either via Receive Functions or programmatically via the IInterchangeObject.

5.4.8 Custom components

BizTalk Server 2002 capabilities can be extended by developers who create their own custom components. Custom components can be created for application integration, encryption, encoding, signing, parsing, and serializing. An Application Integration Component (AIC) is a COM object used by the BizTalk Server to deliver data to an application.

BizTalk provides a set of COM interfaces that must be implemented for each component type:

IPipelineComponent, IPipelineComponentAdmin

- For pipeline components such as: encryption, encoding, signing; or for Application Integration Components (AICs).
- Supports configuration of components and a design-time user interface.

IBTSAppIntegration

- For Application Integration. This interface has a single method, ProcessMessage, that takes a message (string) as input and returns a reply (string).
- Does not support component configuration or a design-time user interface.

IBizTalkParserComponent

- Parsers are responsible for converting documents to XML, and for getting the necessary routing information (source, destination, and document name) so that BizTalk can select a channel.
- Default parsers are provided by BizTalk for XML, Flat Files, Edifact, and X12 documents.

IBizTalkSerializerComponent

- Serializers are responsible for converting documents from XML back to their native format.
- Default serializers are provided by BizTalk for XML, Flat Files, Edifact and X12 documents.

5.5 BizTalk 2002 tools

In this section, we discuss the main BizTalk development and administration tools.

5.5.1 BizTalk Editor

BizTalk Editor is a standalone tool that enables users to create definitions of document types (schemas) based on Microsoft's proprietary XML Data-Reduced (XDR) mark-up language. The tool provides a graphical representation of hierarchical records and fields to represent the underlying XDR definition. Existing XDR schemas can also be imported.

The editor handles records and fields, and is influenced by database terminology. A record corresponds to an entity, and the fields are equivalent to

properties. Records can be nested. When writing an XML-based specification, the records and fields are described in terms of the basic XML data types, as well as the derived types permitted under the XDR schemas technology. All the options available in a schema are available to specification builders. This includes being able to specify the cardinality (repetition) of specific records and fields.

BizTalk Server Editor can handle the following text formats:

- XML
- Flat files (delimited, positional, or hybrid delimited and positional)
- EDI (X12 and UN/EDIFACT)

The specification produced by BizTalk Editor is an XML document that follows an XDR schema, but also has elements (annotations) added by the BizTalk Editor to record specification-specific information. These extra elements are important when working with non-XML specifications. For example, XDR schemas cannot capture delimiter and field position information. It is the extra elements that capture this information, and they are proprietary to BizTalk.

BizTalk Editor provides a collection of basic XML template specifications for common business messages, such as purchase orders and invoices.

5.5.2 BizTalk Mapper

BizTalk Mapper is a stand-alone tool that enables users to define transformations (maps) from a source document to a target document. Each map is expressed as a graphical correlation between two XML schemas to define a relationship between elements in those schemas. The W3C has defined the Extensible Stylesheet Language Transformation (XSLT) as a standard way to express these kinds of transformations between XML schemas. Maps in BizTalk Server 2002 are implemented as XSLT transformations.

The transformation defined in a map can be simple, such as copying an identity field from one document to another. Direct data copies like this are expressed using a link, which is shown in BizTalk Mapper as a line connecting the appropriate elements in the source schema with their counterparts in the destination schema. More complex transformations are also possible using functoids. A functoid is a COM component that can define arbitrarily complex mappings between XML schemas. Because some of those transformations are

fairly common, BizTalk Server 2002 includes over 60 built-in functoids. These built-in functoids are grouped into categories that include the following:

- **String functoids** perform string manipulation (finding strings within strings, string trimming, and so forth) of fields in the source document and store the result in fields in the target document.
- **Mathematical functoids** perform operations such as adding, multiplying, and dividing the values of fields in the source document and store the result in a field in the target document.
- **Conversion functoids** perform conversions, such as a numeric value to its ASCII equivalent and vice-versa, or a decimal value to hexadecimal or to octal.
- **Logical functoids** determine whether an element or attribute should be created in the target document, based on a logical comparison between specified values in the source document. Those values can be compared for equality, greater than/less than, and in other ways.
- **Date/time functoids** manipulate date and time data or add current date, time, or date and time data to a record or field in the destination specification.
- **Scientific functoids** convert a numeric value to a scientific value (for example, sine, cosine, logarithm, and so forth).
- **Cumulative functoids** compute averages, sums, or other values from various fields in the source document, and store the result in a single field in the target document.
- **Database functoids** access information stored in a database.
- **Advanced functoids** perform miscellaneous manipulations such as looping, iteration, scripting, and so forth. With the Scripting functoid, users can script their own function in Microsoft Visual Basic Scripting Edition (VBScript) or Microsoft JScript®.

Custom functoids can be created directly in XSLT or by using programming languages like C++ and Visual Basic. Custom functoids must implement the IFunctoid interface or extend the BizTalk CannedFunctoid class and, as any other custom component, they must be registered prior to use. Functoids can also be combined in sequences, cascading the output of one into the input of another. The output of the BizTalk Mapper is an XML document that incorporates the source and target message specifications, links between source and target fields, and an XSL stylesheet that implements the mapping.

5.5.3 Orchestration Designer

The Orchestration Designer runs inside Microsoft Visio and graphically organizes a defined group of shapes to express the conditions, loops, and other behavior

that define the flow of the business process. After an XLANG schedule is drawn, it can be compiled and run.

BizTalk Orchestration Designer provides a set of flowchart shapes that are used to describe the business process:

- **Begin** represents the start of an XLANG schedule.
- **Action** represents a process that receives a message from a port or sends a message to a port. The send or receive action can be synchronous or asynchronous, depending on the component or implementation to which the port is bound. This shape is a placeholder for a call out to an implementation shape.
- **Decision** represents a process that evaluates one or more rules sequentially. This shape has one inbound flow and one or more outbound flows. Each outbound flow is associated with a rule (a VBScript expression) that evaluates to TRUE or FALSE. The first rule that evaluates to TRUE determines which outbound flow is followed in the business process. The sequence of the business process follows the flow from the first rule that evaluates to TRUE. If no rules evaluate to TRUE, the Else flow is followed. The Decision shape must contain at least one rule. Each rule must contain a script expression.
- **While** represents a process that can be repeated. It contains one rule (a VBScript expression). If the rule evaluates to TRUE, the flow from the rule is followed to completion and then it repeats. If the rule evaluates to FALSE, the Continue flow is followed. When the business process sequence flows from a rule in a While shape, the sequence must conclude in a single End shape.
- **Fork** represents concurrency in a business process. One flow can enter a fork, and as many as 64 flows can leave a fork. Each flow that leaves a fork is executed concurrently. All business process sequences that flow from a *single* Fork shape must connect to a single Join shape or terminate in an End shape.
- **Join** synchronizes concurrent flows in a business process. As many as 64 flows can enter this shape, but only one flow leaves it. Logical operators (and, or) are used to determine how to synchronize the flows:
  - OR - enables the first flow that arrives to continue. The other flows continue to execute.
  - AND - synchronizes all incoming flows before the outbound flow can continue.
- **Transaction** represents a collection of actions that are either all executed, or else none are. Three types of transactions are supported: short-lived transactions (which support transaction re-try), long-running transactions (which can contain nested transactions), and timed transactions that are long-running. By defining transaction properties, users can make available

either the Compensation for Transaction page (for nested transactions) or the On Failure of Transaction page. On these pages, developers can model the error-handling processes that are specific to the transaction.

- **End** represents the completion of one process flow.
- **Abort** represents the termination of execution within a transaction group.

Note: There is no Transform shape in BizTalk 2002. Performing a mapping from within an orchestration requires custom code.

BizTalk Orchestration Designer provides a set of implementation shapes that are used to describe the port implementation (for example, where and how messages are sent/received):

- **COM component** represents a port implementation that supports synchronous communication by using a method call for each message that is sent or received. The port is bound to an interface implemented by a pre-existing COM component that has been registered on the system. A send action that is connected to the port represents the invocation of a specific method call by the XLANG Scheduler Engine. The IN and IN/OUT parameters are sent to the port. In return, a message with a schema that is defined by the IN/OUT and OUT parameters of the method call is received from the port. The method call is supported by the interface that is bound to the port. A receive action that is connected to the port waits for an external application to make a method call to the port. The IN and IN/OUT parameters define the schema of the message that is received by the port, and the IN/OUT and OUT parameters define the schema of the message that is sent from the port.
- **Script component** is same as the COM component, except that the port is bound to an interface that is implemented by a pre-existing Windows Script Component that has been registered on the system.
- **Message Queuing** represents a message queue implementation that supports asynchronous communication. The port can be bound to a named queue or to a per-instance queue. In the latter case, a unique queue is created and used for each instance of this XLANG schedule. Per-instance queues provide a convenient way for an XLANG schedule to have a separate queue for each XLANG schedule instance. When an action is connected to a port that is bound to a message queue, the message that is sent or received by the action must be defined. Because multiple message types can be stored in a queue, the message type helps identify the type of message that should be received by the action. During a Send action, the message type is marked on the label property of the message when it is written to the queue. To receive a message, the XLANG Scheduler Engine requires the name of the queue that is used and the name of the root element of the XML schema

that is contained within that message. To send a message, the XLANG Scheduler Engine needs the name of the queue that is used to transmit messages.

- **BizTalk Messaging** represents a port implementation that uses BizTalk Messaging Services, supporting asynchronous communication. To receive a message, the XLANG Scheduler Engine requires the HTTP URL address used by the BizTalk Messaging Service to receive documents, the name of the channel, and the message type of the outbound document definition. Receiving documents for XLANG schedule activation only requires the message type of the document definition. To send a message, the XLANG Scheduler Engine requires the name of the channel and the message type of the inbound document definition.

5.5.4 Management tools

In this section, we specifically discuss those tools used for BizTalk management.

BizTalk Server Administration

BizTalk Server Administration is a Microsoft Management Console (MMC) snap-in that provides a graphical representation of the BizTalk Server 2002 components—such as servers, message queues, and receive functions—which can be managed by an administrator. This tool enables administrators to perform common administrative tasks.

BizTalk Document Tracking

BizTalk Document Tracking is a stand-alone Web application that enables developers to track documents as they move through various stages of the business process. This tool actually performs queries against the Document Tracking and Activity database (DTA). It is mainly used for trouble shooting. Messages can be filtered based on:

- Date range (the date and time the message transited BizTalk Messaging Services)
- Message source
- Message destination
- Message type

BizTalk Messaging Manager

BizTalk Messaging Manager is a wizard-based tool for configuring the BizTalk Messaging Service to manage the exchange of documents between applications. This tool is used to create messaging ports, channels, distribution lists, document definitions, and so forth. The Messaging Manager actually

accesses the BizTalk Messaging Management database (BTM). A programming alternative to this graphic tool is provided by the BizTalk Messaging Configuration object model application programming interface, which enables developers to automate all, or part, of the configuration process.

BizTalk SEED Wizard

BizTalk SEED Wizard supports partner enablement through the creation and use of SEED packages. Companies (initiators) can package their Microsoft BizTalk Server 2002 configurations into a SEED package and make it available to trading partners (recipients) through the Internet. A SEED package is an XML package containing a company's business document configuration, which is created using BizTalk SEED Wizard. After a package is created, trading partners (recipients) can use BizTalk SEED Wizard to install the configuration within their organizations in order to conduct business with the initiating company on the Web. Companies that create a SEED package (initiators) still have to manually configure BizTalk Server to receive documents. However, by creating a SEED package, trading partners (recipients) are able to configure BizTalk Server, test their configuration, and begin exchanging documents with the initiating company. If the recipient is successful in testing documents with the site, the deployment initiator can integrate the SEED package configuration into the production environment and begin transactions with the recipient.

6

WebSphere Business Integration fundamentals

This chapter provides a detailed description of the WBIS platform. It first describes the various WBIS server families that exist as part of the IBM portfolio, and the supporting transport they are using. It also provides a short description of the relational database used by the platform and the WBI adapters. Next, a comprehensive description of the development, testing, and administration tools of the WBI SE is presented. This section concludes with a description of a set of tools currently not included in the WBI SE tool suite.

6.1 WBI Server Express architecture

In the following sections, we focus our discussion on the WBI Server Express architecture.

6.1.1 WBIS business objects and application connectivity

The WBIS platform implements message routing functionality using WBI adapters. Adapters are responsible for connecting applications to business processes and for transforming application data formats into a standard data format that can be understood by the WBIS runtime. The WBIS architecture is more rigid in its application connectivity aspects and provides fewer routing capabilities.

6.1.2 WBIS mapping

WBIS supports the notation of Generic Business Object (GBO) and Application Specific Business Object (ASBO). The WBIS mapping tool lets you define a mapping from ASBO to GBO and vice versa. The WBIS runtime implicitly executes the required mapping upon receiving an ASBO that requires transformation to GBO or before sending a GBO to an adapter that only understands ASBO. In addition, you can programmatically execute a WBI map to translate a source BO (either GBO or ASBO) into a target BO.

6.1.3 WBIS collaboration

Collaboration is the WBIS terminology for workflow unit. WBIS has a rich development environment for building sophisticated workflows.

WBIS supports two types of interfaces to external applications from within a workflow:

- **Service calls**: This is the standard way in which WBIS interacts with external applications. A service call is an entity within the WBIS process designer. This entity is usually bound to one of the WBI adapters. The service call interface can also be bound to other WBIS collaboration.
- **Java API from activity entity**: WBIS collaboration uses activity entities for workflow execution. WBIS allows you to interact with external components directly from within an activity entity. WBIS collaboration (in contrast to BizTalk orchestration) lets you add business logic directly into the collaboration activity entity by using the Java API. You can add almost any Java-based logic into the WBIS collaboration.

You can use either of these options when migrating the BizTalk orchestration interfaces into WBIS collaboration. You can replace any BizTalk orchestration external interaction with either a service call or a specific Java-based logic.

6.2 WBIS Runtime servers

The WBIS Runtime components can be divided into several categories. This section describes the WBIS server families. The following sections describe other parts of the runtime platform.

To date, the WebSphere Business Integration comprises three business integration runtime environments:

1. **WebSphere BI Message Broker** for a scalable, message brokering enterprise backbone. This runtime also includes other components from the MQ family. Among them are the WebSphere BI Event Broker, and the WebSphere MQ Workflow for human workflow.
2. **WebSphere InterChange Server** (WICS) for business process integration. This runtime is based on the CrossWorld product suite, and is currently branded as WBI Server Express (SE). The WBI SE is targeted for the SMB market, and thus represents the target migration platform for this Redbook.
3. **WebSphere Application Server Enterprise Edition**, whose runtime environment is based on the WebSphere Application Server (WAS) and on the Java 2 Enterprise Edition (J2EE) technology. This runtime currently supports a subset of the BI functionality and components.

IBM is currently unifying these three business integration environments into a converged architecture powered by the robust, scalable, and secure WebSphere Application Server platform. Based on that runtime, and on the Service Oriented Architecture, IBM intends to introduce a unified, open, standards-based platform for business integration.

The new platform will provide state-of-the-art business integration functionality for service invocation; advanced, graphical, development tools; and underlying runtime technologies.

The new development environment will provide a common approach to service choreography, full support for Web Services interaction, and an inventory of common components, with each component delivering a specific service or function.

To support the runtime part of the WBIS environment, IBM intends to provide all its middleware solutions under this unified environment, including its base application server, portal server, and business integration products, including

EAI and B2B capabilities. This new runtime architecture is built on the J2EE standard and heavily leverages Web Services standards to facilitate interoperability with external systems.

The following sections detail the products and components under each of the three runtime families.

6.2.1 MQ-based family

WebSphere Message Queue (MQ) provides assured, once-only delivery of messages between IT systems. IBM has developed a family of products based on the WebSphere MQ product. Messaging transport is one of the foundations of every BI platform. The enabler for every level of application integration is the ability to communicate over a messaging middleware. Extending the messaging transport layer, the MQ-based products provide the fundamental requirements of secure, reliable information exchange, and serve to incorporate services and business process support for application integration and human interaction.

WBI Message Broker

WebSphere Business Integration Message Broker (WBI MB) emerges from the IBM WebSphere MQ. WBI MB provides a powerful message broker solution driven by business rules. Messages are formed, routed, and transformed according to the rules defined by an easy-to-use graphical user interface. Diverse applications can exchange information in dissimilar forms, with brokers handling the processing required for the information to arrive in the right place in the correct format, according to the rules that the user defined. The applications do not need to know anything except their own conventions and requirements. Applications also have much greater flexibility in selecting which messages they want to receive because they can specify a topic filter, or a content-based filter, or both, to control the messages that are made available to them. WBI MB provides a framework that supports basic functions along with user-defined enhancements, to enable rapid construction and modification of business processing rules that are applied to messages in the system.

WebSphere MQ Workflow

WebSphere MQ Workflow works with WebSphere MQ messaging to add a further dimension to business integration by aligning and integrating an organization's staff resources, processes, and capabilities with its business strategies. It enables organizations to accelerate its process flow, optimize costs, eliminate errors, and improve workgroup productivity. WebSphere MQ Workflow is designed to enable the integration of all participants in the business process, including those external to the organization. It ensures that the right information gets to the right person at the right time. WebSphere MQ Workflow can be used

in combination with WBI MB, providing a high level of flexibility for business and message processing.

6.2.2 InterChange Server family

The InterChange Server family is based on a product developed by CrossWorld, which was bought by IBM three years ago. The product was adjusted and improved to align with IBM's strategy and needs.

WBI Server Express

The WebSphere Business Integration Server Express (WBI SE) is the current IBM solution for process integration, workforce management, and application connectivity for the small and medium business environment. The WBI SE is built on top of the former CrossWorld InterChange Server.

Built on open industry standards with components that seamlessly work together, WBI SE enables accelerated integration of business processes that span multiple applications. The WBIS is the core runtime component of the system and is responsible for providing a runtime environment for:

- Collaborations, which are encapsulated pieces of business logic that are responsible for brokering the interaction between business applications.
- Maps, which are used to convert between different representations of business objects.

WebSphere BI Adapter controllers (known also as connector controllers) are used to communicate with the runtime adapters (known also as connector agents).The WBIS interacts with business applications through WBI adapters. It uses collaborations to provide pre-built solutions for business process automation. Application-independent collaborations graphically define end-to-end processes and encapsulate basic integration and business rules for business processes.

6.2.3 WebSphere Application Server family

The IBM WebSphere Application Server (WAS) is a comprehensive Java 2 Enterprise Edition (J2EE) and Web Services technology-based application server. WAS integrates enterprise data and transactions with today's e-business world. Through a rich application-deployment environment, users can build, manage, and deploy dynamic e-business applications, handle high-transaction volumes, and extend back-end business data and applications to the Web. IBM WebSphere Application Server Network Deployment offers advanced Web Services that can operate across disparate application frameworks and

business-to-business (B2B) applications, and provides virtually any-to-any connectivity with transaction management and application adaptivity.

WBI WAS EE 5.0

The WebSphere Application Server Enterprise Edition (WAS EE) is the first server from the WAS family that supports business integration. WAS EE provides business integration capabilities using its J2EE-based runtime, Java Messaging Services (JMS) based middleware, and Web Service Invocation Framework (WSIF) for message processing. Users develop business processes using the add-on Process Choreographer tool and can develop business logic for application integration using the Enterprise Java Beans architecture. The Message Driven Bean can be used to support asynchronous communication with WBI adapters. Session Beans are used to encapsulate business logic behavior and to control the synchronous communication with external applications.

WBI Server Foundation 5.1

The WebSphere Application Server Foundation is IBM's first step toward unifying the various business integration components and tools under a single product. The major addendum to this version of WBIS is the integration of the BPEL process choreographer into the tool suite.

WBIS

WBIS is the IBM flagship business process platform. In this WAS-based version, the WBIS becomes a unified integration product suite that addresses the entire integration space and forms a smooth continuum with the core application server. WBIS is based on a service-oriented architecture that allows integration of composite applications in business processes, and shared tooling between business and technical staff. The WBIS is based on a set of open standards across all platform components:

- BPEL-supported processes
- Eclipse-based tools
- J2EE-CA adapters

The WBIS supports a clustered scalability and single security model.

6.3 Message-oriented middleware

The WBIS platform uses message-oriented middleware (MOM) for communication between the runtime server and its adapters. Users can choose from three communication types: CORBA-based communication, MQ, and JMS.

More than a single communication type is sometimes involved to provide diverse capabilities. Not all servers support all communication types.

1. **CORBA Internet Inter-ORB Protocol (IIOP)** - All handshake communication between the adapter agent and the server is done via IIOP. Delivery of the event/data from the agent to the server is done via IIOP. All services calls that transport data from the server to the agent are done via IIOP. This option is supported only by the WBI SE.
2. **MQ** - All handshake communication between the adapter agent and the server is done via IIOP. The delivery of the event/data from the agent to the server is done via MQ. All services calls that transport data from the server to the agent are done via IIOP. Thus, MQ is only used for the actual delivery of the data event to the server. This option is supported only by the WBI SE.
3. **JMS** - All handshake communication between the adapter agent and the server is done via JMS. The delivery of the event/data from the agent to the server is done via JMS. All calls for services transporting data from the server to the agent are done via JMS. Thus, the ORB is not used at all and everything is done via JMS. All WBI servers support the JMS communication type.

6.4 Relational database

The WBI Server Express uses a relational database (usually DB2) for storing all persistent information. Component states and configuration properties are saved in database tables, as is all server transactional support information. Logging events are also saved in the database.

6.5 WBI adapters

The IBM WebSphere Business Integration offering includes the IBM WBI adapters that allow customers to quickly and easily create integrated processes, which exchange information between ERP, HR, CRM, supply chain, and other systems. IBM provides a number of different adapter types:

Application Adapters - The WBIS Applications Adapters extract data and transaction information from both leading and industry-specific packaged applications like SAP, PeopleSoft, Siebel, and others. They enable access via application programming interfaces where possible.

e-business Adapters - The adapters for e-business are proven solutions for securely connecting over the firewall to customers' desktops, to trading partners'

internal applications, and to online marketplaces and exchanges. Examples are the iSoft Adapter, TPI Adapter, and Web Services Adapter.

Mainframe Adapters - These leverage a best-of-breed technology to access application data in OS/390® systems, as well as provide connectivity approaches for AS/400. Examples of the Mainframe Adapter suite are the ADABAS Adapter, the IMS™ Adapter, and the CICS® Adapter.

Technology Adapters - These adapters provide various ways to access data, technologies, and protocols for enhancing the integration infrastructure. Examples of Technology Adapters are the COM, CORBA, JMS, and EJB™ Adapters.

6.6 WBI Server Express toolset

This section describes the tools and supported resources supplied for WBI SE developers under the WSAD IDE umbrella. The subsequent sections cover tools that do not currently exist in WBI SE, but that are part of the new WBI Server platform.

6.6.1 Eclipse platform and WSAD

The Eclipse platform is an open-source integrated development environment for the creation of tools and for software development. The Eclipse platform provides a useful environment for two levels of software development. First, Eclipse provides the software developer with a full graphical integrated environment. Second, it provides tool developers and software vendors with a development kit and runtime, which enables them to write plug-ins to allow their users to continue with software development in a familiar and unified development environment. IBM has branded its enhanced version of the Eclipse platform as the WebSphere Studio Application Developer (WSAD).

WSAD development platform

IBM delivers the WebSphere Studio Application Developer with the WBIS platform and users install it along with the core infrastructure. WSAD is capable of running all the plug-ins necessary to develop WBIS integration components.

System Manager

The System Manager (Figure 6-1) is an Eclipse perspective that supplies the user with a means to control, navigate, and administer the WBIS platform. From the System Manager perspective, users can create a new project, develop all types of project components, and deploy components to the server.

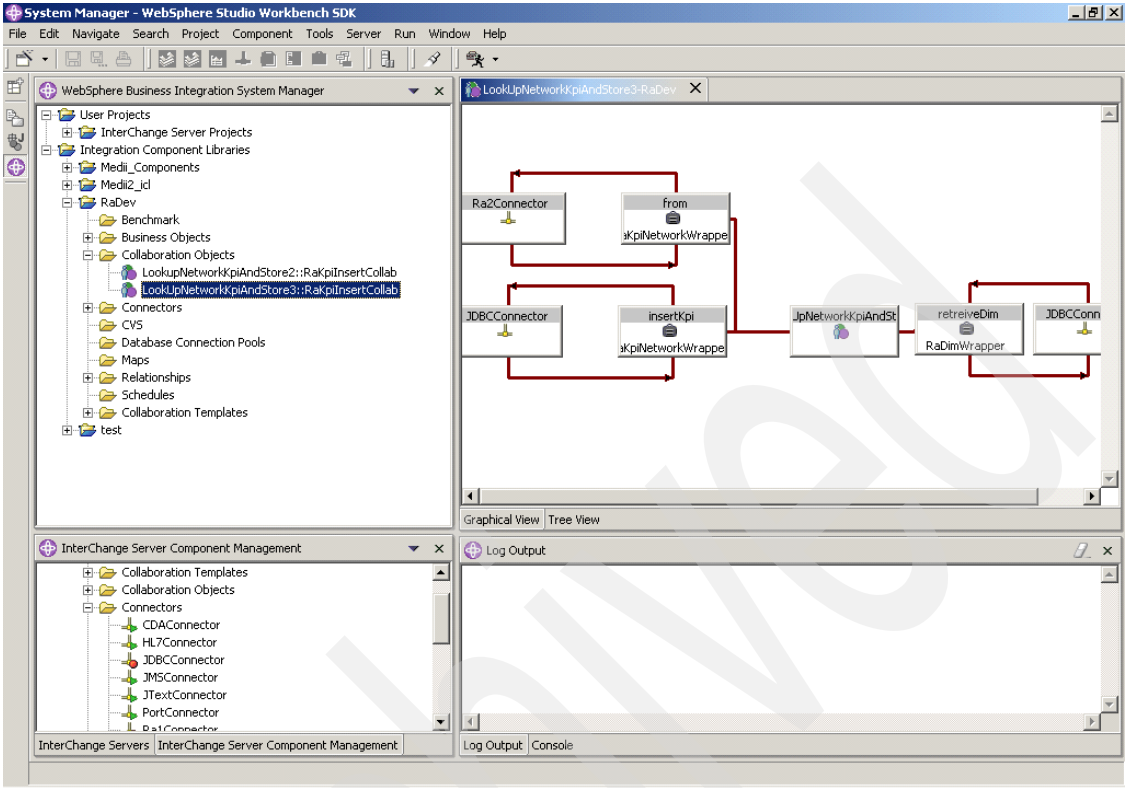

Figure 6-1 System Manager

At runtime, users can start and stop the server, the adapters, and other runtime objects. Users can also open the monitoring and debugging tools from within the System Manager perspective.

The System Manager provides a starting point and an umbrella for a set of development tools, which are described in the following sections.

6.6.2 Business Object Designer

The Business Object Designer tool (Figure 6-2) is used to create and edit business objects. Business objects (BOs) are the data pieces that contain the information exchanged between applications.

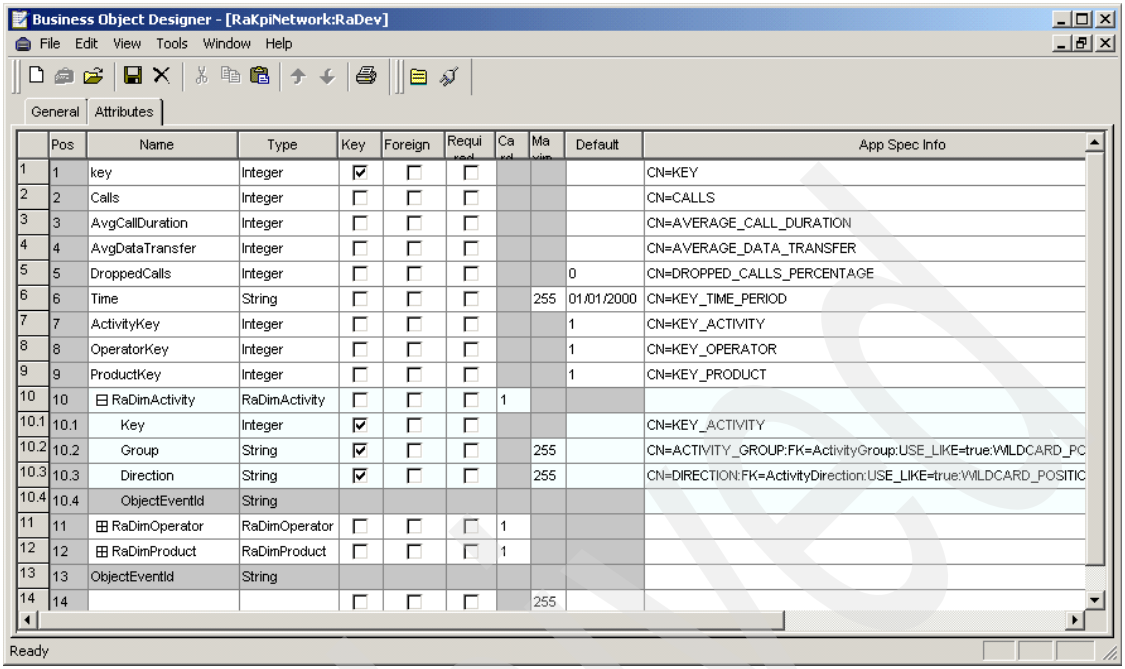

Figure 6-2 Business Object Designer

Two types of BOs exist in the WBIS environment: application-specific business objects (ASBO) and generic business objects (GBO). ASBOs are used for communication between the WBIS platform and a specific application. Additional processing and metadata information exists in the ASBO to allow the WBI adapter to understand the specific application, and to support communication with it. For example, in the JDBC™ adapter, the name of a database table from which a given BO gets its values is specified as application-specific information in that BO. Collaboration, on the other hand, is a generic business process, which does not depend on a specific application. To keep this generality, collaboration uses GBOs and does not refer to application-specific information in the business process.

Business object schema

Business objects are represented using a pre-defined XML schema. Such schema must follow the accompanying metadata XML schema. The metadata XML schema describes the rules that all business object schemas must follow. For example, each business object must have a verb, a version, and at least one key attribute. Figure 6-3 shows the structure of a business object.

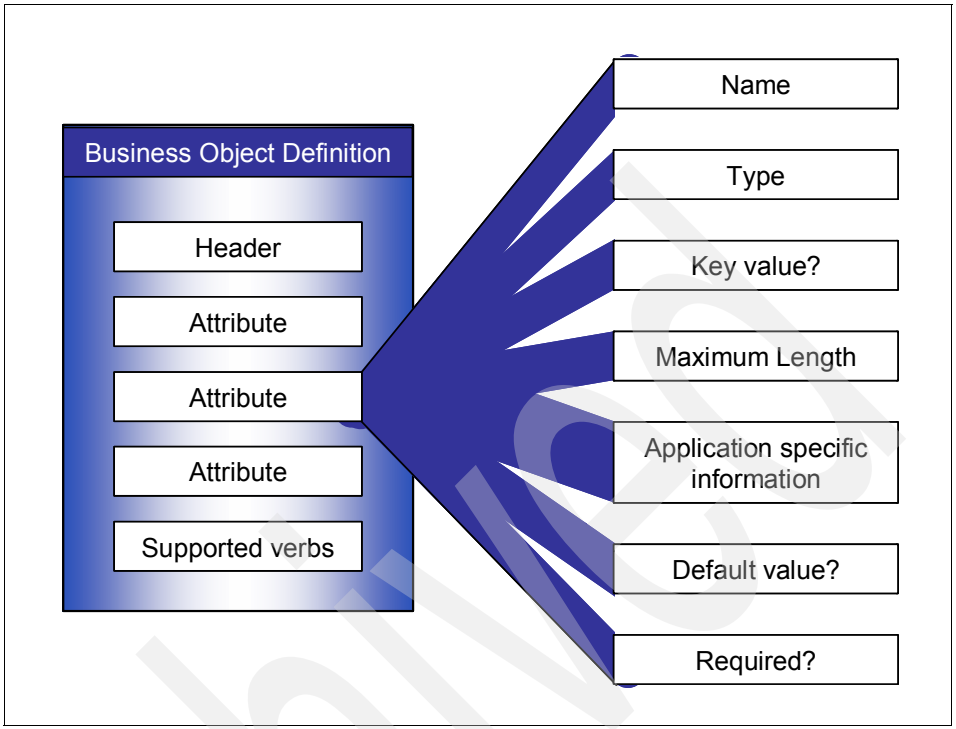

Figure 6-3 Business object structure

This structure is defined in the business object meta-schema. The BO's schema defines the exact verbs for that BO and the specific attributes it contains.

6.6.3 Connector Configurator

The Connector Configurator (Figure 6-4 on page 66) allows users to configure all adapter (also called connector) properties using a graphical tool. In the Connector Configurator, properties are divided into groups, where each group has its own tab. With this tool, users can configure all aspects of a connector's behavior, including:

- Specifying the business objects that the connector supports
- Specifying bounded maps for use between specified business objects
- Setting configuration properties to be used by the connector agent and the connector controller

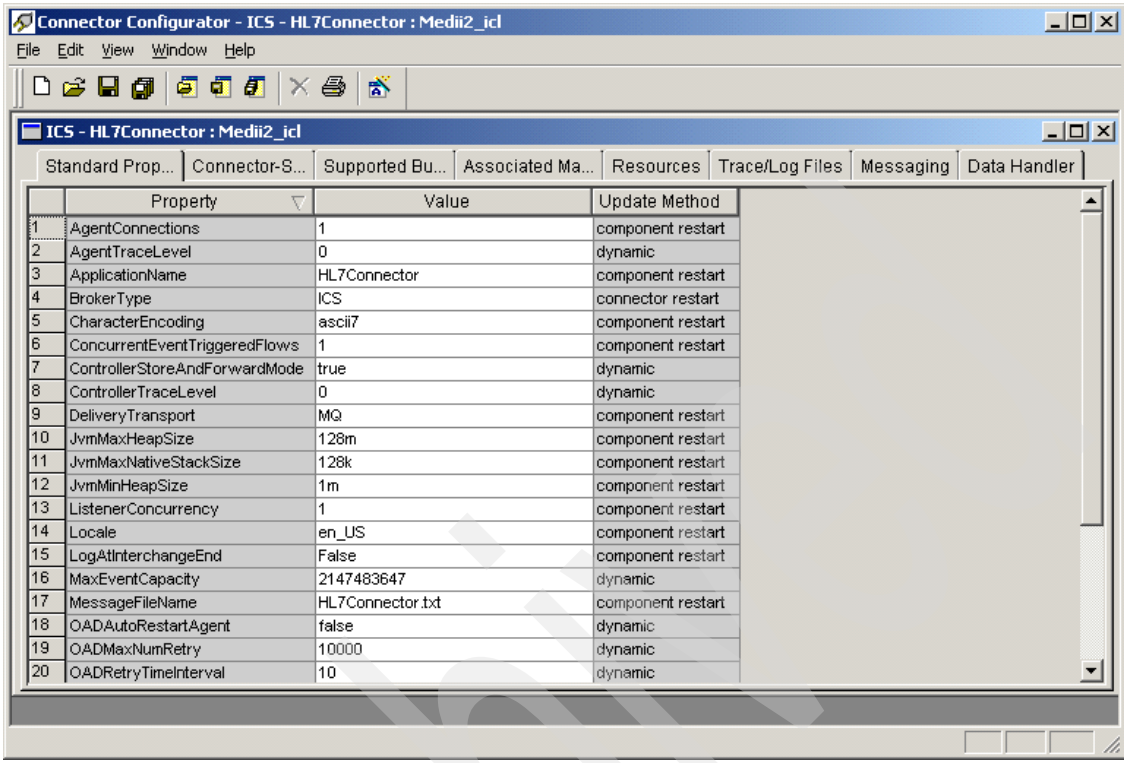

Figure 6-4 Connector Configurator

There are two types of connector agent properties: standard properties and application-specific properties.

Standard properties are available for all connectors, and include the following:

- Type of communication with the integration server (for example, JMS)
- An indication of whether the connector agent logs messages locally or sends them to the WBIS for inclusion in the general log file
- Location of the log file for connector agents that log locally
- Application login and password

Application-specific properties are typically values that a specific connector agent needs to establish a session with the application. The following are examples of application-specific properties for various connectors:

- Name or IP address of the machine running the application
- Identification of application gateway systems

- Name of the application database
- Name of the event inbox

With certain restrictions, connector properties can also be set in local configuration files that reside on the machine where the connector agent is installed. Some connector configuration properties can also be set using the command line.

6.6.4 Map Designer and Relationship Designer

The IBM WBI Server Express system contains a Java mapping API that includes methods for handling common data transformation situations.

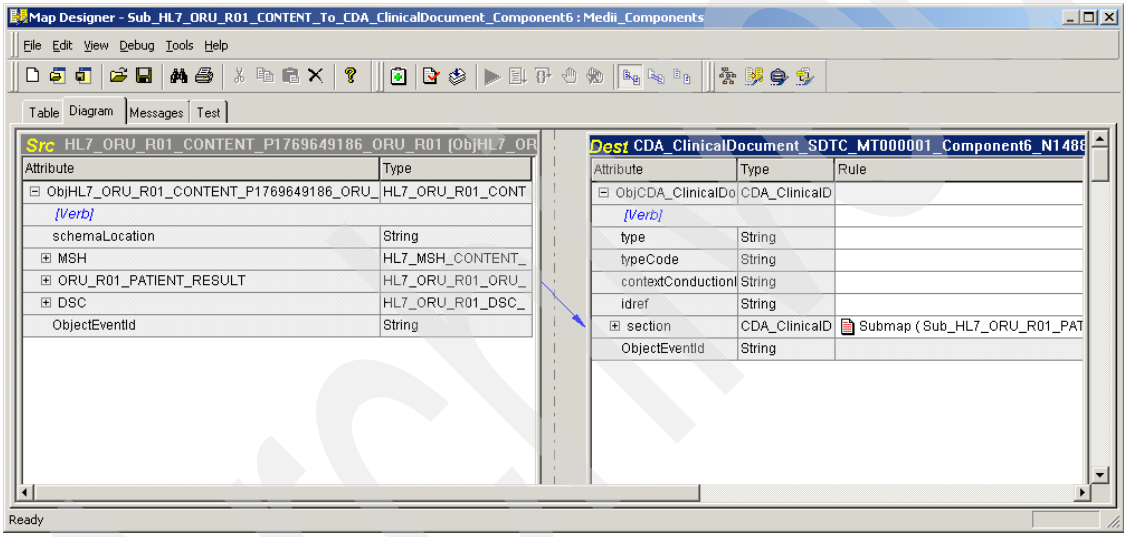

Figure 6-5 Map Designer

Graphical design tools are provided for creating the two principal components of IBM WBI SE mapping, maps and relationship definitions.

- The **Map Designer tool** (Figure 6-5) is used for creating and compiling maps. A map contains the Java code that specifies how to transform attributes from one or more source business objects to one or more destination business objects. Users need a map for every business object to be transferred between different applications. When users modify business objects, they may also need to modify the associated maps. Users typically create a map for each source business object they want to transform.
- The **Relationship Designer tool** is used for creating relationship definitions and the table schemas that are used to store the runtime relationship instance

data. A relationship definition establishes an association between two or more data entities in the WBI SE platform. Relationship definitions within maps are most often used for transforming business object attributes in which the data has a similar purpose but is represented differently in each application. Most maps use one, or a few, relationship definitions.

Both map and relationship definitions reside in the server's repository. Like business object definitions, relationship definitions function as specifications or templates for the instances that are created. Unlike instances of business objects, relationship instances persist, and are stored in special tables for each relationship. Each time the system receives a request to transform a given business object, it executes the associated map, and, depending upon the purpose of the transformation, creates one or more instances of its associated relationship definitions. Relationship instances created during map execution contain the runtime data from the attributes they associate, and this data is stored in the relationship tables.

6.6.5 Collaboration and Process Designer

Collaboration provides the integration logic that controls the interaction between applications and processes. The collaboration is composed of integration logic, state, and transactional control information, and is realized as a Java class. Interaction with collaborations is two-fold:

- Collaborations are event driven. They subscribe to particular (triggering) events within application environments. When these events occur, a collaboration instance is initiated. Multiple collaborations can subscribe to the same event, in which case, multiple collaboration instances are created when such an event occurs.
- Collaborations can interact with other entities within the server in a request/response mode. This may be a synchronous or an asynchronous action, usually dependant upon the target entity that is involved. Collaborations may wait for a response to requests. Such collaborations are referred to as long-lived business processes.

  The partner entity in this interaction can be an application environment, via an adapter or another collaboration.

Collaborations are designed to take advantage of the common business object model, so collaborations generally operate with generic business objects.

Collaboration templates and objects

When users develop a collaboration, they develop a *collaboration template*. A collaboration template contains all of the collaboration's execution logic, but it is not executable. To execute a collaboration, the user must first create a

collaboration object from the template. The collaboration object becomes executable after the user configures it by binding it to connectors or to other collaboration objects, and by specifying other configuration properties.

Collaboration templates are implemented as Java classes; therefore, any extensions made to them must use Java programming language.

The toolset available with WBI SE includes a collaboration design tool called the Process Designer (Figure 6-6), with which users can create and customize collaboration templates. A graphical Java builder called the Activity Editor is available with Process Designer, to reduce the amount of Java code the user is required to develop.

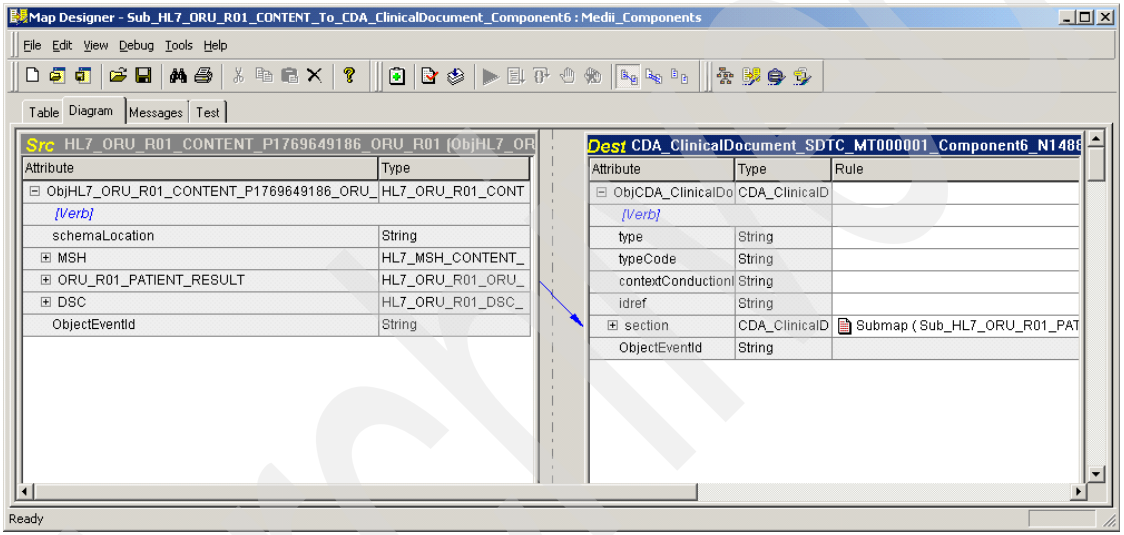

Figure 6-6 Process Designer

6.7 Additional development resources

In this section, we discuss the additional resources used for development.

6.7.1 Object discovery agent

Object discovery agent (ODA) is a methodology and an API for the automatic creation of business objects. The idea behind ODA is to build a tool that automatically introspects the source application's data structure and can create a business object to carry this data to and from the application. A set of APIs is supplied with the WBI SE tools that facilitate creation of object discovery agents.

In addition, two important ready-made ODAs are supplied with the tool set. The JDBC ODA introspects a given database schema and creates a set of business objects that reflect the schema. The XML ODA introspects a given XML schema file and creates a set of business objects that reflect that schema.

6.7.2 Scheduler

The WBI SE tools currently provide only limited scheduling services. Users can configure the runtime component (Collaboration, Map) to start or stop at a given time. This functionality can be overwritten from the system monitor.

6.7.3 Database connection

The database connection provides connectivity to the server repository, which is the runtime data store used by the entire WBIS environment.

6.8 WBI SE tools for administration and control

In this section, we discuss the WBI SE tools used for administration and control.

6.8.1 Integrated Test Environment

The Integrated Test Environment is a WSAD perspective in which users test business integration components they developed. It provides graphical interfaces to emulate connectors, start the required components, and examine business object data.

6.8.2 System monitor

The WBI SE has both Windows-based and Web-based monitors. In addition, a set of APIs is supplied for SNMP-based monitoring:

- **Windows-based monitor** - With this tool, users can monitor and get informational messages for all component status changes in the WBI SE system. Users can also start, stop, pause, and shut down WBI SE components such as collaborations and connectors. The tool has several views related to the various aspects of the system.
- **Web-based monitor** - Using the Web-based system monitor, users can monitor the WBI SE system from the Web. Unlike the Windows-based version of system monitor, it allows the users to configure how they view the data and also allows them to view historical data in addition to current data. Like the Windows-based system monitor, it allows users to start, stop, and pause components.

- **SNMP-based monitor** - The SNMP agent that can be installed with the WBI SE allows a remote SNMP manager to monitor and perform limited management of the servers, collaborations, and connectors, based on a Management Information Base (MIB) protocol.

6.8.3 Flow Manager

The Flow Manager is a tool supplied with the WBI SE tool set to locate, view, and handle failed events. The Flow Manager allows users to construct a query to locate and display unresolved flows. After the tool displays the unresolved flows, the user can select any flow in the display and choose to submit it, discard it, or perform other actions on it. The Flow Manager's main task is to supply the user with additional monitoring and debugging tools. The Flow Manager sometimes adds additional information and logging to error events and allows the user to more easily identify and solve the problem.

6.8.4 Log Viewer

The Log Viewer is a simple file viewer supplied with the WBI SE tool set. This tool enables users to view structured log messages generated by the WBIS system, and hence more easily identify error and warning messages.

6.9 Tools not included in WBI SE

The following tools are not part of the WBI Server Express and are therefore not expected to play a role in the migration process. They are added here for completeness and to pave the way for the next migration steps that will be required from BizTalk to WBIS.

6.9.1 WBIS Modeler (Version 4.2.4)

The WBIS Modeler provides business professionals with easy-to-use tools to design, test, and communicate complex business processes. Users can simulate the operational efficiency of business processes and analyze potential business results, all before committing resources to development and deployment. WBIS Modeler comprises two components:

- **WBI Workbench** to help design and implement complex process workflows. This component provides advanced analysis, reporting, integration, and deployment capabilities.
- **WBI Workbench Server** to provide repository management capabilities. This component helps manage different versions of the business process models

and selectively shares them among users. It also allows users to publish models, policies, procedures, and business rules online.

The WBI Workbench offers a full range of business-process modeling capabilities, from creation of process flow diagrams to validation of process outcomes. This component enables users to export business process definitions to WebSphere MQ Workflow (formerly MQ Series Workflow) or WBIS for implementation, or to Rational® Rose® or Rational XDE™ for further refining by IT architects and developers.

The WBI Workbench is built on three components: business modeler, Unified Modeling Language (UML) modeler, and Xform designer.

- **Business modeler** - The WBIS business modeler enables users to simulate the operational efficiency of the processes and analyzes potential business results. Users can include collaboration tasks from the WBIS and MQ tasks from the WBIS Message Broker in the process models. The WBIS business modeler allows users to edit objects from within the WBI Toolset - System Manager and enables them to capture business processes and guidelines in a single place for easy reference and company-wide consistency. The modeler supplies a simplified graphical interface to facilitate modeling diagrams of current and new business processes. The different business scenarios can be simulated before they are implemented, allowing users to make the most of the designed business processes. Users can identify the optimum solution for the current business environment by projecting outcomes before implementation. Processes can be quickly adjusted when business factors change in the future. The modeler produces process metrics for analysis in multiple chart types, and includes capabilities for export to spreadsheet applications, where additional data management options can be deployed.
- The **UML modeler** component bridges the gap between software developers to business analysts. The UML modeler conveniently exports UML output to rapid development tools, such as Rational Rose or Rational XDE software, allowing extended data use and flexibility. Object models generated by the Rational Rose visual modeling tool can be used with the UML modeler.
- The **Xform designer** component places the capacity of customized workplace design in the hands of the everyday business user. WBI Workbench users can quickly design GUIs to meet specific enterprise needs, and then provide screen mock-ups of new interfaces.

By linking WBI Workbench and WBIS, users can create a single repository for all business process models, business objects, collaborations, and maps within the enterprise. In addition, WBI Workbench allows users to launch the WBI Process Designer and use it to edit collaborations.

6.9.2 WebSphere Process Choreographer

The WebSphere Process Choreographer provides a rich standards-based (BPEL4WS) process choreography engine that combines human and Enterprise Application Integration (EAI) workflow technologies. The WebSphere Process Choreographer is a component of a broader industry-leading application platform that forms the foundation of the new converged WBIS.

7

BizTalk to WebSphere Business Integration architecture mapping

In this chapter, we divide the business integration environment into several architectural parts. For each part we refer to a functional task or an encapsulated component and describe both the BizTalk and WBIS approaches to that part. We then describe the various migration techniques we can take for migrating this component or task from the BizTalk environment into WBIS. We conclude each part with a summarization table of the architecture mapping and the migration options.

7.1 Messages

Messages, in BizTalk terminology, or *business objects* as they are called in the WBIS environment, are the data structures used to carry application data into the BI server and back. Messages are the fundamental data structures in the business integration environment, and are referred to by all other parts and activities in the environment.

7.1.1 BizTalk approach

BizTalk messages are mostly, but not exclusively, in XML format. Flat file formats (delimited or positional) are also supported, in which case BizTalk specifies an intermediate XML format. Pre- and post-processors transform flat files to and from this intermediate XML format so that the bulk of the transformation is done in XSLT. The intermediate XML format contains annotations with application-specific information to facilitate these transformations. In some scenarios, the original message (which could even be a binary file) is passed through to a destination, without transforming it and without any workflow handling.

BizTalk orchestrations use XML messages internally, regardless of the format in which incoming messages are received, or in which outgoing messages are sent. These internal XML messages have a pre-defined schema, which is described using the standard XML schema definition language. BizTalk Server 2002 supports XDR, a reduced and non-standard version of XSD.

There are several ways to create BizTalk messages:

- BizTalk Editor can be used to create new, self-contained schemas, or schemas that use other schemas.
- Existing schemas can be imported from the file system.
- Schemas can be generated based on given XDR, DTD, or well-formed XML files.

BizTalk message schemas can be classified into three types, all of which conform to the XDR in BizTalk 2002:

- **XML schema** supports XML message types.
- **Flat file schema** supports delimited or positional flat file formats.
- **Envelope schema** supports one or more XML messages wrapped together in an envelope.

In BizTalk 2002, document specifications are created by BizTalk Editor and stored as XML files in the WebDAV repository. WebDAV stands for Web-enabled

Distributed Authoring and Versioning. This repository provides a collaborative environment for users to edit/manage files on Web servers. Technically, WebDAV is an extension to the http protocol.

Document definitions are objects created by the BizTalk Messaging Configuration object model (either through BizTalk Messaging Manager or the API) and stored in the BizTalk Messaging Management database, an SQL Server database used by BizTalk Server. Document definitions are XML specifications that have been converted into an object form.

Although BizTalk 2002 does not support XSD, there are tools that transform XDR schemas into XSD schemas, such as the XML Schema definition tool (xsd.exe) in .NET. Furthermore, BizTalk Editor itself claims it does support exporting XSD. When migrating BizTalk 2002 messages, a recommended first step would be to normalize BizTalk messages by transforming their specifications into XSD.

7.1.2 WBIS approach

The WBIS platform uses the business object (BO) concept and structure to send information back and forth to applications. The BO is an XML-based structure used mainly as a data container. The WBIS platform also uses BOs as complex data type inputs and outputs for maps and collaborations. The BO follows a pre-defined XML Schema Definition (XSD) and is based on several architectural guidelines.

Each BO has a header segment and a data segment.

- **Header segment**: The most important part of the BO header is the verb. The verb describes the action to be carried out on the BO's data. WBIS has a set of standard verbs: Create, Delete, Update, and Retrieve. Users can also add new verbs to the BO.
- **Data segment**: The BO data segment is composed of a set of attributes. Each attribute can be either a simple type, such as *String* or *Integer*, or it can be a more complex type. Each complex type is a BO in itself. Each attribute has an additional list of properties. The main properties are: type, cardinality, default value, and whether this attribute is a primary or foreign key. Each attribute also has a field for application-specific information (ASI). In this field, users specify processing instructions for the application that processes this BO. If a single BO is targeted for handling by different applications, it may have different ASI.

The main task of the WBI adapter is to translate a given data format into a business object. The adapter serves, in that sense, as a gate to the WBIS environment. Once it passes the adapter, data can be accessed and manipulated using the business object API.

The Data Handler is a runtime component used mainly by the WBI adapter to parse outgoing data from business objects, and to construct business objects from incoming data. The Data Handler uses the information from the BO's ASI field, structure, and directives to create the required application format, and to construct a specific BO instance from data coming from a given application. The Data Handler is implemented as a Java jar file and has a set of APIs. The API is usually called from within the WBI adapter, but can also be called from within a collaboration inside the WBIS.

WBIS users create BOs using the WBIS Business Object Designer tool. This tool enables users to graphically create BOs, so they supply only those properties and attributes that are legal. With this tool, users don't have to be familiar with BO syntactic and semantic constraints. After creating a business object, the user saves it to the file system, where the business object is saved as XML schema. Each BO is saved to a separate file. Because a BO can be nested inside other BOs, the tool performs a check to ensure that all references are valid. When users open an existing, already saved BO, the tool reads the BO's schema and presents its content to the user.

The Business Object Designer uses an internal text format (*.in) to describe business objects. This simple text format allows the tool to import BOs from files previously created by the ODA.

Migration options

For migration from BizTalk to WBIS, a business object must be created for any message schema that is used in the BizTalk environment. We believe that a set of BOs can be created to describe every possible XML schema. We know, on the other hand, that creating BOs for complex XML schema can be very tedious work, especially if done manually.

Object Discovery Agent: The WBI tool set contains a useful tool designed exactly to address this need. The Object Discovery Agent (ODA) can be used to introspect a given XML schema and to create the BOs needed to describe the schema.

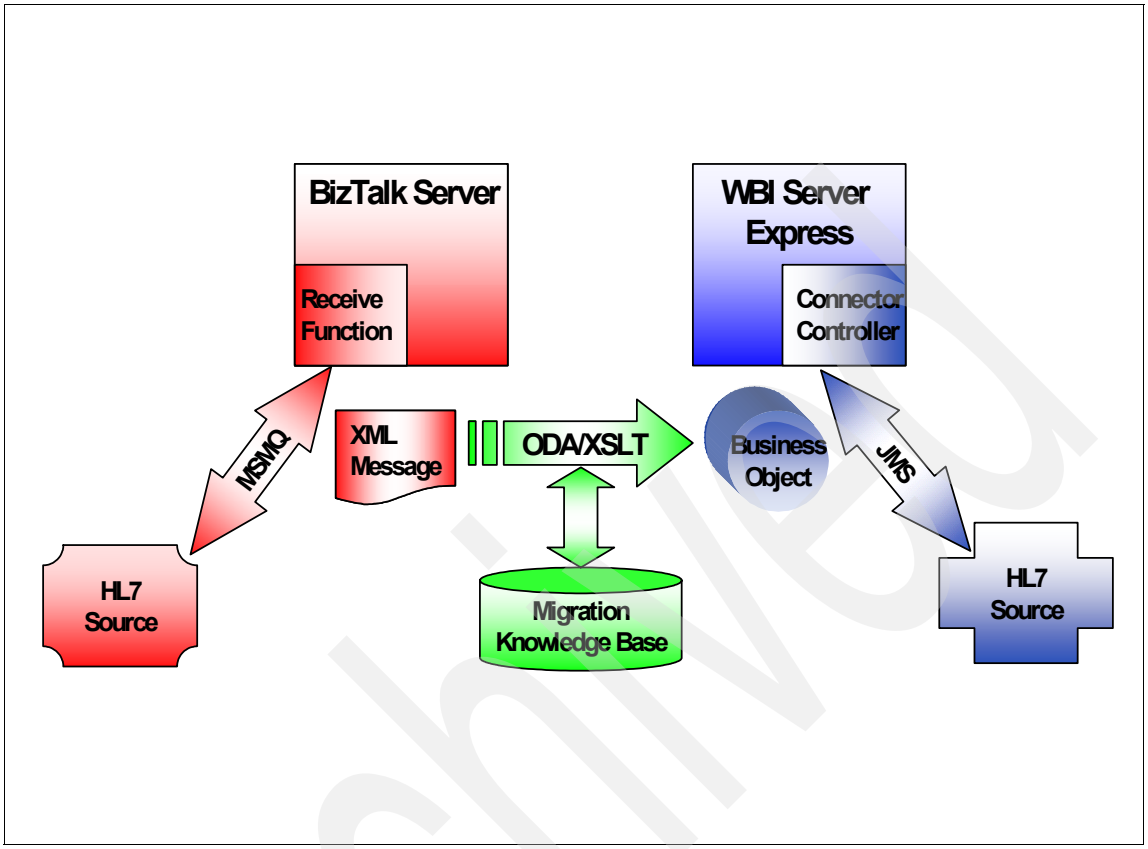

Figure 7-1 BizTalk message to WBIS business object transformation

Custom transformation tools: An alternative to the ODA approach is to use a transformation tool to transform the BizTalk message schemas into WBIS business objects (Figure 7-1). Using a transformation engine, users can transform one XML schema into another. In our case, transformation is needed from the BizTalk message schema into the WBIS business object schema. Several commercial tools are available on the market today for XML transformation. Using such tools, the user can better control the way information is mapped from one environment to the other. In addition, BizTalk's schema editor adds many annotations to the output message schema it creates. These annotations are well-structured because they are generated by the tool and not by the user. Using a custom transformation for BO creation, users can take advantage of that information and improve the migration capabilities. The disadvantage of this approach is that it requires manual work from the user.

Manual transformation: At the other extreme, users can work with the Business Object Designer tool to manually migrate BizTalk messages into WBIS. Although this doesn't sound attractive, it may be the most appropriate approach for cases where only a few message types exist in the system. Working with the Business Object Designer is straightforward and simple, and creating BOs with the tool is an easy task, especially when transforming from an existing structured format.

Message encapsulation: It may be useful in some cases to build a shell BO and to insert the whole BizTalk Message into one of its attributes (for example as a String or a Binary Large Object - for example, a BLOB). In this way, the WBIS is able to receive the BizTalk message without migrating it. The main disadvantage of this migration option is that no knowledge regarding message structure and values is gained. The BizTalk message remains opaque to the WBIS. The message must be parsed or transformed at some other point inside the WBIS if such knowledge is important for application integration or for the creation of business processes.

Comparison table

Table 7-1 summarizes the architecture mapping and the migration options for the messages part of the business integration environment.

Table 7-1 Comparison table

Feature	BizTalk	WBIS
Underlying technology	XML	XML and Java
Development tool	BizTalk Editor	Business Object Designer
Major migration technology	Uses ODA to translate BizTalk messages into BOs. An extension to the ODA is needed to allow saving the XML field to BO field mapping. This mapping can then be used to assist in transforming the BizTalk transformation artifact into a WBIS mapping artifact.	
Other migration options	Uses XML transformation engines, other than the ODA (for example, ContentMaster from ItemField). Manual transformation. Message encapsulation.	
Major migration gaps	No easy way to generate the WBIS application-specific information.	

7.2 Adapters and application connectivity

The main task of the business integration platform is to interact with external applications. The business integration architectural concept is to let the adapters

deal with the external applications, and supply the users with a tool set to design transformations and business process flows independently of application-specific interfaces.

7.2.1 BizTalk approach

The BizTalk platform interacts with external applications and other software components using adapters.

In BizTalk 2002, *receive* adapters communicate with BizTalk Server, either via a receive function or via the IInterchange COM object. IInterchange provides two ways of submitting documents to BizTalk Server: Submit and SubmitSync. The Submit method uses MSMQ to store messages, and it is asynchronous and reliable. SubmitSync bypasses MSMQ; it is synchronous and performs better, but it is not reliable (for example, if the BizTalk Server is not available, the message will be lost).

Send adapters do not have a clear-cut definition in BizTalk 2002. BizTalk communicates with applications in one of three ways: via orchestrations that post messages to a message queue, or using a messaging port that passes messages directly to their destination (not via an orchestration), or by having orchestrations or messaging ports invoke an AIC.

BizTalk invokes AIC through a well-defined interface. AIC has knowledge of the application-specific protocol. So, by placing an AIC between BizTalk and the application, BizTalk can communicate with AIC without having any knowledge of the application-specific protocol.

7.2.2 WBIS approach

The WBIS platform interacts with external software components using WBI adapters (also called *connectors* in WBIS terminology). An adapter generally has two main interfaces: the application interface and the server interface. The adapter communicates with the server using one of three communication options: CORBA, MQ, or JMS. The adapter communicates with an application using the application's protocol. Hence, communicating with external applications requires using an adapter suitable for that application.

The adapter serves as a gate into the WBIS environment. Passing the adapter, data is encapsulated in a known, common way (the business object structure) and is communicated to the server and back using a known, unified protocol. The adapter's, or connector's, main task is to adapt the application protocol and data format for the WBIS platform. The adapter uses a Data Handler and API to execute the data format adaptation.

7.2.3 Migration options

Migrating the BizTalk connectivity components (for example, receive functions, channels, ports, adapters, AICs, and so on) into the WBIS environment is not a straightforward task. BizTalk and WBIS use different architectural approaches for application connectivity. While WBIS uses a very consistent and uniform method for application connectivity, namely the adapter approach, BizTalk uses a bundle of technologies, components, and interfaces to accomplish this same task.

In general, all BizTalk connectivity components should be migrated to a counterpart adapter. In some cases, a ready-made adapter can be used. In other cases, the adapter may need to be developed to target all tasks of existing BizTalk connectivity parts. Hence, the main migration techniques for this section of the architecture will be component substitution and configuration, as well as the adapter development technique.

Component substitution is more practical in cases where a BizTalk adapter is being used to communicate with an application (for example, SAP) and the WBIS platform has a counterpart adapter (WBI SAP adapter in that case). It is then possible to substitute the BizTalk SAP adapter with the WBI SAP adapter, and configure it with the relevant properties to the SMB runtime and administration environment. Best practices may be used in that case to solve migration and configuration issues.

Adapter development can be used in cases where the BizTalk application connectivity is spread across several components, and includes data-related tasks, such as transformation, validation, and so on. A counterpart adapter may be developed (or reused from similar cases) to implement the connectivity part and a Data Handler can be developed (or reused) for the data manipulation tasks.

Migration option table

Table 7-2 summarizes the architecture mapping and the migration options for the adapters and other data connectivity areas of the architecture.

Table 7-2 Migration option summary

Feature	BizTalk	WBIS
Architectural concept	Set of interfaces and other connectivity components such as: receive function, channel, AIC, port, and adapter.	A unified adapter approach for connectivity implementation, with Data Handler for data manipulation.

Feature	BizTalk	WBIS
Development tools	Many ready-made adapters for application connectivity. Ready-made receive function for common cases. A set of interfaces for custom development.	Many ready-made adapters and Data Handlers for solving common cases and an Adapter Framework API for adapter and Data Handler development.
Migration issues	No easy migration path. Manual work is expected. Best practice recommendation should advise on mapping existing BizTalk connectivity components into corresponding WBI adapters and their supporting components. A set of counterpart WBI adapters and Data Handlers can be used to substitute BizTalk components. Additional adapters may need to be developed to address specific BizTalk solution needs.	

7.3 Data transformation

Data transformation is the task of taking data in a given format or structure and transforming it into another format or structure. Data transformation is the basic building block of application integration. Data transformation is almost always necessary to allow different applications to communicate. Both BizTalk and WBIS supply tools for data transformation.

7.3.1 BizTalk approach

BizTalk 2002 provides a graphical tool (BizTalk Mapper) for users to visually define the transformation map. Based on this map, a corresponding Extensible Stylesheet Language Transformation (XSLT) is generated by BizTalk. This XSLT is used by Microsoft XML Parser MSXML (a DOM-based XML parser, which supports SAX, XSLT and XPath W3C specifications) at runtime to transform the schema on the server. A rich set of functoids is provided, and additional custom transformations can be written by developers using native code (VB, C++, and so on). Custom transformations must either implement a known BizTalk interface, or extend a known BizTalk class (IFunctoid and CannedFunctoid in BizTalk 2002).

File system structure in BizTalk 2002: Maps are XML files (stored with an .xml extension) that hold the following information:

- Source and destination schemas
- Links and their properties
- Functoids and their properties
- Compiled XSLT

7.3.2 WBIS approach

The WBIS platform uses the maps and the Map Designer tool to transfer business objects from one format to the other. Map Designer is a graphical tool that allows users to graphically map attributes from one BO to the other. Map Designer uses a set of ready-made transformations: *Set Value*, *Move*, *Join*, and *Split*. Users can call a submap from within a map to transform child business objects to their destination. Specific utilities are provided to transform dates and strings.

Users can also generate custom transformations using Java code snippets. These code snippets can be developed with a graphical tool called the Activity Editor, or programmatically, using Java.

The WBIS platform uses the Relationship Designer tool to associate BOs that cannot be directly mapped due to the fact that each application maintains the data in its own format. The tool uses Identity Relationship to perform dynamic cross-referencing between data which evolves frequently, and Lookup Relationship to find the equivalent value to the source attribute.

The BizTalk platform does not have an equivalent tool so no migration efforts are expected in this area.

File system structure: The WBI SE platform uses a set of file types to support the Map Designer and the Relationship Designer tool:

- The Relationship Designer `.cwr` file type uses XML format to describe the input for the Relationship Designer tool.
- The Map Designer `.cwm` file type uses XML format to describe the input for the Map Designer tool, and then uses this file to graphically present the developed mapping. The Map Designer generates a `.cwm` file for each newly created map and modifies this file every time users update the map.
- At deployment time, the Map Designer creates a `.java` file that describes the exact processing needed for each map.
- Map Designer then compiles the `.java` file into a `.class` file, which can be executed by the WBIS.

7.3.3 Migration options and issues

Transforming XML files from one format to the other is a complex task. Today's market has more than a few commercial products for generating transformations. Some of these products use transformation standards, such as XSLT, while others use proprietary methods. Transformation, in many cases, requires a lot of flexibility and options, which necessitates that users write custom code using a full-fledged programming language. Transformation migration from one

environment to another is expected to be even more complex. An exception to this is when the two environments use standards-based transformers, such as XSLT. WBIS currently uses Java as its transforming language and not XSLT.

Several options exist for migrating BizTalk transformation artifacts into WBIS maps:

- **Migrate BizTalk transformations to WBIS .cwm file format**

  Building on the observation that both BizTalk transformation output format and WBIS map input format are XML-based formats, the XML transformation technique can be used to translate from one format to the other. Special attention must be given to the mapping inputs and outputs when transforming the XML format. While BizTalk uses the source message XML field as its input and the target message XML field as it output, WBIS uses business objects. Hence, the message's field translation must be preserved when transforming a BizTalk map into a WBIS mapping (Figure 7-2). This translation may be obtained directly from the ODA tool, if the tool was used for message transformation, or from another message structure analysis tool.

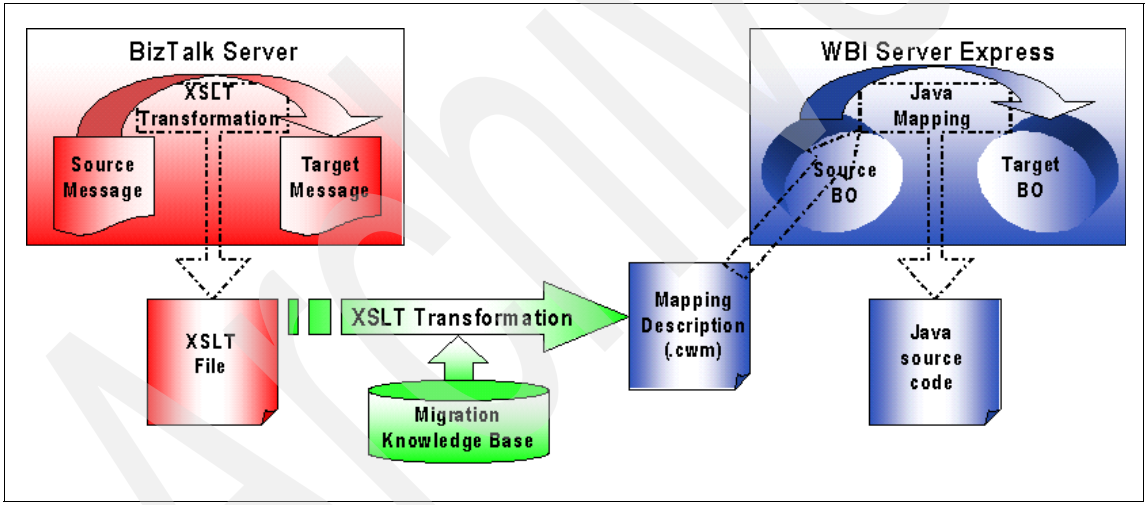

Figure 7-2 Data transformation migration

- **Functoid migration**

  A functoid is a short piece of code with a well-defined interface. Functoids are used by the BizTalk mapping tool for specific data transformation tasks. Functoid code is usually written in Visual Basic script. The BizTalk Mapping tool supplies a set of ready-made functoids. Developers can develop and add their own functoids as needed.

Using the interface adjustment technique, a counterpart interface may be supplied to WBIS to allow it to execute the functoid code without any changes. In addition, a set of counterpart WBIS mapping components can be developed to match existing BizTalk functoids. Once we have the full set of mapping components, the component substitution technique can be used to substitute BizTalk functoids with their WBIS counterparts.

- **Run BizTalk XSLT-based artifact using XSLT engine in the WBIS runtime environment**

  The BizTalk transformation output provides an XSLT that reflects the exact transformation from the source BizTalk schema to the target BizTalk schema. Using the black-box technique, an interface to an XSLT engine can be developed to support running BizTalk XSLT artifacts from within WBIS, thereby producing the desired transformation. The main disadvantage of this option is that no corresponding WBIS transformation is generated. If developers will want to view or edit the transformation, this could not be achieved using the WBIS Map Designer tool.

- **Migrate custom transformation**

  The BizTalk development environment allow users to add their own custom transformation using a known interface. The problem introduced is related to the input and output for that interface. While BizTalk developers may send the entire BizTalk message into the custom transformation component, such a message does not exist in the WBIS environment. This is because BizTalk messages are transformed into WBIS business objects while entering into the system. Two possible solutions are:

  - Add a field into the BO structure that will hold the entire BizTalk message as a String or a BLOB.
  - Supply a Data Handler that is capable of parsing the BO back into the original BizTalk message and executing it prior to sending the message into the custom transformation code.

- **Migrate BizTalk transformations to WBIS .java file format**

  It is possible, at least theoretically, to generate a .java file directly from the BizTalk transformation file format. Moreover, we could think of an option to generate the .java code directly from within the BizTalk development environment. The main disadvantage of this option, as described in previous sections, is that it requires two development environments. Users continue to develop in the BizTalk platform and have no way to see their maps using the Map Designer. Hence, this technique is beyond the scope of this Redbook.

Migration option table

Table 7-3 summarizes the architecture mapping and the migration options for the data transformation part of the WBIS platform.

Table 7-3 Migration option summary

Feature	BizTalk	WBIS
Underlying technology	XML and XSLT	XML and Java
Development tool	BizTalk Mapper	Map Designer; Relationship Designer; Activity Editor
Additional development components	Functoids; Custom transformation	Components for Activity Editor, Custom map
Major migration technique	Uses XML transformation to transform BizTalk transformation artifacts into WBIS map artifacts. Transformation must take into account the relevant transformation of BizTalk message to WBIS BO.	
Additional migration techniques	Uses interface adjustment for functoid migration. An interface for executing XSLT transformations can be added to WBIS.	
Major migration gap	Custom transformation can be developed and plugged-in into various components of the WBIS platform. There is no easy way to migrate this custom code.	

7.4 Business processes choreography

Business processes represent the heart of a business integration platform. With process choreography tools, users of the BI platform can create business process flows. Process choreography deals mainly with two aspects of flows: human interaction and application interaction. Roles are an additional dimension dealt with by the business process environment.

Recent developments in business process modeling try to decouple the tools for different organizational roles. Different sets of business process development tools are supplied for the analyst, the developer, and the administrator. While separating the user interface for these roles, BI platforms must keep the model and data for the process orchestration shared for all users, and must support collaborative interaction among users, and between users and data.

7.4.1 BizTalk approach

BizTalk 2002 provides a graphical tool (BizTalk Orchestration Designer) for users to visually define their business processes. Once the orchestration (or schedule) is created, it has to be compiled to a proprietary XML grammar, known as XLANG. This grammar describes the business process and the binding of that

process to the application services. XLANG formally specifies business processes as state-full, long-running interactions. Business processes involve more than one participant, and the description of the process must show the visible behavior of each participant and the messages exchanged. XLANG provides the following features:

- Sequential and parallel control flow constructs
- Long running transactions with compensation
- Custom correlation messages
- Handling of internal and external exceptions
- Dynamic service referral
- Multi-role contracts

File system structure in BizTalk 2002: Orchestrations are XLANG files (XML files stored with a .skx extension). BizTalk 2002 has no BPEL support.

7.4.2 WBIS approach

The WBIS platform uses collaboration terminology to describe business process flows. Users create collaborations using the Process Designer tool. The collaboration then runs under the WBIS runtime. Collaborations represent the business logic required for application integration. Business objects are the main input and output of collaborations.

A collaboration is a two-part entity. In the first stage of collaboration development, users create the collaboration template. This template defines the input and output business objects and the business scenario to be executed. In the second stage, users create a collaboration object, based on the template, and bind it to specific connectors, or to other collaboration objects.

File system structure: WBI SE uses a set of file types to save a collaboration:

- WBI SE uses a .cwt XML file to save the collaboration template structure and properties. The .cwt file contains both the template's runtime behavior and the presentation information required by the Process Designer GUI. The Process Designer is capable of opening a .cwt file and graphically presents its content to the user.
- The user is able to modify and save the collaboration template. Saving the collaboration template updates the .cwt file.
- Users can also compile the collaboration template, at which time a Java source code file (.java) is created.
- The Process Designer then compiles this file into Java byte code with a .class extension.

- WBI SE uses the .cwc file extension to describe a collaboration object. The System Manager uses the .cwc file to preset the collaboration object and to create and configure the object's bindings and properties.

Import BPEL and UML files: The Process Designer can export and import Business Process Execution Language (BPEL) and Unified Modeling Language (UML) files. Users can import files for use as the basis for a new collaboration template, or export a collaboration template to BPEL or UML format for use with other applications.

The Process Designer uses three file types to describe a collaboration template:

- The .bpel file contains the main template information.
- The .wsdl file contains information about the external interfaces.
- The .bpelGUI.xml file contains information about the graphical representation of activity diagrams. It is used in situations where BPEL files are imported back into the Process Designer.

7.4.3 Migration options

Some migration options and techniques can be developed to transform BizTalk orchestrations into WBIS collaborations:

- **Export from BizTalk and transform to WBIS .cwt file**

  We can create an XML transformer that transforms the BizTalk XLANG format into WBIS .cwt format. We can then open this new .cwt file and save it as a new collaboration template. Special attention must be given to artifact input and output while transforming BizTalk orchestrations into WBIS collaborations. The same mapping of BizTalk Messages to WBIS business objects must be preserved in order to maintain the correct collaboration functionality.

- **Transforming orchestration components and code snippets**

  The BizTalk Orchestration Designer uses a set of Visio components and blocks for orchestration creation. Building on the well-defined interface of such components, the counterpart WBIS collaboration activities can be generated and used. A set of activity components can be added to the Activity Editor to support the existing BizTalk components' functionality.

- **Export from BizTalk and import to WBIS using the Process Designer**

  BizTalk 2002 uses a Microsoft proprietary predecessor of BPEL called XLANG. Hence, relying on BPEL export-import technique may be better left for future migration plans.

Migration option table

Figure 7-4 summarizes the architecture mapping and the migration options for the process choreography part of the business integration platform.

Table 7-4 Migration option summary

Feature	BizTalk	WBIS
Underlying technology	XML and XLANG	XML and Java; limited support of BPEL
Development tool	Orchestration Designer	Process Designer; Collaboration Object Wizard; Activity Editor
Major migration technique	XML transformation of BizTalk orchestration file (*.skx) into the WBIS Collaboration Template file (*.cwt) and WBIS collaboration object file (*cwc).	
Additional migration techniques	Supply a set of activities to match BizTalk Orchestration Designer component capabilities.	

7.5 System configuration

System configuration is the process in which developers configure the developed artifacts and other system components to adjust their operation to the platform runtime requirements. A set of utilities and tools is supplied by the business integration platform to support the configuration task.

7.5.1 BizTalk approach

BizTalk is a highly configurable platform, where configuration information is stored in the configuration database. The BizTalk Server 2002 messaging service, which manages the exchange of documents between applications, can be configured via the BizTalk Messaging Manager UI or programmatically via the BizTalk Messaging Configuration object model. The configuration model contains the following main objects:

- **IBizTalkChannel** -configures a channel for processing documents. This channel contains information about the source entity and its binding to an IBizTalkPort object.
- **IBizTalkDocument** identifies and describes the specification of a document.
- **IBizTalkOrganization** identifies the source or destination point for the exchange. This can designate a source application in a channel or a destination application in a port.

- **IBizTalkPort** identifies the source and destination application/organization, the transport type, and the associated envelope for transmission (if any).
- **IBizTalkPortGroup** configures a port group that is used to distribute the same document to many organizations.

Configuring BizTalk Messaging service means:

- Creating document definitions, envelopes, and organizations
- Creating messaging ports and channels to manage the exchange of documents
- Creating distribution lists

7.5.2 WBIS approach

System configuration is fundamental to the WBIS platform, and all parts of the platform are highly configurable. WBIS uses a set of utilities and methods for system configuration, where the most configuration-intensive part in WBIS is the WBIS connector.

WBIS supplies the Connector Configurator as a dedicated tool for configuring connectors. The connector has a generic part, common to all WBIS connectors, and a specific part in which users can specify and set specific configuration properties for the connector. The WBIS connector uses the meta-object design pattern to support additional configuration flexibility. Using meta-objects, implemented as a regular business object, users can further configure the connector and the Data Handlers to handle different data types and formats. The Data Handler is a set of data manipulation APIs designed to transform data from an application-specific structure into business objects. By configuring the Data Handler, users achieve further control over the system's behavior.

The Connector Configurator saves its configuration in XML format to a file with the `.cwc` extension. The connector receives its configuration during runtime by reading this file. Connectors can either read this file from the file system or receive it at initialization time from the server.

In addition to the Connector Configurator, users can also configure the WBIS collaborations and maps. Using the Process Designer, users can specify the collaboration properties and set values for these properties, thereby controlling the collaboration behavior runtime. Using the Map Designer tool, users can specify the map properties that control the runtime behavior of each map.

7.5.3 Migration options

Although system and component configuration is usually a manual task, a set of rules and a good knowledge base may assist in migrating configuration from the BizTalk platform to WBIS. A set of configuration wizards can be supplied to assist with system and components configurations. These wizards would take their default values and options from the existing BizTalk environment, thus helping the user with the configuration.

Export configuration file using XML format

BizTalk 2002 stores configuration information (channels, ports, documents, and so forth) in the BTM database. This information can be accessed directly (SQL queries) or programmatically via the BizTalk Messaging Configuration model. BizTalk document specifications and maps are stored in WebDav (only metadata is stored in the database, not the document specification itself). A configuration export wizard can be provided to export all relevant configuration information to an XML file. Once the entire relevant BizTalk configuration is available in an XML file, the next step would be to map this information to the corresponding WBIS components.

Migration options table

Table 7-5 summarizes the configuration technologies and migration options for the configuration part of the business integration platform.

Table 7-5 Migration option summary

Feature	BizTalk	WBIS
Underlying technology	- Configuration database holds information about channels, ports, documents, and so on. - Document specifications and maps are stored in WebDav. - A programming model is provided to access the configuration database.	- Component-specific properties. - Adapter configuration is backed with an XML configuration file. - Business Objects can be used for adapter configuration.
Development tool	BizTalk Messaging Manager UI	Connector Configurator
Major migration technique	A set of wizards to support system configuration, which get their inputs from the existing BizTalk configuration.	
Additional migration techniques	- The knowledge base can provide best practice rules for system configuration. - XML transformation may be suitable in some cases to translate BizTalk component transformation into its WBIS counterpart. This is possible specifically for the WBIS connector configuration file.	

7.6 Executing custom software components

Utilizing the open architecture of the business integration platform, custom software components can work as plug-ins to the platform's runtime in several locations along the business process execution path.

7.6.1 BizTalk approach

Custom components are developed as an extension of BizTalk to provide application-specific functionality. Extensions can be provided at different points of the solution execution path:

- Upon message arrival, during receive pipeline or receive function pre-processing or channel processing.
- During orchestration processing, within an orchestration step.
- When sending messages to the target application, during send pipeline or as an AIC invoked by an orchestration to send a message.

Custom functoids can be developed to provide specific transformations for a map.

BizTalk provides several built-in custom components, thus reducing the amount of code necessary to integrate applications.

7.6.2 WBIS approach

Using the Java language, the WBIS platform has the ability to execute additional external software components. With the Collaboration Designer, users can add Java code snippets to interact with basically any software component that has a Java interface. Users must pay special attention to long-lived processes, due to the stateless nature of the WBIS collaboration.

With Map Designer, users can add Java code to their maps. Map Designer supports interaction with external software components in the same way as the Process Designer.

7.6.3 Migration options

While migrating custom code from one platform to the other is not expected to be an easy and straightforward task, it is sometimes possible to accomplish using the interface adjustment and the black-box technique.

- **Interface adjustment and the black-box technique**

  By identifying and encapsulating the interfaces used by the BizTalk platform for custom code execution, it is possible to build an identical interface toward this custom code and execute it from within the WBIS platform. This approach may work only for cases where there is no platform change involved and the custom code can still run in the same execution environment.

- **Code migration**

  If no well-defined interface exists, or if an execution environment change prevents the original custom code from running, a code migration is needed. Code migration can be done using dedicated tools, which are beyond the scope of this redbook. Manual migration is one alternative to the automatic migration tools for custom code.

Migration options table

Table 7-6 summarizes the custom code migration options.

Table 7-6 Migration option summary

Feature	BizTalk	WBIS
Underlying technology	Interface implementation	Import and use external Java libraries
Development tool	Microsoft IDEs for VB, COM, .NET, and so forth	Java development environment
Major migration technique	Adapt BizTalk interfaces to the WBIS Java-based environment.	
Additional migration techniques	Custom code migration and manual migration.	

7.7 Summary

In this chapter, we divided the business integration solution into several functional and architectural parts. After comparing the BizTalk and WBIS approaches for each part, we suggested a set of techniques to migrate the part from BizTalk to WBIS. Subsequent sections of this redbook will provide further details on the exact methodology that can be used to migrate each part of the business integration solution.

Part 2

BizTalk to WBIS migration process

8

Collecting BizTalk artifacts

This chapter describes techniques for collecting BizTalk artifacts and migrating them to WBIS.

8.1 BizTalk artifacts overview

In BizTalk 2002, artifacts are not associated with projects. In fact, there is no “project” concept. The artifacts are stored in different location types (database, file system, and WebDAV repository) and there is no apparent relationship between them. Therefore, mining is essential both to locate BizTalk artifacts and to acquire a better understanding of the structure of the applications to be migrated. Once you have a clear picture of the existing artifacts, you can select the relevant artifacts to migrate.

When developing your migration toolset, the first step would be to write a collector component that collects all the existing BizTalk artifacts. Such a collector would perform the following main tasks:

1. Collect information about BizTalk artifacts from the BizTalk database.
2. Build a model (for example, an EMF Model) that reflects the current applications’ structures and their relevant artifacts.
3. Collect the selected BizTalk artifacts from the different location types into a migration project.

BizTalk artifacts reside in various location types, including:

- BizTalk database
- WebDAV repository
- Local file system

The following sections describe the nature of these locations and present a step-by-step scenario to demonstrate how you could automatically migrate BizTalk artifacts to WBIS using your own migration wizards (for example, Eclipse-based wizards).

8.2 Database artifacts

The BizTalk Messaging Management database, usually called InterchangeBTM, stores most of the BizTalk solution configuration information. This SQL Server database may reside on the same host as the BizTalk server or on a different host.

Your collector application could be written in Java and it could retrieve information from the BizTalk database via JDBC calls. To ensure that the SQL Server database can be accessed, the following requirements must be fulfilled:

- The host name where the SQL Server database resides must be known.

- The port number of the SQL Server must be known (the default is 1433).
- You must supply a valid SQL Server login account and password, with the proper access rights. You can add new logins using SQL Server Enterprise Manager. (Under the server name, expand the Security folder, right-click the Logins icon, and click New login).
- The SQL Server's authentication mode must be configured as mixed mode (for example, SQL Server and Windows). You can change the authentication mode using the SQL Server Enterprise Manager (Right-click the server name and click Properties, then go to the Security tab).

8.3 WebDAV artifacts

BizTalk schemas (document definitions and envelopes) and maps are stored in a WebDAV (Web Distributed Authoring and Versioning) repository. The URL links to these artifacts are stored in the BizTalk Messaging Management database (usually named InterchangeBTM).

Your collector application could be written in Java and it could retrieve documents from the WebDAV repository via HTTP calls. WebDAV is a set of extensions to the HTTP protocol, which allows users to edit and manage files on remote Web servers. To ensure that the WebDAV repository can be accessed, the following requirements must be fulfilled:

- The WebDAV Internet Information Services (IIS) server must be in the network, and up and running.
- You must provide a valid user name and password with the proper access rights to read WebDAV documents. To manage WebDAV security, use the IIS Manager. (Right-click the BizTalk repository folder and click Properties, then go to the Directory Security tab).

8.4 Artifacts from local directories

Information about the location of BizTalk orchestration artifacts (for example, host name and full path of the orchestration's .skx file) is stored in the BizTalk Messaging Management database. Your collector application must have access to this location in order to retrieve the artifacts.

If your collector application runs locally, on the same machine where the BizTalk orchestrations to be migrated are stored, these orchestrations could be retrieved directly from the local file system.

8.5 Migration process

In this section, we present an step-by-step scenario demonstrating how you could automatically migrate BizTalk artifacts to WBIS using your migration wizards.

Note: This section describes an existing BizTalk to WBIS Migration Wizard that is currently not part of the WBIS product.

The BizTalk to WBIS migration process consists of four steps:

Step 1: Open the WBIS System Manager

Assuming your migration wizards are WSAD plug-ins, the first step would be to open the WBIS System Manager.

Step 2: Create the migration project and define project properties

1. In the WBIS System manager, click **File** → **New** → **Other**. The selection dialog shown in Figure 8-1 is displayed.

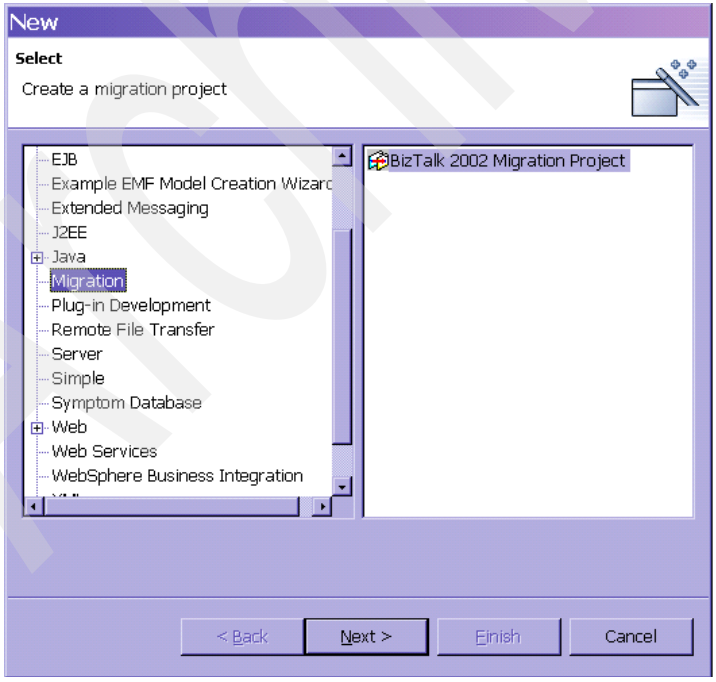

Figure 8-1 New migration project dialog

2. Select **Migration** in the left pane, then **BizTalk 2002 migration project** in the right pane.
3. Click **Next**. The first page of the New Migration Project wizard is displayed (Figure 8-2).
4. Type the name of your new migration project in the Project name field and click **Next**.

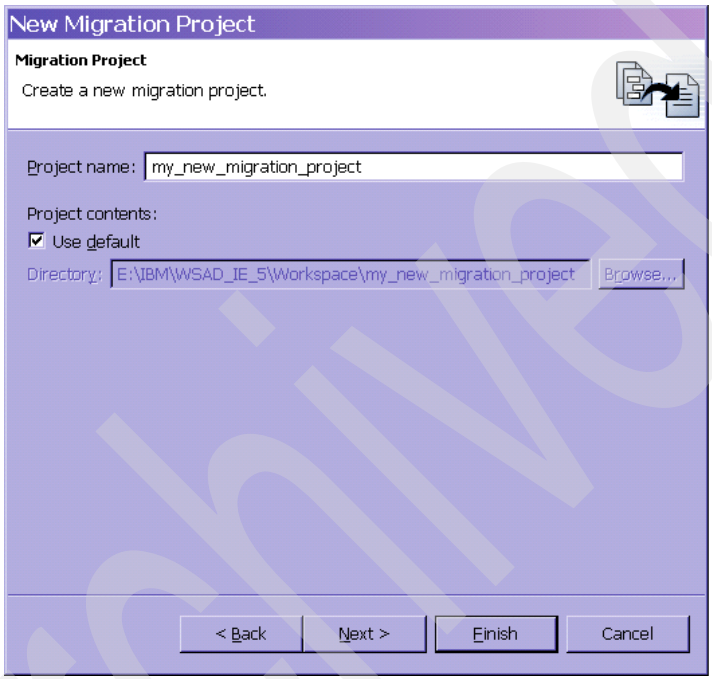

Figure 8-2 New Migration Project wizard - Page 1

5. The second page of the New Migration Project wizard is displayed (Figure 8-3). On this page, enter information about the location of the BizTalk database.
   - In the Database Server field, type the hostname of the SQL Server where the BizTalk database resides.
   - In the Server Port field, type the port number of the SQL Server.
   - In the Database Name field, type the name of the BizTalk database.

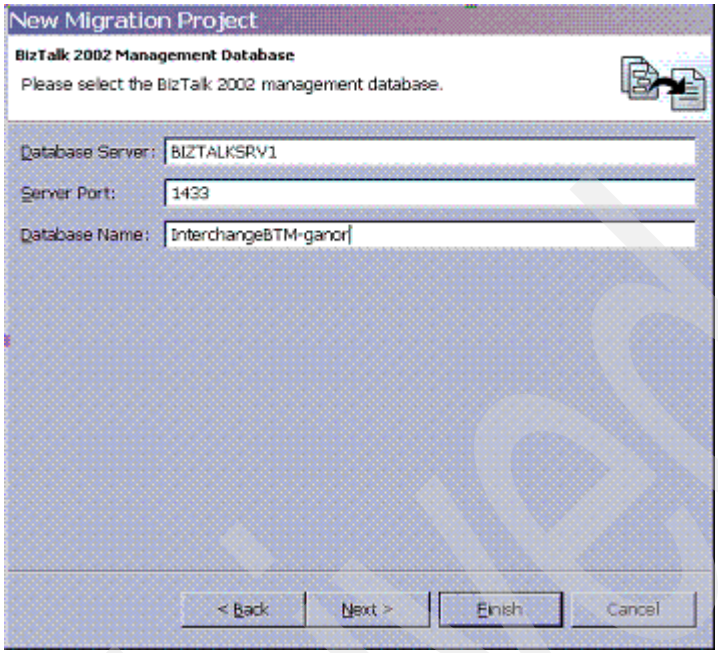

Figure 8-3 New Migration Project wizard - Page 2

In the example in Figure 8-3, the SQL Server resides on a machine named `BIZTALKSRV1`, it listens at port `1433`, and the BizTalk database is named `InterchangeBTM-ganor`.

6. Click **Next**.

   The last page of the New Migration Project wizard is displayed (Figure 8-4). On this page, enter information about user names and passwords needed to access both the SQL Server database and the WebDAV repository.

   – In the WebDAV User Name and WebDAV User Password fields, type the user name and password required to access the WebDAV repository.

   – In the Database User Name and Database User Password fields, type the user name and password required to access the SQL Server.

   – The Schedules Directory field is only required if the original XLANG schedules to migrate do not reside in the local machine (for example, the machine that runs this wizard). If this is the case, type the full path name of a local directory where you manually copied the non-local .skx files to be migrated.

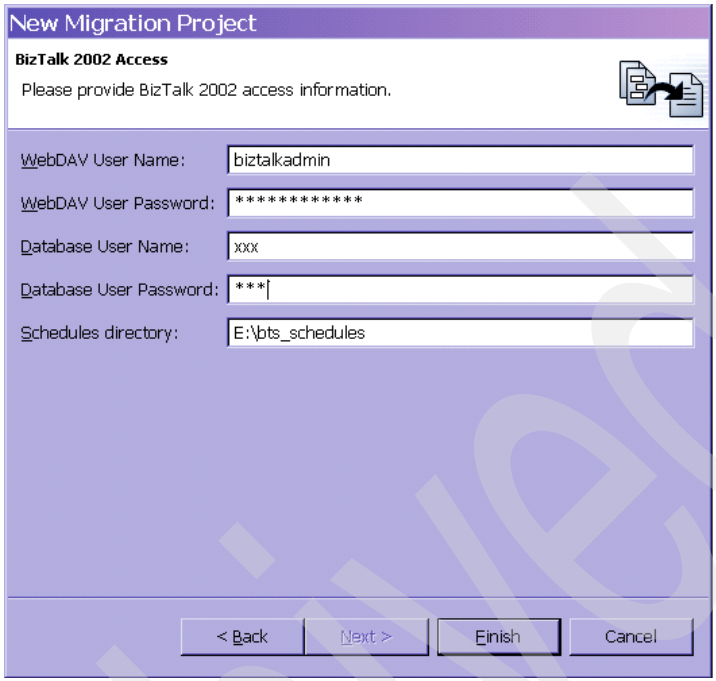

Figure 8-4 New Migration Project wizard - Page 3

In the example in Figure 8-4, `biztalkadmin` is both the user name and password with access to the WebDAV repository, and `xxx` is both the user name and password with access to the SQL Server database. Because the migration wizard is not running on the same machine where BizTalk is running, all XLANG schedules to be migrated were manually copied to `e:\bts_schedules`.

7. Click **Finish** to close the New Migration Project wizard.

   The new migration project is displayed in the migration perspective. It has three empty directories, named Maps, Schedules, and Schemas, and an EMF model file named btsmodel.model. The results for your New Migration Project wizard should look similar to Figure 8-5.

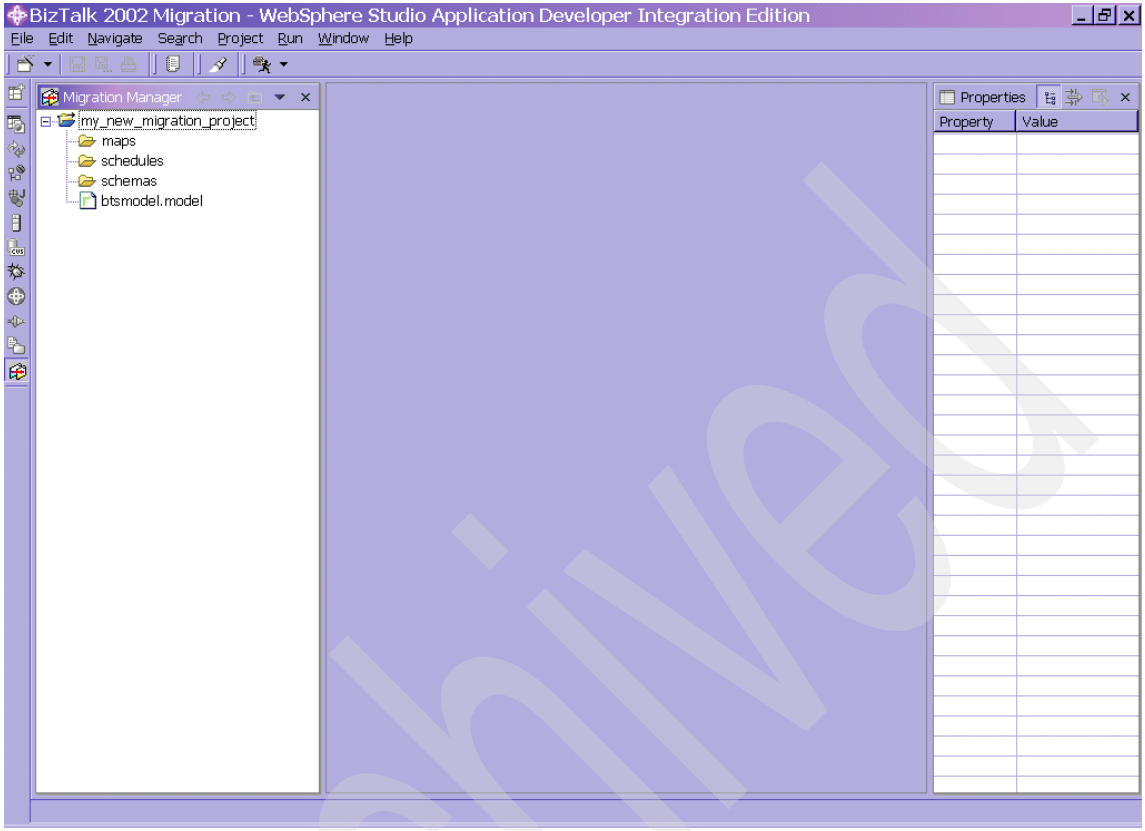

Figure 8-5 New migration project

At this point, you have successfully created a new migration project. This new project will not yet contain any resources, but it will contain an EMF model showing the various BizTalk artifacts as they exist in the BizTalk repositories (WebDAV, the BizTalk database, and local directories).

To view the EMF model, right-click the btsmodel.model file, then select **Open With → Model Model Editor**. A navigation tree view of all BizTalk elements is displayed in the migration perspective.

Note: The Properties pane on the right contains information about the currently selected tree node. By navigating the EMF model, you get a full picture of all the available BizTalk resources. This way, you can make informed decisions on what resources need to be migrated.

In the example in Figure 8-6, you can see a list of BizTalk ports, schedules, and documents. The port `P_FundInvestors` is expanded and its channel `C_FundInvestors_HenryFonda` is selected. The Property pane on the right shows the channel information, specifically the input and output documents, which are both named `FundInvestors1`.

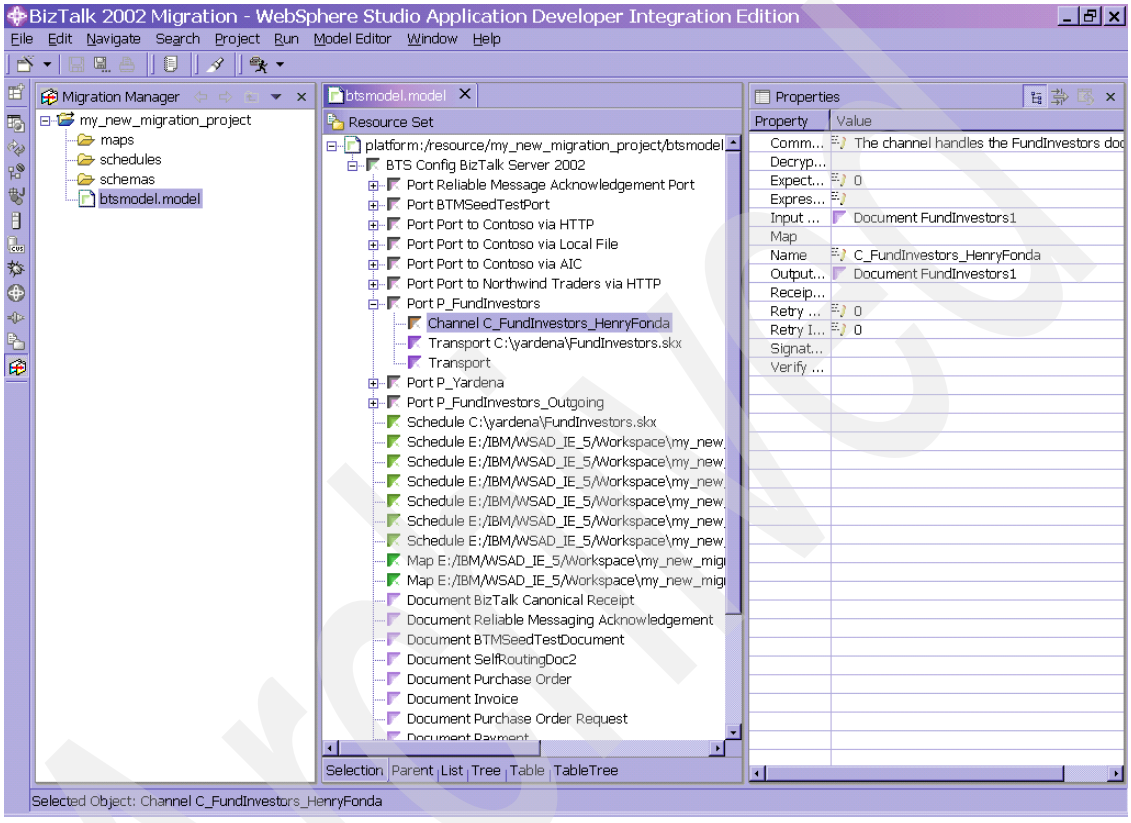

Figure 8-6 Example of a migration project

Step 3: Choose and retrieve BizTalk artifacts

The next step in the migration process is deciding which BizTalk artifacts are relevant for migration, and retrieving them from the various BizTalk repositories into the recently created migration project. Resources can be collected incrementally or all at once.

1. To start the import wizard, right-click the recently created migration project in the Migration Manager pane and select the **Import BizTalk 2002 resources** menu item. The Migration Import wizard opens and shows a selectable tree of BizTalk resources.

In Figure 8-7, you can see that the BizTalk document `SelfRoutingDoc3` is selected.

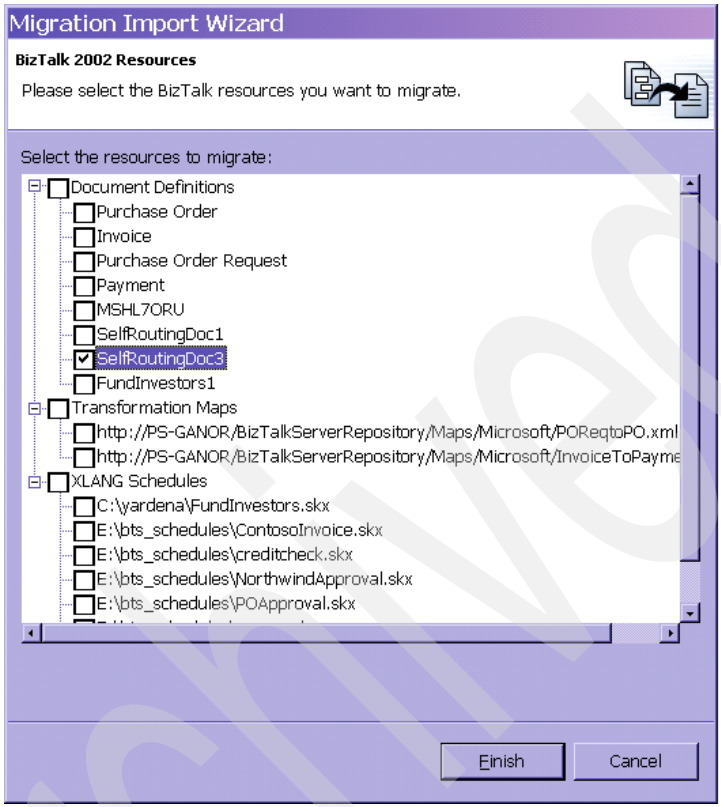

Figure 8-7 Migration Import wizard

2. Select the desired resources.
3. Click **Finish**.

   After you finish using the Migration Import wizard, the selected BizTalk artifacts will be stored as a local copy under the recently created migration project. Document (for example, Message) definitions will be stored under the Schemas directory, transformation maps under the Maps directory, and XLANG schedules will be stored under the Schedules directory.

   You can invoke the Migration Import wizard several times, until all desired resources are copied to the migration project.

Step 4: Migrate the relevant BizTalk artifacts into the destination WBIS Integration Component Library (ICL) project

In the last step of the migration process, migrate the resources to a target ICL project by invoking the Migration Export wizard.

1. To start your Migration Export wizard, right-click the recently created migration project in the Migration Manager pane and select **Export BizTalk 2002 resources to ICL menu** item.

   The Migration Export wizard opens, showing a selectable tree of BizTalk resources.

2. Select the desired resources.
3. Select the target ICL project.
4. Click **Finish**.

   Figure 8-8 shows an example of a Migration Export wizard execution where the BizTalk document `SelfRoutingDoc3` and the target ICL project `ICL_Project1` are selected.

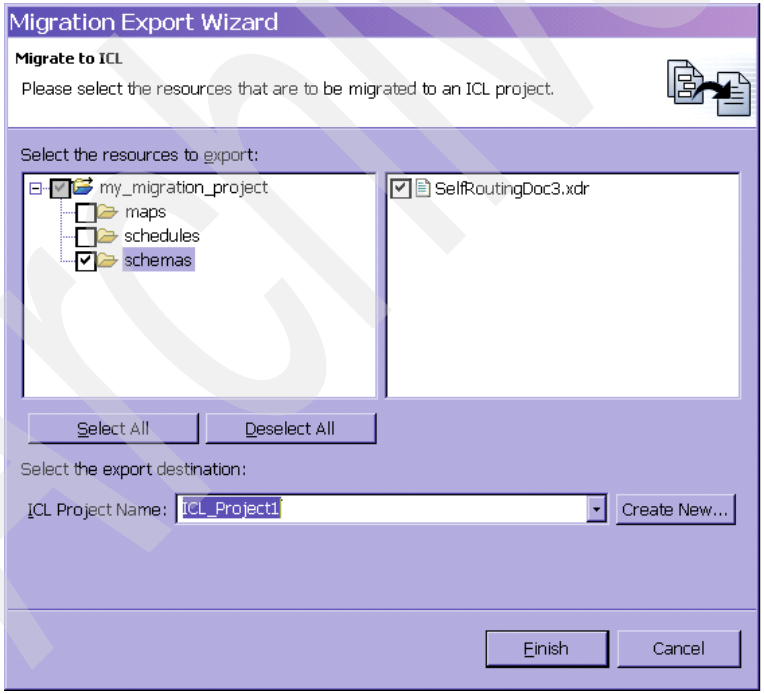

Figure 8-8 Migration Export Wizard

After you finish using the Migration Export wizard, the selected WBIS ICL project will hold the migrated artifacts as standard WBIS artifacts.

In the example in Figure 8-9, the BizTalk Document `SelfRoutingDoc3` was migrated into three WBIS business objects.

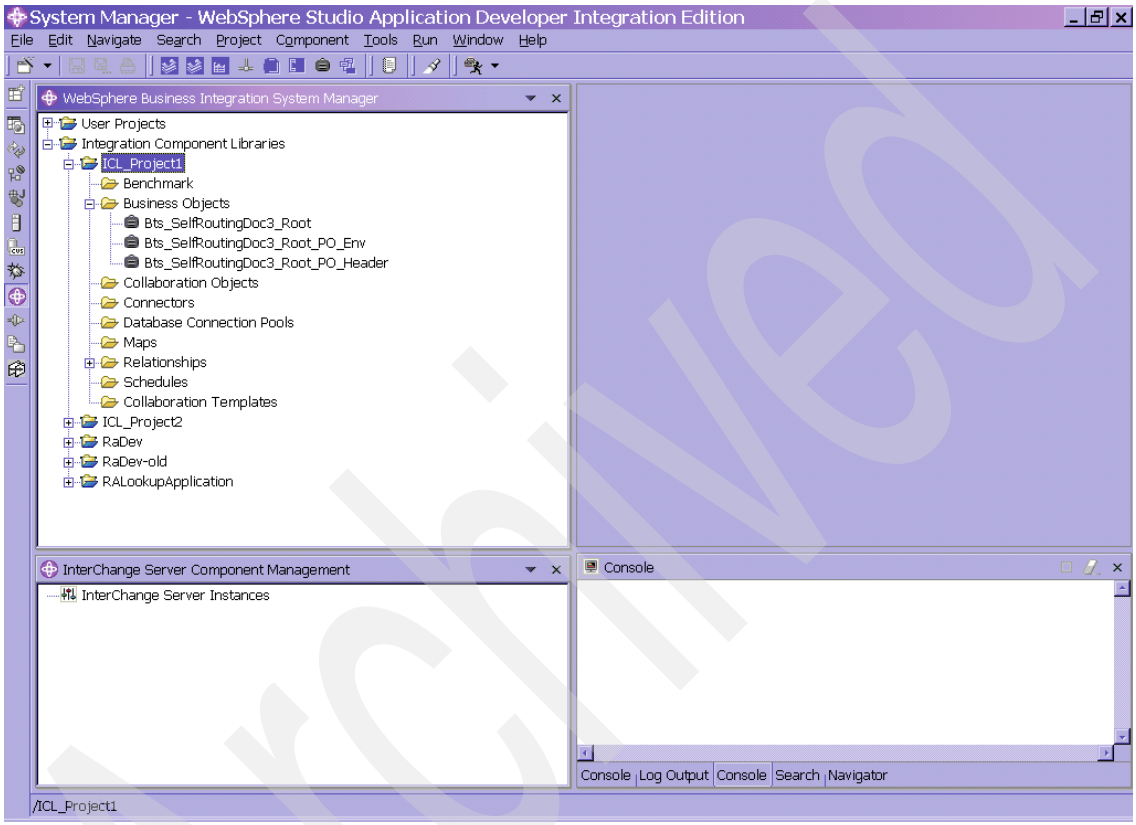

Figure 8-9 Example of a migration project

Note: You must have a working WBIS System Manager with the wizard plug-ins in order to use the migration toolset.

9

Migrating messages

This chapter discusses the BizTalk messages, WBIS messages, and use of the Messages Migration Wizard.

9.1 BizTalk messages

BizTalk uses various methods and interfaces for sending and receiving messages. BizTalk may receive documents through one of three receive functions (File, MSMQ, HTTP). COM-aware applications may submit documents directly to BizTalk via the IInterchange COM interface.

BizTalk supports various message formats. The basic BizTalk message format is the XML message. This XML message must comply with an XML Data Reduced (XDR) schema. BizTalk uses other message formats such as flat file, Electronic Data Interchange (EDI), and X12.

BizTalk messages that don't comply with the XML standard (for example, flat file or EDI messages) can flow only through the BizTalk messaging part. BizTalk orchestration can handle only XML-based messages. BizTalk messaging must transform a message into XML format before sending it to BizTalk orchestration. This transformation can be done by a ready-made or a user-defined parser.

BizTalk also uses messages to interact with external objects from within the orchestration. It interfaces with these external objects by using input and output messages. These messages are constructed based on a well-defined schema.

9.2 WBIS messages

WBIS uses the business object (BO) data format for internal data representation across the entire data path. A BO must comply with an XML schema. The XML schema must be available to the WBIS or the acting adapter, while creating the BO instance corresponding to schema.

9.2.1 Data handler

WBIS uses the DataHandler API to create a BO from a given data format. For example, a DataHandler may be used to generate a BO from a Comma Separator Values (CSV) file format or from an SQL result set returned by an SQL query. The DataHandler logic uses application-specific information (ASI) inside the BO (in the XML schema of the BO instance) to generate the correct BO from a given data format. WBIS also uses the ASI in the opposite direction while serializing the BO into the correct application-specific data format.

9.3 Using message migration wizard

A message migration wizard lets you migrate BizTalk XDR-based message types into WBIS business objects. (See 8.5, “Migration process” on page 100 for more details.)

A migration wizard transforms XML files from XDR schema structure into BO schema structure. You can write your own migration wizard based on available transformation technology like XSLT, or by using programming language such as Java.

While executing the migration wizard on a set of XDR schemas (generated by the BizTalk editor), the wizard generates the corresponding BOs. Note that many BOs are usually generated from a single XDR schema.

9.4 Migrating flat file messages

The BizTalk editor provides parsing capabilities that let you convert structured file formats into XML format. You provide the parsing instructions for records and fields in the input file. The BizTalk editor keeps these directives along with the generated XDR schema for the incoming file.

WBIS uses a data handler to parse a flat file into a BO. The data handler, usually executed at the adapter, receives the structured file and translates it to BOs using directives. The WBIS provides these directives to the adapter using a DataHandler meta-data BO.

10

Migrating application connectivity components

This chapter describes how to migrate application connectivity components.

10.1 BizTalk approach for application connectivity

BizTalk uses various methods and interfaces for sending and receiving messages. BizTalk can receive messages through one of three receive functions (File, MSMQ, HTTP) or by using the Interchange interface.

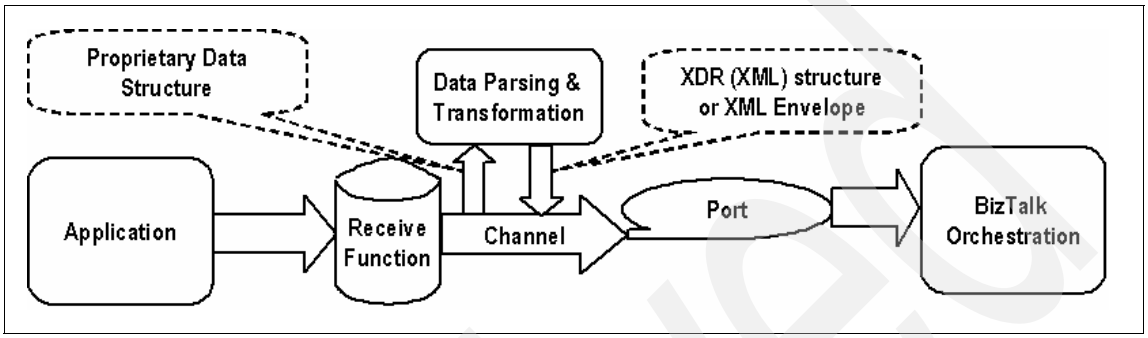

Figure 10-1 BizTalk receive function data path

Figure 10-1 shows a sample BizTalk data path where the application submits messages through the BizTalk receive function. The message is then picked up from the receive function and passed through the BizTalk channel. In the channel, BizTalk can parse the message and transform it into XDR format or use an Envelope to add any required meta information. BizTalk then sends the message to the Message port and to its destination, which in this case is a BizTalk orchestration.

10.2 WBIS approach for application connectivity

The WBIS interacts with external software components using an adapter. The adapter's main function is to transform application-specific data into BOs and vice versa. To fulfill this task, the adapter uses the DataHandler and the ASI within the BO (Figure 10-2).

The WBI adapter is responsible for converting any incoming data format to the standard BO format. Further BO transformations (for example, from application-specific BO to generic BO and vice versa) may be executed by various WBIS runtime components along the data path.

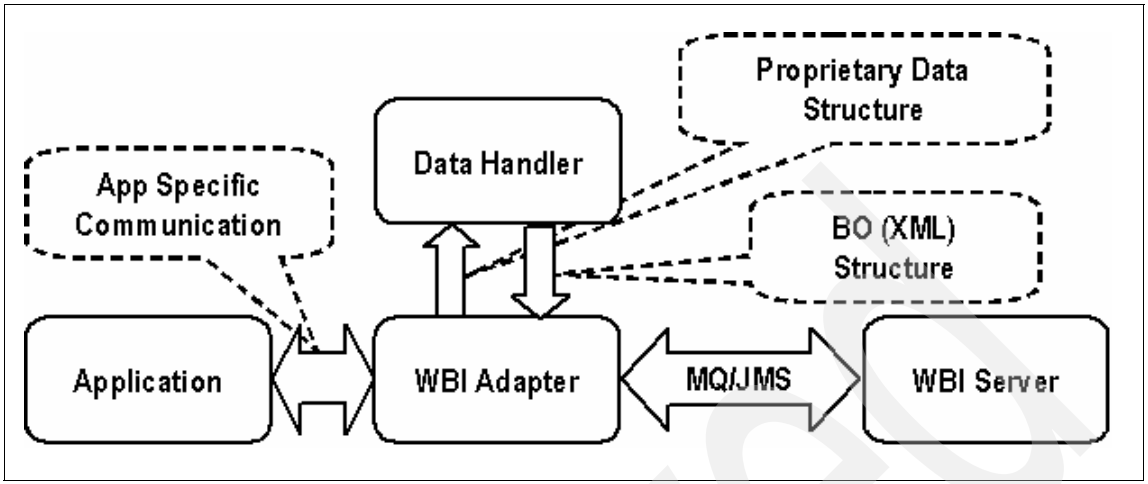

Figure 10-2 WBIS application connectivity architecture

The WBI adapter is also responsible for translating the WBIS communication protocol into an application-specific communication protocol.

10.3 Migrating receive functions, channels, and ports

BizTalk uses the receive function to read messages from some location (queue, file, URL) and submit them to a messaging channel (Figure 10-3).

This channel is then used to route messages to a messaging port and an orchestration.

After completing the processing, the orchestration calls one or more channels that lead to outgoing ports in order to send its output data to the application. These ports are bound to the queue or file location from which data is sent to the destination application.

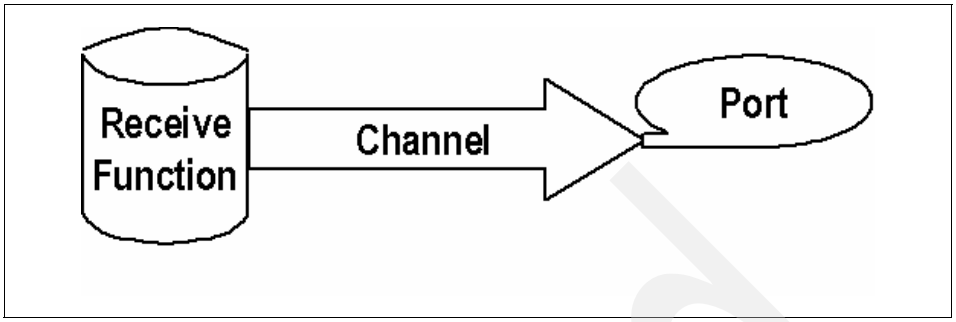

Figure 10-3 BizTalk Receive function, channel, and port

WBIS uses adapters and data handlers for message handling. Each BizTalk receive function, channel, and port combination is replaced by a suitable WBI adapter, where the adapter type is decided according to the type of location (message queue, file, URL) from which WBIS reads or writes the messages. The WBI adapter collects information from the application and sends information back to the application. WBIS then routes the BOs (WBIS messages) to the appropriate destinations, based on BO name and verb.

Table 10-1 Message handling

BizTalk receive functions	WBI adapters
Queue receive	JMS adapter; MQ adapter
File receive	JText adapter with the appropriate DataHandler (according to file format)
HTTP (conventional) receive	JText adapter with XML DataHandler
HTTP file receive	JText adapter with the appropriate DataHandler (according to file format)

10.4 Advanced routing features

In this section, we focus our discussion on several advanced routing features.

10.4.1 Many-to-one

BizTalk messaging provides flexible routing capabilities for incoming messages. Messages coming from various receive functions and channels may be sent into the same port and orchestration (Figure 10-4).

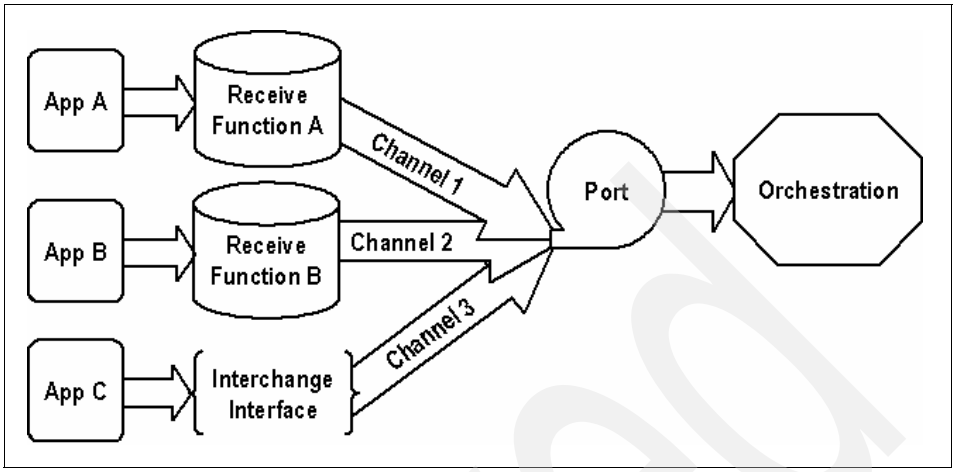

Figure 10-4 BizTalk many-to-one routing

The current WBIS architecture enforces rigid associations between an adapter and a collaboration. This association is set at design time. A specific collaboration port can only receive BOs coming from a specific adapter.

Using multiple scenarios is the best way to support communication between many applications and one collaboration (Figure 10-5). Each scenario will have its input port connected to a different adapter, while all scenarios will send their output to the same port. This port is connected to the collaboration that implements the required business process.

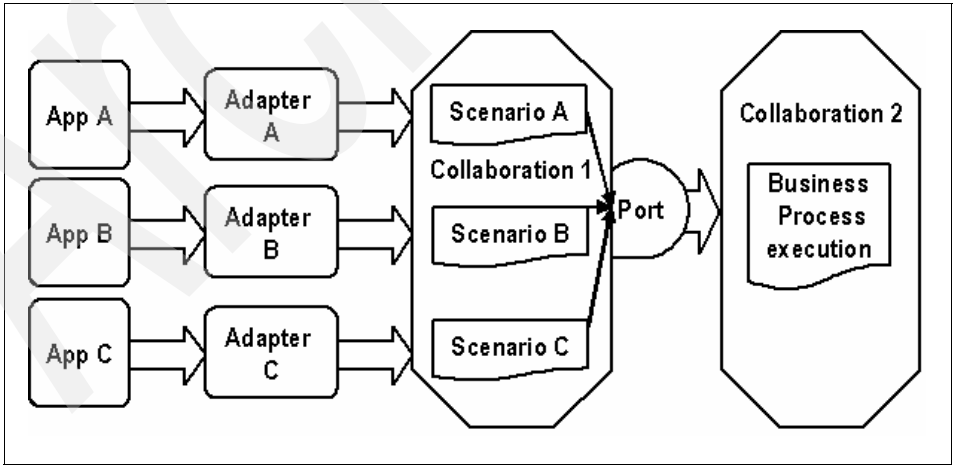

Figure 10-5 WBIS collecting collaboration for solving the many-to-one problem

10.4.2 One-to-many

BizTalk allows you to send an orchestration's output to a list of destinations using a distribution list (Figure 10-6). The distribution list receives the document from a channel and sends it out to a list of destination ports.

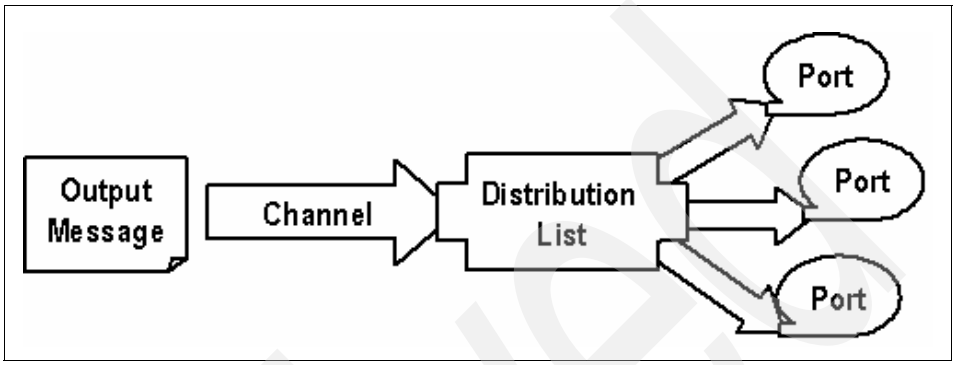

Figure 10-6 BizTalk distribution list

WBIS does not have built-in support for distribution lists. To emulate the distribution list functionality, you need to create a dedicated collaboration that mimics the distribution list functionality. This collaboration has a list of service calls connected to output ports and to the relevant adapters (Figure 10-7). You need to pass every input BO that needs to be sent to multiple destinations into this collaboration. The collaboration then dispatches it to the appropriate destinations.

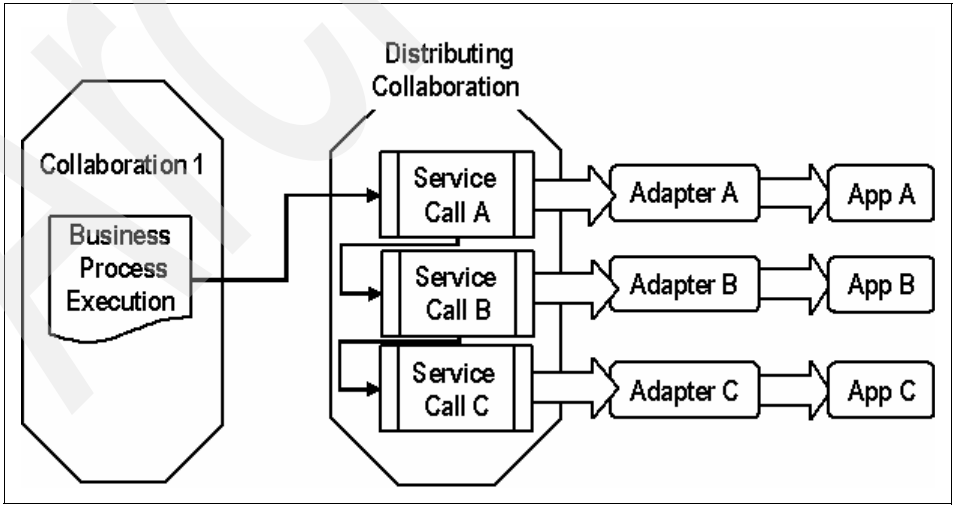

Figure 10-7 WBIS distributing collaboration

10.4.3 Content-based routing

BizTalk Messaging supports content-based routing (CBR). In CBR BizTalk runtime chooses the correct channel and port based on data inside the incoming message (for example, document source, destination, and document specification).

WBIS associates incoming BOs with collaborations based on the BO name and verb. With WBIS, these associations are created during development. Associations are static and cannot be changed at runtime. It is important to note that although BizTalk resolves incoming messages to channels at runtime, all possible CBR values within a document are known at development time. Two main options are available when you are migrating an application that uses CBR:

1. You can add specific verbs into the BO verb list, where each verb describes the possible CBR value of a document. You then need to create a map that transforms the relevant BO attributes inside the incoming BO into the required BO verb (Figure 10-8). If you created a collaboration that subscribes to the BO name and the new verb, WBIS runtime will deliver the newly mapped BO to the collaboration.

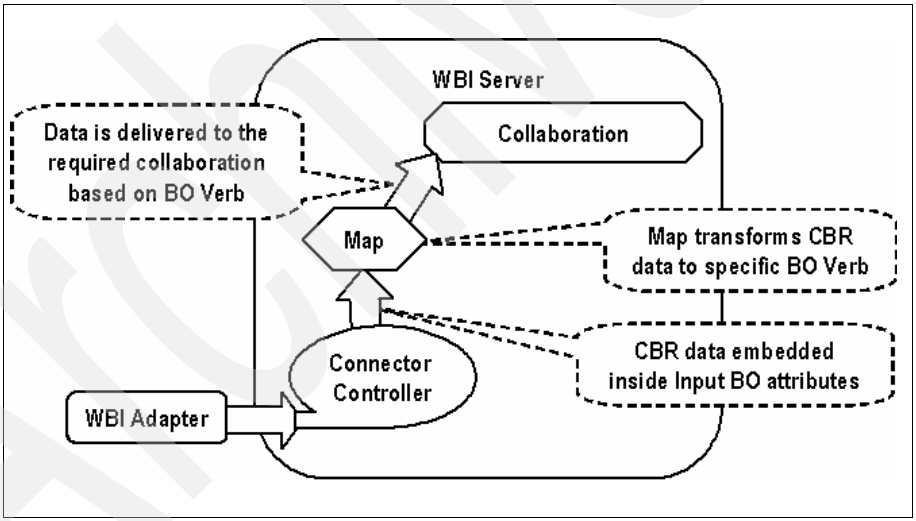

Figure 10-8 Using BO verb and a map for content-based routing

2. You can use a decision node inside a collaboration to implement the required CBR (Figure 10-9 on page 120). The collaboration receives the incoming BO, queries its content, and routes it to its destination.

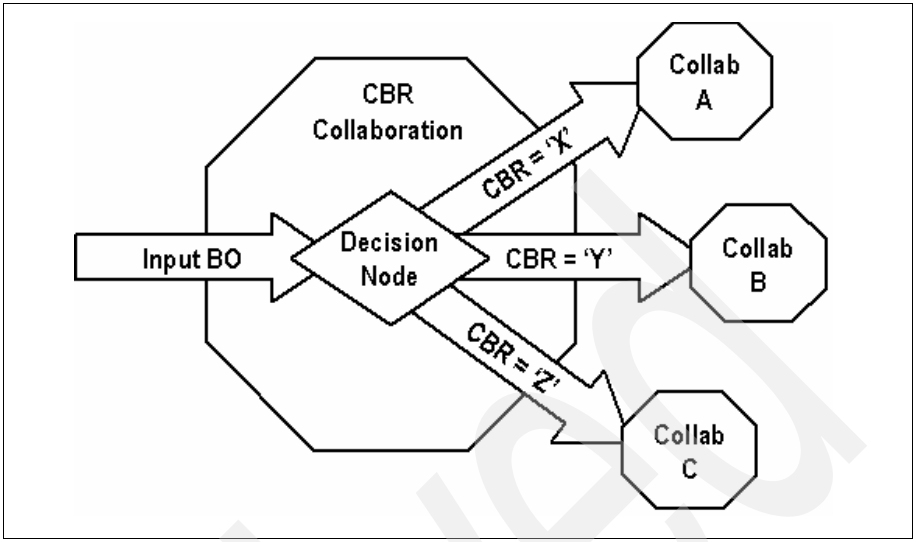

Figure 10-9 Using decision node in a collaboration for content-based routing

11

Migrating transformations

This chapter discusses the BizTalk transformation to the WBIS map.

11.1 BizTalk transformation

BizTalk lets you transform one message format to another using the BizTalk Mapper tool (Figure 11-1). The BizTalk Mapper is based on XSLT (Extensible Stylesheet Language Transformation).

BizTalk saves the output transformation artifact into the WebDAV repository. Each transformation artifact is associated with a channel. The channel transforms every passing document from source to target, using the transformation artifact.

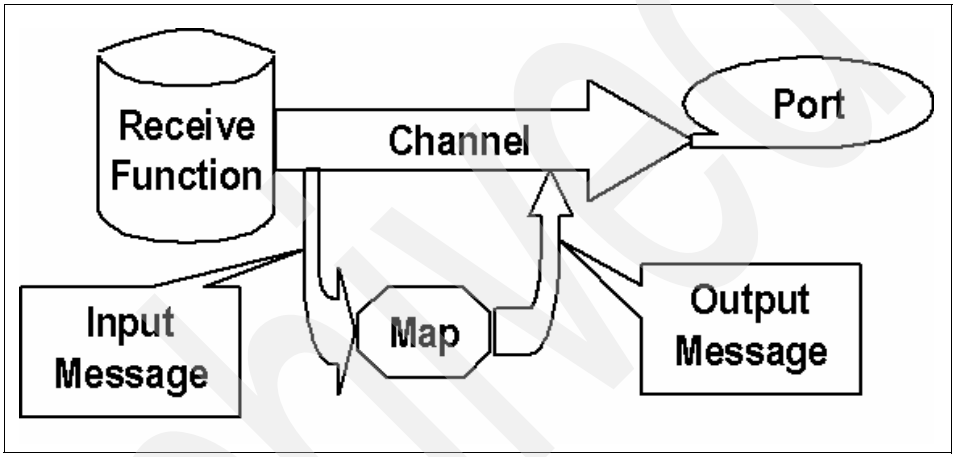

Figure 11-1 Channel transforms each passing message

The basic BizTalk transformation is the Link operation. Link copies data from the source field to the target field without making modifications. You have to use a BizTalk functoid for any other kind of data transformation. The BizTalk Mapper supplies a wide range of functoids in nine subject areas, and includes a template for adding custom functoids.

The BizTalk Mapper output artifact is an XML-based file that contains an XSLT segment. BizTalk parses the XSLT segment at runtime to perform the simple Link operation or to execute the functoid. The functoid's script appears as code fragments in this file. These fragments are linked into the rest of the XML structure. The BizTalk server executes these code snippets together with the rest of the XSLT at runtime.

11.1.1 WBIS mapping

WBIS mapping lets you transform one BO to another, using the WBIS Map Designer tool. WBIS uses the term generic business object (GBO) to describe a

BO that has no application dependency. WBIS uses the term application-specific business object (ASBO) for a BO that depends on a specific application or technology. WBIS maps generally transform ASBOs to GBOs and vice versa. The WBIS runtime implicitly executes the map in one of the following ways: after receiving an ASBO from an adapter and before sending it to the subscribed collaboration (ASBO to GBO transformation); or, after receiving it from a collaboration and before sending it to the adapter.

WBIS users also have the option of explicitly executing WBIS maps programmatically from various places within the WBIS environment (Figure 11-2).

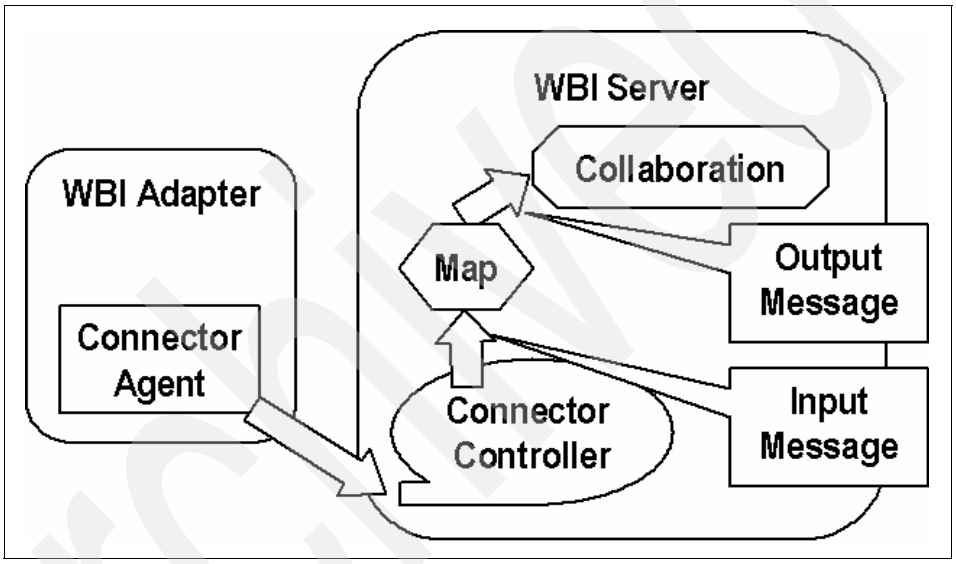

Figure 11-2 WBIS map execution process

WBIS supports several primitive transformations: Move (the equivalent of BizTalk's Link), Split, Join, and Set Value. WBIS supplies custom made transformations in addition to the primitive ones. The WBIS Activity Editor is a GUI based tool for creating transformations. You can also use any required Java code to define actions not included in the Activity Editor.

11.2 BizTalk functoids and WBIS activities

WBIS has support for only a subset of the BizTalk functoids (see Appendix A, "BizTalk functoids and WebSphere Business Integration activities" on page 155). You can add your own set of activities to match BizTalk functoids. These new

migration activities can be added into the Activity Editor under a location such as general/migration/functoids.

You can also add a generic activity to transform any functoids. This activity can be based on technology such as Bean Scripting Framework (BSF) and can allow you to execute any Visual Basic script (or any other script written in scripting language supported by the BSF) under WBIS. BSF can be used only in cases where the WBIS runtime is running over Windows.

Figure 11-3 shows a generic approach for using the Activity Editor to execute a script from within a WBIS map.

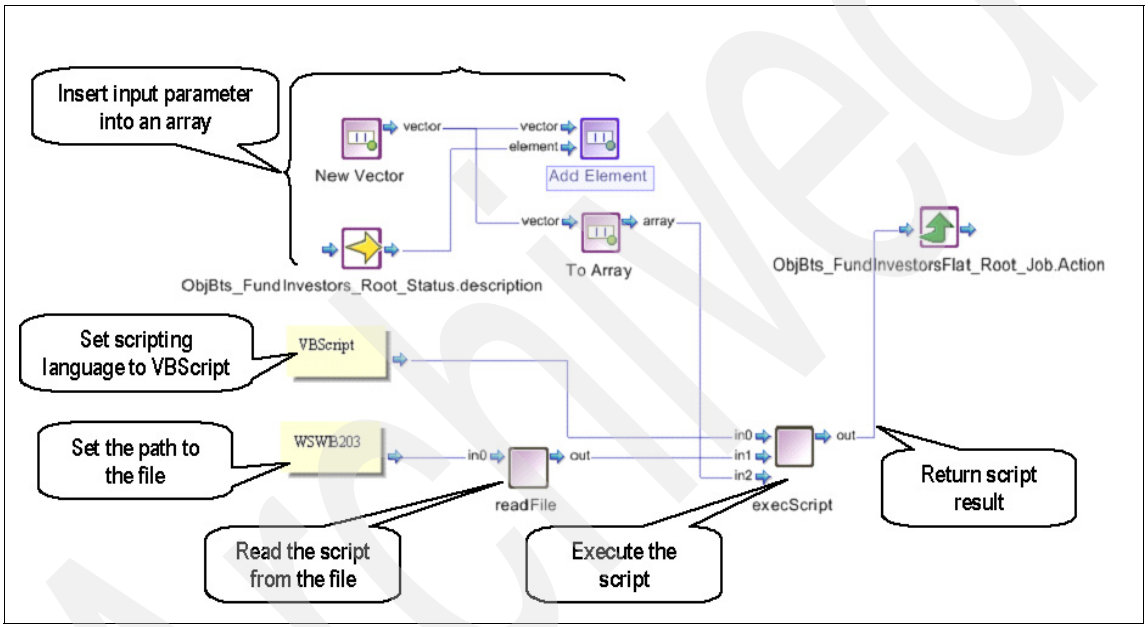

Figure 11-3 Executing script using the Activity Editor

Assuming you have your script saved to a file, you first need to create an array to hold all your input parameters. Next, you need to read the script's file and send its contents together with the input parameters array and the language name (for example, VBScript, JScript, and so forth) as inputs to your Generic Activity component. The Generic Activity component executes your script and returns its result to the output port, which you can assign to any output parameter as needed.

Configuring WBIS for BSF support

To configure WBIS for BSF support, perform the following steps:

1. Import scripts.jar and bsf.jar to the ICS design time tools:
   a. Open System Manager by selecting **Start** → **All Programs** → **IBM WebSphere Business Integration Express** → **Toolset Express** → **Administrative** → **System Manager**.
   b. Open the activity settings view by selecting **Window** → **Show View** → **Other** → **WebSphere Business Integration System Manager** → **Activity Settings**.
   c. Right-click **BuildBlock Libraries** and select the **Add Library** menu item.
   d. Select **scripts.jar** file and add it.
   e. Select **bsf.jar** file and add it.

   Note: The generic activity and the jars are currently not part of the WBIS product.

2. Import scripts.jar and bsf.jar to the ICS runtime:
   a. Add the BSF dlls (bsfactivescriptengine.dll, bsfactivescriptengine_g.dll and msvcp60.dll) to the PATH environment variable.
   b. Add the Java packages (scripts.jar and bsf.jar) to the CLASSPATH environment variable. The recommended way to import third-party package libraries to ICS is to add the corresponding jar files to the JCLASSES variable in the script used to start the ICS server (for example, start_server.bat). For example, if the jar files to be imported are stored in directory c:\, add the following lines to the script used to start the ICS server:

      ```
      set BSF_JARS= c:\scripts.jar;c:\bsf.jar
      JCLASSES=…existing jars ….;%BSF_JARS%
      ```

   c. Restart the ICS server.
3. Once you successfully imported the jars and dlls to ICS, you can use your generic activity in the Activity Editor. This activity invokes a given script with a given set of parameters and returns its corresponding result.

   Figure 11-3 shows an example of how to use the generic activity.

12

Migrating business processes

This chapter describes how to migrate business processes utilizing the BizTalk Orchestration Designer.

12.1 BizTalk orchestration

You can create a BizTalk orchestration using the BizTalk Orchestration Designer. The Orchestration Designer uses Visio as its user interface.

The Orchestration Designer has two tabs: the Business Process tab and the Data tab.

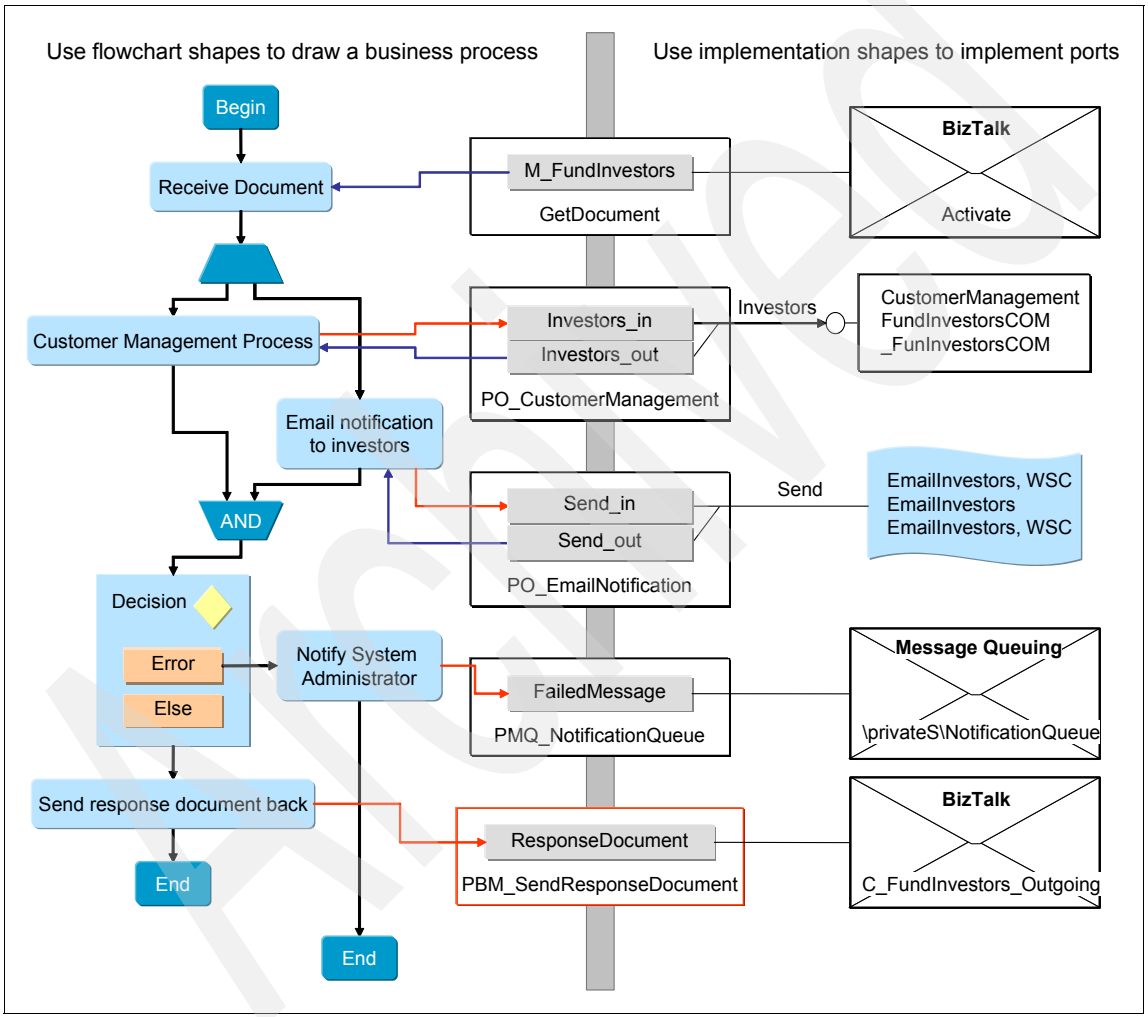

Figure 12-1 BizTalk Orchestration Business Process tab

Under the Business Process tab, you define the data flow between processing units (Figure 12-1).

Under the Data tab, you define data input and output for each processing unit (Figure 12-2). There are no processing capabilities in the Orchestration Designer itself. All processing is done using external components (usually COM components). The Data tab currently supports only moving fields from one message object to another. It does not support other, more advanced, mapping capabilities.

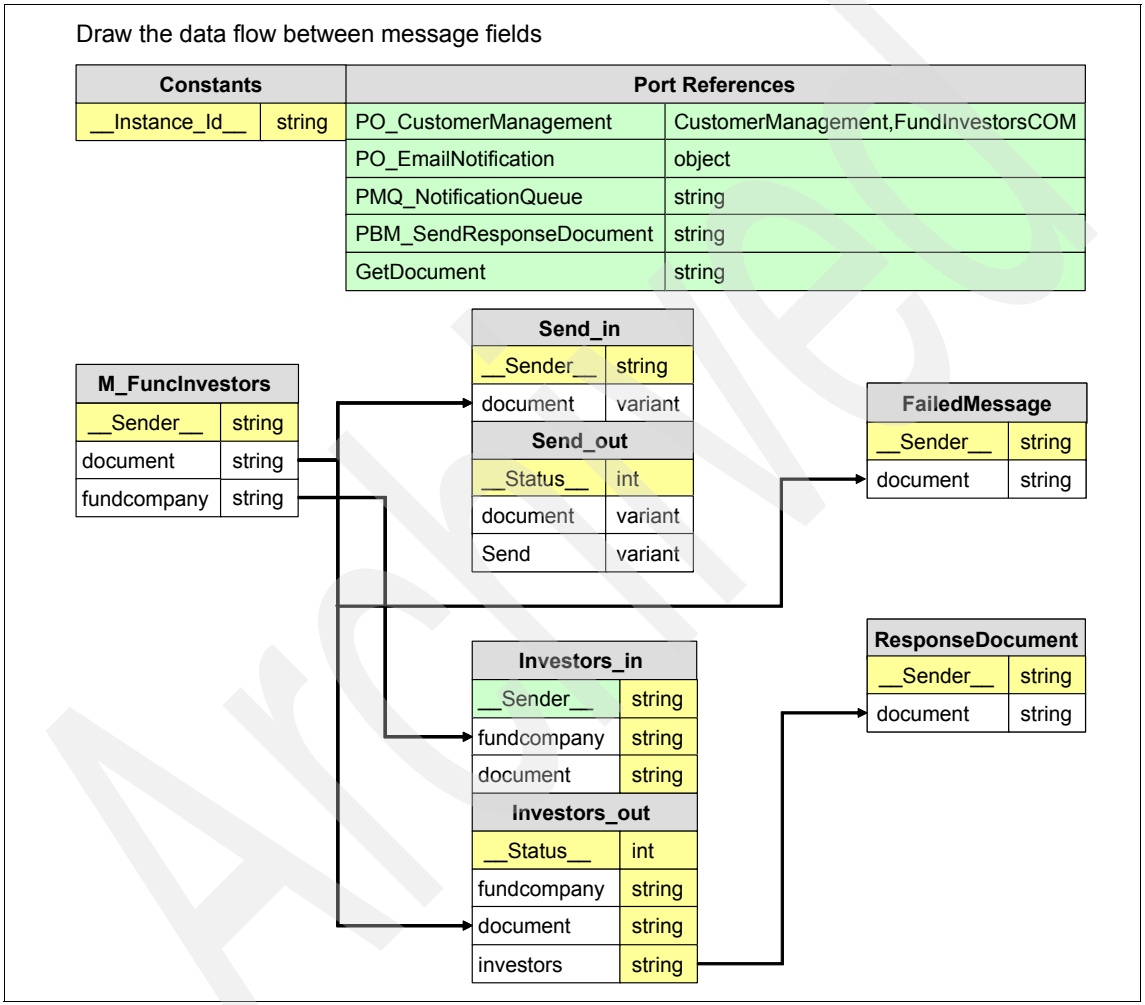

Figure 12-2 BizTalk Orchestration Data tab

The orchestration receives its input from the BizTalk messaging process through an input port. It can send its output back to BizTalk messaging or directly to MSMQ.

After completing the orchestration, you need to compile it into an XLANG file. XLANG is an XML-based process description language. The output XLANG file is then executed by the BizTalk server runtime. The generated XLANG file contains all the information about the orchestration from which it was created.

12.2 WBIS collaboration

You can use the WBIS Process Designer (Figure 12-3) to create the WBIS collaboration. The collaboration can contain multiple scenarios.

Note: There is no BizTalk orchestration equivalent to a collaboration scenario.

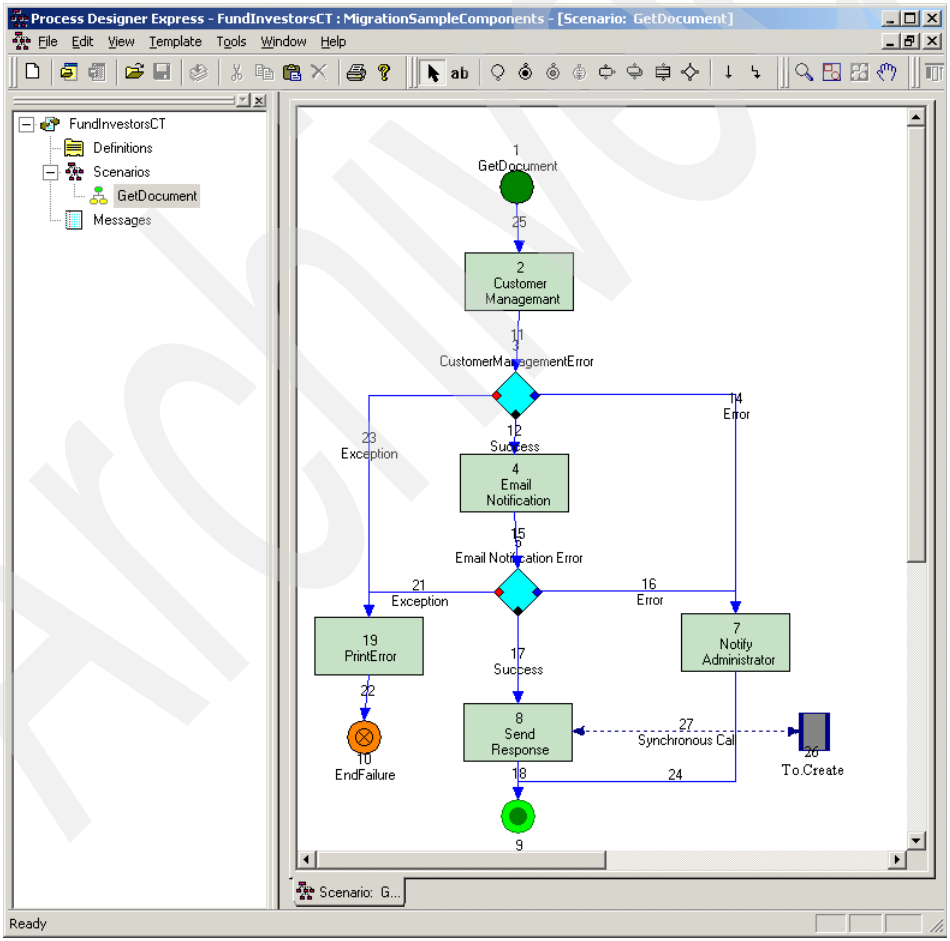

Figure 12-3 WBIS Process Designer

The collaboration receives its input from the Connector Controller in a process that is hidden from the user. To receive its input, the collaboration subscribes to a specific BO and a verb. Each collaboration scenario can subscribe to a different BO and verb.

A collaboration has two main groups of entities:

- Activities
- Service calls

The collaboration uses activities to execute all types of logic. For example, you can add information to a receiving BO in the activity.

The collaboration uses a service call to interact with any application or logical entity outside of the WBIS server itself. For example, you can send the received BO to a message queue using a service call to the MQ adapter. You can also use a service call to invoke other collaborations.

The Process Designer saves the created collaboration to an XML file, with a WBIS proprietary format and a .cwt extension. The .cwt file contains both GUI information and logical information about the collaboration. After creating the collaboration, you need to compile it. This compilation generates Java code that can be executed by the WBIS.

12.3 Migrating orchestration into collaboration

The migration from orchestration to collaboration needs to be done manually. This section describes the process.

Migrating the Business Process tab

The Business Process tab in the BizTalk Orchestration Designer contains two shape groups (Figure 12-4 on page 132): the flowchart shapes (to the left of the of the Designer's work area) and the implementation shapes (to the right).

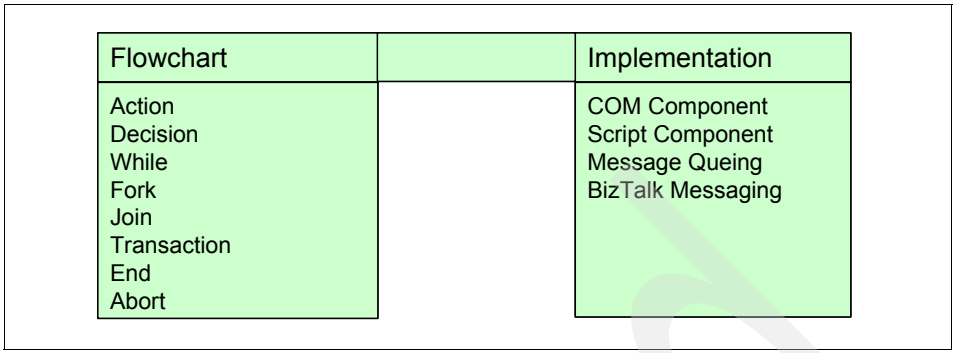

Figure 12-4 Sample flowchart and implementation in BizTalk orchestration (in text)

The BizTalk orchestration's flowchart shapes are mapped onto WBIS flowchart shapes, while BizTalk orchestration's implementation shapes are mapped onto WBIS service calls.

12.3.1 Migrating flowchart shapes

Table 12-1 summarizes the correlation between BizTalk orchestration's flowchart shapes and WBIS collaboration's flowchart shapes.

Table 12-1 Flowchart correlation

BizTalk Orchestration	WBIS Collaboration	Description and migration action
Begin	Start	The point where execution begins. There is only one starting point in any given collaboration scenario or orchestration. The orchestration receives its input by binding to a port. Messages coming through this port are used as the orchestration inputs. The orchestration gets the input explicitly by using an implementation shape. The collaboration scenario receives its input by subscribing to the business object name and a verb. The collaboration gets its input implicitly from the WBIS server runtime. **Migration action:** Substitute Begin with Start. Configure the collaboration scenario for subscribing to the BO and verb that represent the orchestration input.

BizTalk Orchestration	WBIS Collaboration	Description and migration action
Action	Action	Execute some kind of action/process. Action in a BizTalk orchestration is used only as an anchor for binding an implementation shape. Action in a WBIS orchestration can hold any Java logic. The orchestration's implementation shape is usually substituted with a collaboration service call. **Migration action**: Substitute the BizTalk orchestration Action with a collaboration Action followed by a service call. The collaboration action must prepare the BO to be sent to the service call.
Decision	Decision	If-then-else logic. WBIS's Decision logic allows up to seven decision branches. It provides a default branch and an exception branch. **Migration Action**: Substitute the BizTalk orchestration's Decision with the WBIS collaboration's Decision.
While	Iterator	Execute actions in a loop. **Migration action**: Substitute BizTalk orchestration's While with the WBIS collaboration's Iterator.
Fork	No equivalent	Split the execution process into multiple threads and execute them concurrently. The BizTalk orchestration allows you to split the execution path into multiple branches that can be executed concurrently. The WBIS collaboration has no flowchart equivalent to Fork. **Migration action**: Two options exist for trying to migrate orchestration's Fork into WBIS collaboration's programming model: 1. Give up concurrency and translate the fork into a set of sequential activities. 2. Split the orchestration into multiple collaborations and adapters and utilize the parallel execution environment provided by the WBIS for triggering events. For more information about this technique, see Appendix C, "Fork Join migration" on page 165.
Join	No equivalent	Merge multiple concurrent threads or execution branches back into a single execution process. BizTalk orchestration supports the execution of multiple branches concurrently. Hence, it provides a way to merge these branches back into a single execution branch. Merging can be done using the or condition or the and condition. The WBIS collaboration has no Fork capabilities and therefore no Join capabilities. **Migration action**: There is no straightforward way to migrate the orchestration's Join into a WBIS collaboration. See Appendix C, "Fork Join migration" on page 165 for a workaround to this problem.

BizTalk Orchestration	WBIS Collaboration	Description and migration action
End	End Success	Indicate the successful ending of a collaboration scenario or an orchestration. **Migration action**: Substitute the orchestration's End with the collaboration's End Success.
Abort	End Failure	Indicate that the collaboration scenario or orchestration ended with errors. The BizTalk server aborts the execution of the current transaction. It also terminates the execution of the workflow if no corrective actions are defined for the transaction. The WBIS stops the execution of the collaboration scenario and all its dependent children. The execution control is passed to the upper layer. No transaction compensation is executed. **Migration action**: Substitute the orchestration Abort with a collaboration End Failure.
Transaction	Transaction (scenario's property)	Enable the definition of a transaction around multiple actions. BizTalk orchestration's Transaction is a group of actions that either succeed or fail completely. WBIS collaboration defines Transaction as a collaboration property. Collaboration may or may not support transactions. WBIS lets you provide a compensation action in case of transaction failure and if rollback is needed. **Migration action**: Set the needed transaction level at the collaboration. Set compensation activities for each service call that requires compensation (using the Service Call Property tab). See 13.1, "Migrating transaction support" on page 146 for more details about transaction migration.
Implementation Shape	Service Call	Allow an action to communicate with an adapter or other collaboration. See the next section for details on migrating implementation shapes.
No BizTalk equivalent	Sub Diagram	Encapsulate a logical unit into a separate flow and associate it with the caller diagram.

12.4 Migrating implementation shapes

BizTalk orchestration supports four types of implementation shapes:

- COM components
- Scripting components
- Message queuing

- BizTalk messaging

Two options exist for migrating an implementation shape:

- Replace the implementation shape with a collaboration service call and a suitable adapter (for example, MQ adapter or COM adapter).
- Replace the implementation shape with Java logic inside a collaboration activity.

Replacing the implementation shape with a service call and adapter

WBIS best practices require the collaboration to interact with external components using a service call. The service call allows the collaboration to interact with an external application (using an adapter) or with other collaborations. Using a service call and an adapter to replace an implementation shape is somewhat cumbersome. It requires an additional independent process (the adapter), a set of configuration properties, and business objects that can be understood by the target adapter.

Replacing the implementation shape with Java logic

The WBIS Collaboration Designer allows you to add Java logic into the collaboration action. Using this technique, you can implement the required business logic directly in the orchestration without having to call external processes. This method is particularly valuable when only a simple processing task is needed.

12.4.1 Migrating a COM component implementation shape

The COM component implementation shape allows the orchestration to use a COM component (either local or remote) to execute the needed business logic. The shape allows you to both send and receive messages, to and from the COM component.

You can replace the COM component shape with the WBIS COM adapter or with a specific Java logic inside collaboration action. The preferred approach depends upon the issues discussed in this section.

The runtime environment is the first thing you must consider while migrating COM components. To execute a COM component without changing it, you must have operating system support for COM component execution (for example, Windows). You will probably need to rewrite your COM code if you are migrating to Linux, UNIX, or another environment that does not support the COM programming model.

You can use the hybrid approach as a workaround for rewriting your COM component from scratch. Using the hybrid approach (Figure 12-5), the WBIS

server runs over a non-Windows machine (for example, Linux or UNIX) while the WBIS COM adapter and your COM component run on a separate Windows-based host.

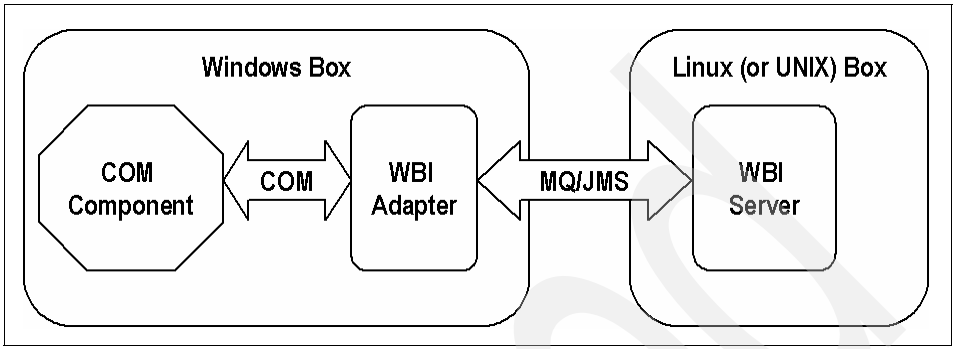

Figure 12-5 Hybrid solution for executing COM components

You can use more sophisticated semi-automatic porting techniques to transform your code into Java-based environments if you cannot use the hybrid solution and a Windows-based machine.

If you migrate your application into a runtime environment that does support COM execution, the preferred method will depend upon the type of logic required by the COM component. A simple, well-encapsulated logic might be better replaced by Java logic inside an action. The adapter approach is preferable if you need more complicated transaction-based communication.

Replacing the COM component shape with a WBIS COM adapter

A WBIS collaboration interacts with COM components using the WBIS COM adapter. You will need to create a suitable business object to carry the business data back and forth between the collaboration and the COM adapter and to control the process execution by the COM component.

You can use the Object Discovery Agent (ODA) to create the correct BOs for communicating with the COM adapter (Figure 12-6). After using the ODA and creating these BOs, you need to provide them to the service call.

You then need to programmatically map the triggering BOs into the newly created COM BO, in the action just before the service call.

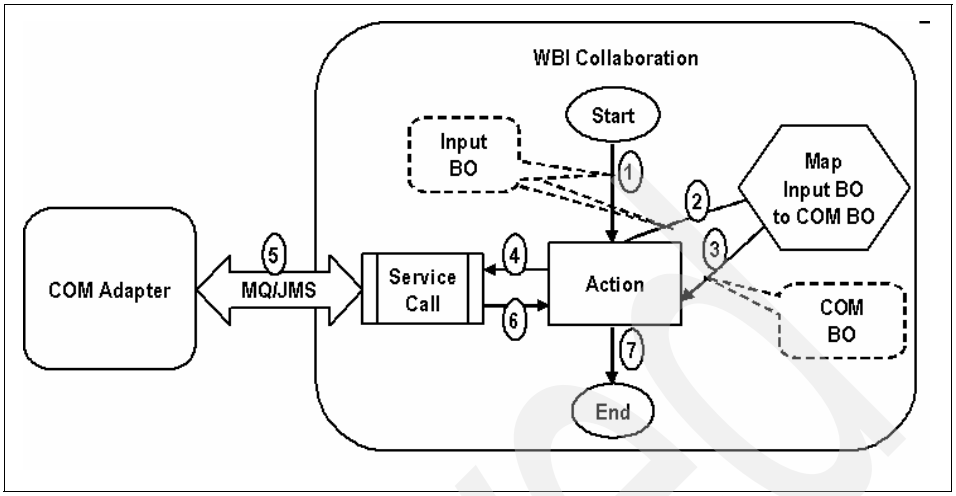

Figure 12-6 Transforming Input BO to COM BO and sending it to COM adapter

Most BizTalk COM components use the entire message (BizTalk document) as their input. This is due to the limited mapping capabilities of the BizTalk orchestration tool. You can use the DataHandler API (getStringFromBO() method) to serialize BOs back into their original string representation. (Recall that a single BizTalk message is translated into a set of BOs.) You can then send this string to the COM adapter inside the appropriate attribute of the COM BO.

At runtime, the service call sends the BOs to the COM adapter and returns the response back to the caller's collaboration.

Replacing the COM component shape with Java logic

Instead of using the service call with an adapter, you can replace the BizTalk COM implementation shape with an equivalent logic inside a collaboration action. Three options are available for this:

- Execute the original COM code using the bean scripting framework (BSF).
- Use a Java-COM bridge.
- Rewrite the COM code with Java.

Option 1: Execute the original COM code using the bean scripting framework (BSF)

If your COM component is written in a scripting language supported by BSF and you have the original code, you can use the BSF execution activity in the Activity Editor to execute your script from within the collaboration. This approach is similar to the functoid migration presented in 11.2, "BizTalk functoids and WBIS activities" on page 123.

Option 2: Use the Java-COM bridge

You can use a Java-COM bridge to wrap your COM component and then use the COM executer activity in the Activity Editor to execute your COM component from within a collaboration. Figure 12-7 shows a generic approach for executing a COM component from within a collaboration, using the Activity Editor.

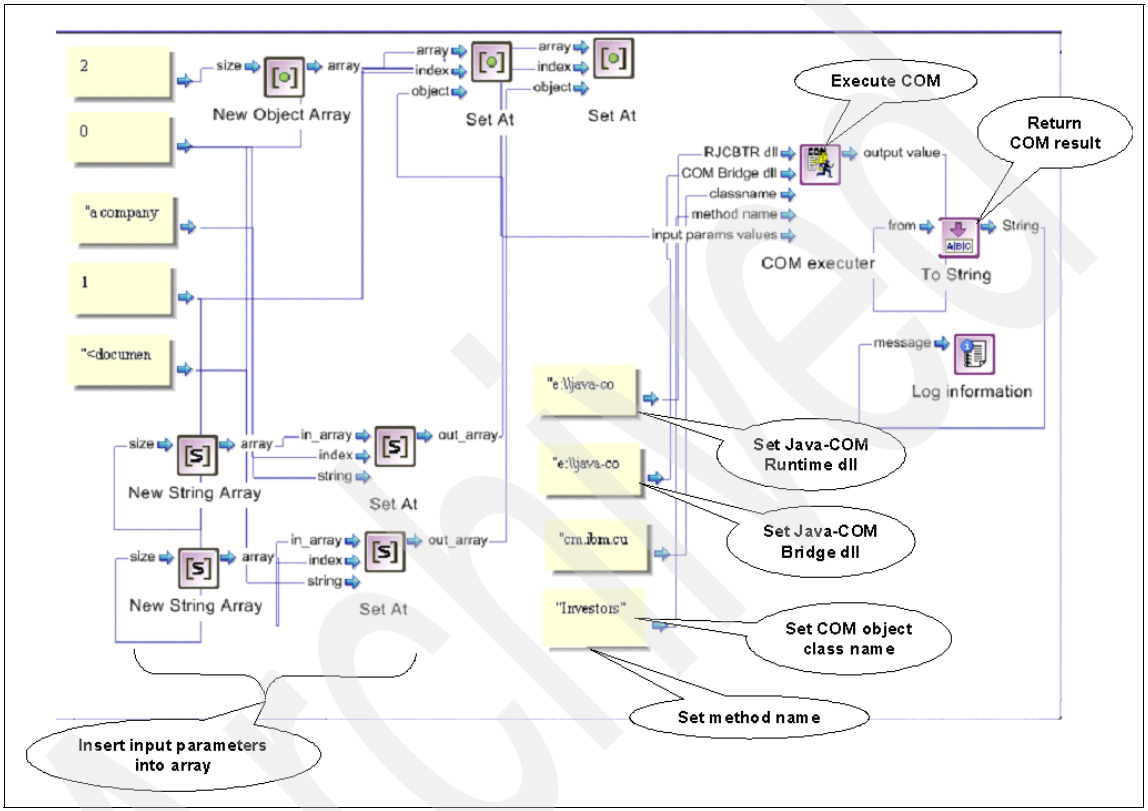

Figure 12-7 Executing COM component using the Activity Editor

You must first create an Object Array to hold all your input parameters. Each input parameter is a String Array with one element. You then need to set the following input parameters for the COM executer activity:

- Full path name of both the bridge dll and the RJCBRT runtime dll
- Fully qualified name of the class and the name of the method you want to invoke
- Input parameters object array

The COM executer activity executes your wrapped COM component method and returns its result to the output port. You can then assign the result to any output

parameter as needed. In the example shown in Figure 12-7, the result is first converted to a String using the To String activity, and then logged using the Log information activity.

Configuring WBIS for Java-COM Bridge support

This section describes the steps needed to configure WBIS for Java-COM Bridge support.

1. Register the COM component in the system running WBIS by executing the following command:

   regsrv32 *theCOMcomponent.dll*

2. There are several Java-COM Bridge tools that you can use to wrap your COM component. Some are commercial and some open source. For example, JACOB is an open source Java-COM Bridge. Another option is to work with the IBM Rational Java-COM Bridge (RJCB).

   To work with RJCB, you need to install the IBM Rational Development Tool for Java-COM Bridge. This is an Eclipse extension and requires JDK1.4 (or above), Microsoft Visual C++® 6.0 Service Pack 5 (or above), Eclipse 3.0, and EMF 2.0.

   For instructions on how to install the Development Tool for Java-COM Bridge, read the IBM Rational Java-COM Bridge documentation (see Appendix B, “Java-COM bridge and Windows Script Components” on page 163).

3. Create a new Java-COM Bridge project using the New Project wizard. Using this wizard you provide a project name and add a Java-COM Bridge. When defining the Java-COM Bridge you must provide the bridge name, the location of the COM component, and the Java package name, which will be later used to access the COM interface.

   For information on how to create a new Java-COM Bridge project, see Appendix B.

4. Build the project created in the previous step. If the build is successful, two files will be generated: a jar file holding the Java wrapper of the COM component and a dll holding the native wrapper of the COM component. If in the previous step, you named your bridge `myCOMbridge`, the generated files will be named `myCOMbridge.jar` and `myCOMbridge.dll`.

   For information on how to build a Java-COM Bridge project, see Appendix B.

5. Since the Java-COM Bridge package generated in the previous step is a third-party package, you need to import it to ICS. To do this, proceed as follows:

   a. Import your bridge to the ICS design time tools.

      i. Open the System Manager (click **Start** → **All Programs** → **IBM WebSphere Business Integration Express** → **Toolset Express** → **Administrative** → **System Manager.**)

      ii. Open the activity settings view (click **Window** → **Show View** → **Other** → **WebSphere Business Integration System Manager** → **Activity Settings**.)

      iii. Right-click **BuildBlock Libraries** and select the **Add Library** menu item. Select the jar files generated in step 2 and add them.

   b. Import your bridge to the ICS runtime.

      i. Add the bridge dlls (myCOMbridge.dll and RJCBRT.dll) to the PATH environment variable. myCOMbridge.dll is the native DLL generated in step 4. RJCBRT.dll is the Rational Java-COM Bridge Runtime library.

      ii. Add the bridge packages (myCOMbridge.jar and RJCBRT.jar) to the CLASSPATH environment variable. The recommended way to import third-party package libraries to ICS is to add the corresponding jar files to the JCLASSES variable in the script used to start the ICS server (for example, start_server.bat). For example, if the jar files to be imported are stored in directory c:\, add the following lines to the script used to start the ICS server:

      ```
      Set JAVA_COM_BRIDGE_JARS=c:\myCOMbridge.jar;c:\RJCBRT.jar
      JCLASSES=…existing jars ….;%JAVA_COM_BRIDGE_JARS%
      ```

      iii. Restart the ICS Server.

6. Once you successfully imported the Java-COM Bridge to ICS, you need to write the Java Client code that invokes it. Use the generic COM executer activity in the Activity Editor to do this. COM executer invokes a given Java interface of a given COM component and returns its corresponding result. Figure 12-7 on page 138 shows an example of how to use the COM executer component.

   Note: The COM executer activity is currently not part of the WBIS product.

Option 3: Migrating the original code to Java

You always have the option of porting the original COM code to Java. In this case, you need to understand the exact functionality of the COM and implement it. This method is suitable when the COM component performs simple, well-defined, and easy-to-develop functions.

This method is the only possible method if the targeted WBIS platform does not support the COM programming model. There are several efforts and initiatives for automatic porting from VBScript (and other programming languages) into Java. You can look into these if you need to port a significant amount of software into a non-Windows environment.

12.4.2 Migrating a Window Scripting Component implementation shape

The Window Scripting Component implementation shape (WSC) is very similar to the COM component. It just uses a different interface represented by a WSC file.

Although these interfaces have a lot in common, there is currently no support for WSC interfaces with the Java to COM Bridge. There is also no support for WSC interfaces with the WBIS COM adapter. Hence, you have two primary options while migrating WSC components into the WBIS environment:

- Extract the WSC interface from the COM component and use one of the migration techniques explained in 12.4.1, “Migrating a COM component implementation shape” on page 135.
- Manually migrate the WSC to Java and execute it inside the collaboration action.

12.4.3 Migrating a Message Queuing implementation shape

The Message Queuing implementation shape (Figure 12-8) allows BizTalk users to send messages to, and receive messages from, the Microsoft Message Queue (MSMQ) component within the orchestration.

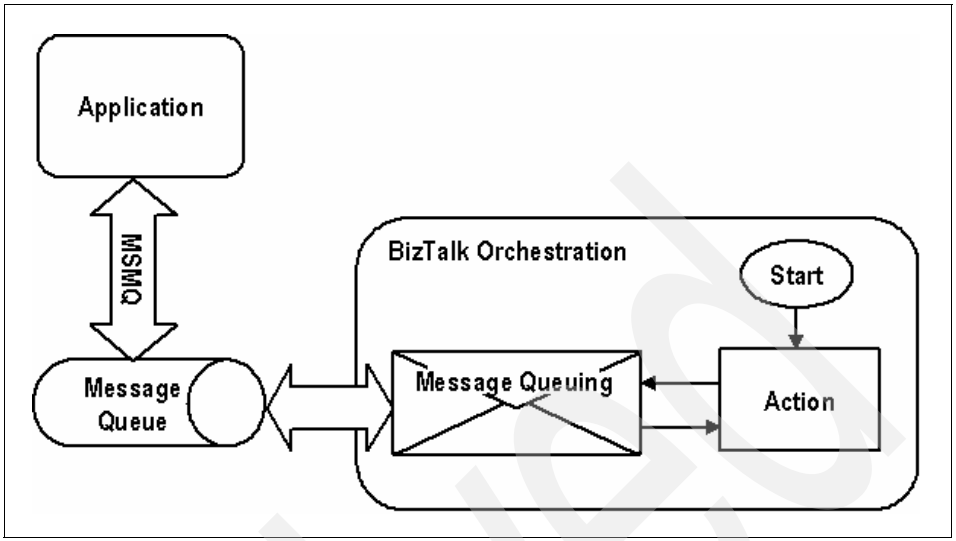

Figure 12-8 BizTalk MSMQ implementation shape

The orchestration uses the MSMQ implementation shape to interact with external applications using a message queue. The IBM equivalent to MSMQ is the IBM Message Queue (MQ) product. WBI Server Express provides the MQ adapter for talking to applications over a message queue. Hence, the straightforward MSMQ implementation shape migration strategy is to replace it with a service call and an MQ adapter (Figure 12-9).

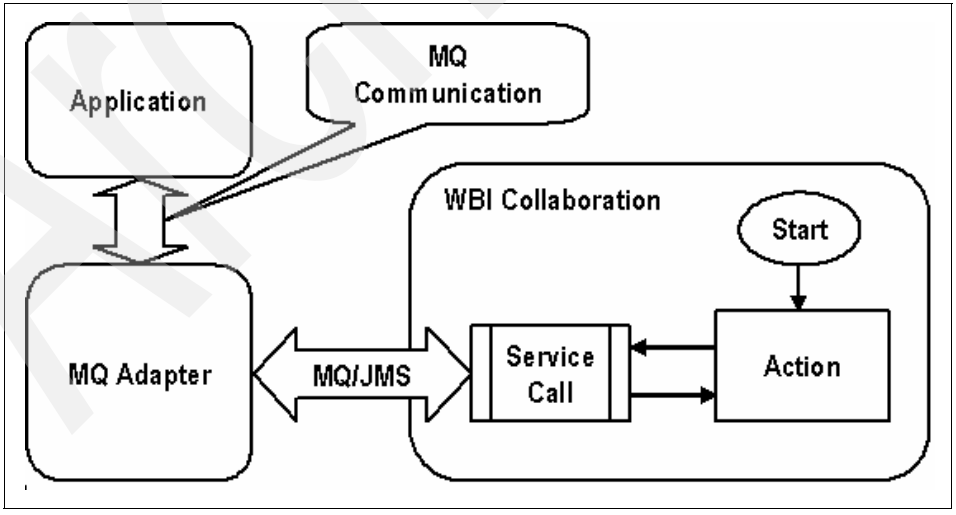

Figure 12-9 Collaboration with a service call connected to WBI MQ Adapter and to an application over MQ

This strategy requires that the existing communication channel to the application (based on MSMQ) is replaced by a new WebSphere MQ-based channel. This change requires changes in the application at the other side of the channel; these are changes that may be impossible to implement.

There are two approaches you can take in cases where the MSMQ channel cannot be replaced with an IBM MQ channel. The first one is to use an MSMQ adapter. The MSMQ adapter interacts with the WBIS in the same way the MQ adapter interacts, but it talks to the application over MSMQ (Figure 12-10).

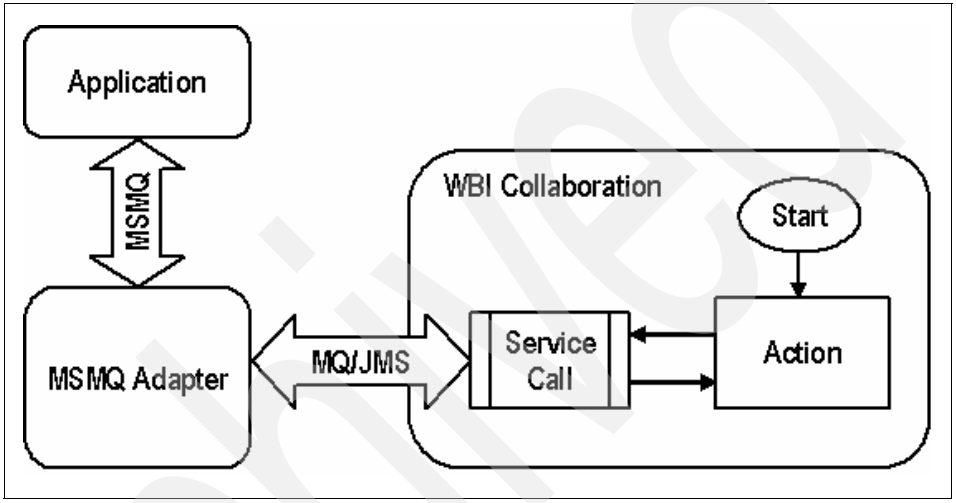

Figure 12-10 Collaboration with a service call connected to the new WBI MSMQ Adapter and to an application over MSMQ

The MSMQ adapter does not yet exist and must be developed to support such an approach.

A second option is to create (if it does not exist) a new COM component that implements the required MSMQ communication for interacting with your application (Figure 12-11). You can then use the WBIS COM adapter as explain in “Replacing the COM component shape with a WBIS COM adapter” on page 136 to interact with your newly created COM and with your application.

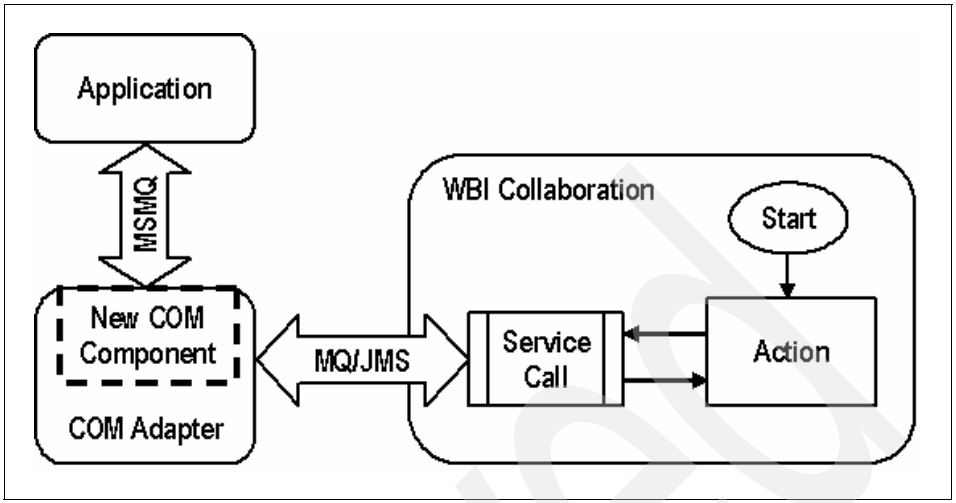

Figure 12-11 Develop a new COM and use the COM adapter

12.4.4 Migrating a BizTalk Messaging Implementation shape

BizTalk users use the BizTalk Messaging Implementation shape to send data to and read data from the BizTalk Messaging process within the orchestration. The BizTalk Messaging process supplies the routing services to the BizTalk orchestration process. WBIS has no such separation. All WBIS-related routing functionality is done implicitly by the WBIS runtime, based on the BO name and verb. WBIS implements a simple point-to-point interaction explicitly using service calls.

Sophisticated BizTalk application routing requirements can be implemented with a specific WBIS collaboration. WBIS supports collaboration groups, in which collaborations communicate with each other. The required BizTalk routing functionality must be implemented with a dedicated WBIS collaboration. This collaboration communicates with its peer collaboration in the group that implements the BizTalk orchestration functionality.

Hence, in most cases, the BizTalk messaging implementation shape will be replaced with a service call to a collaboration that implements the required routing functionality. See 10.4, “Advanced routing features” on page 116 for more information on routing functionality.

13

Migrating advanced features

This chapter discusses the migration of advanced features.

13.1 Migrating transaction support

Transactions pose a significant challenge in business integration systems because they are usually defined over a single component (database, queue, trading application), while business integration interacts simultaneously with multiple components. There are cases where a transaction that is already committed needs to be rolled back. This process is called *compensation*. Users must programmatically define the compensation steps for each transaction that might need to roll back after committing.

BizTalk and WBIS use different transaction models. BizTalk relies on utilities from the Windows operating systems (for example, Distributed Transaction Coordinator) for transaction control. These utilities allow BizTalk users to group several implementation shapes into a single transaction. BizTalk also provides the long-lived transaction feature which allows you to programmatically define compensation steps.

WBIS opens a new transaction for each service call and allows you to specify compensation actions for each service call in case of failure.

The following sections provide details on the architectural differences between the two systems, the properties involved in transaction configuration, and how to migrate BizTalk transactions into the WBIS environment.

13.2 BizTalk approach

BizTalk uses two types of transaction models: COM+ based transaction and XLANG Schedule based transaction. Each model supports four levels of transactions:

- **Not supported** - The transaction is not supported by this orchestration.
- **Supported** - The orchestration will participate in a transaction if its caller is participating in a transaction.
- **Required** - The orchestration will participate in a transaction if its caller is participating in a transaction. If the caller doesn't participate in a transaction, the orchestration will start a new one.
- **Required new** - The orchestration always starts a new transaction.

If the COM+ model is in use, the supported level is set at Begin shape (Figure 13-1) for the entire orchestration.

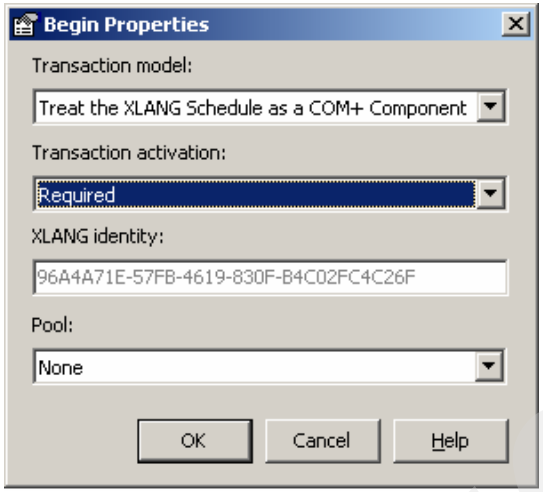

Figure 13-1 Using the COM+ transaction model

If you are using the XLANG Schedule transaction model, you need to set these properties (Figure 13-2) for each COM component implementation shape separately.

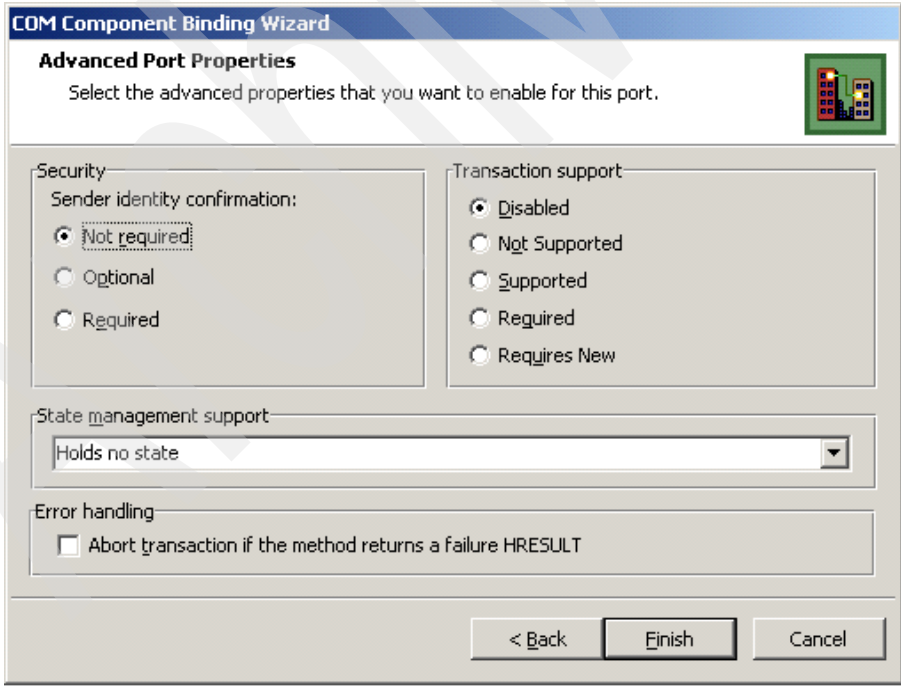

Figure 13-2 Setting transaction properties for each COM component in an XLANG transaction model

13.2.1 Data isolation

Each transaction supports four levels of data isolation:

- Uncommitted read - Reads data regardless of whether it was committed.
- Committed read - Reads only committed data.
- Repeatable read - Locks all the data it has read (but not the table itself).
- Serializable - Locks the table to prevent the insertion of updates and deletions.

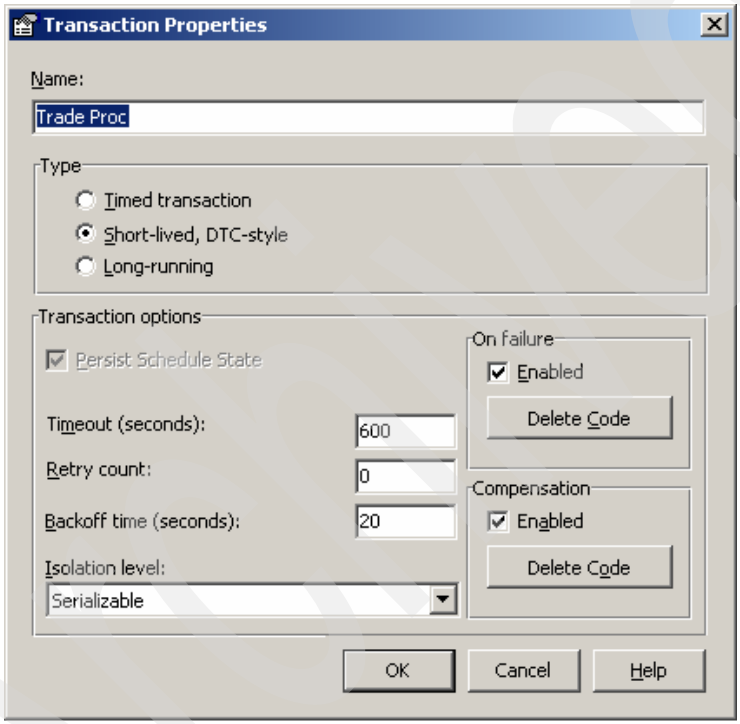

Figure 13-3 BizTalk transaction properties tab

BizTalk provides a separate tab in the Orchestration diagram for handling transaction failure (Figure 13-3). This tab is treated as if it were a regular stand-alone business process. Using this tab, you specify whatever action is needed after the transaction failure (for example, send e-mail to the administrator). When the specified business process is completed, the caller's business process continues its execution.

BizTalk defines a long-lived transaction, which lets you provide compensation actions when there is a need to roll back a transaction that has already been committed. This situation can happen when a business process is composed of

several stand-alone transactions. In this case, the process might fail after it has committed one transaction and before it has committed the other, so you need to provide compensation steps to roll back the transaction that was committed. A long-lived transaction allows you to do exactly that by supplying a separate compensation tab in which you can design your compensation steps.

13.3 WBIS approach

WBIS offers transaction support at the collaboration template level for all of collaboration's scenarios. A collaboration template's Minimum Transaction Level property determines whether the collaboration is transactional, and applies to all of its scenarios.

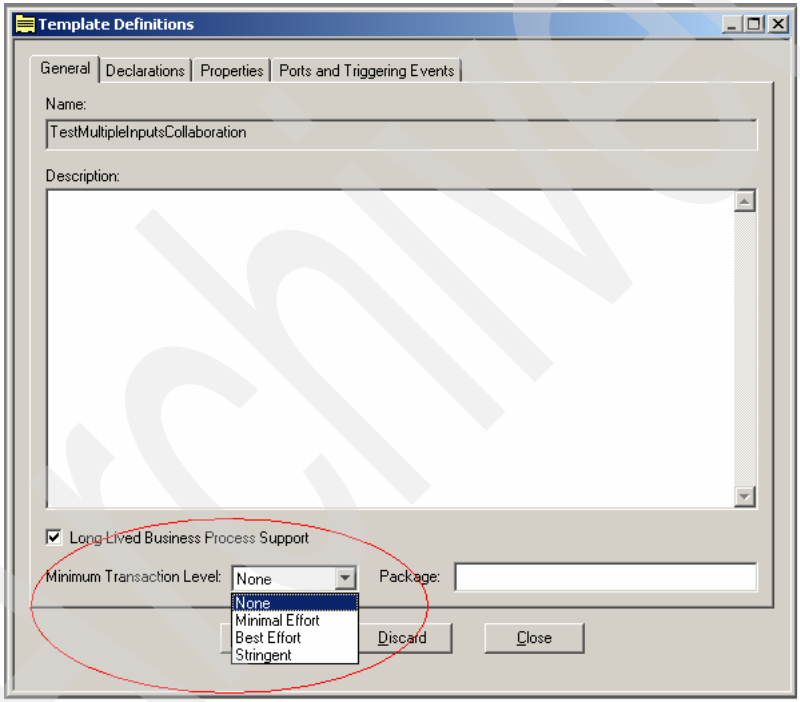

Figure 13-4 WBIS transaction properties

WBIS supports four transaction levels (Figure 13-4):

- **None** - Transactions are not supported by this collaboration.
- **Minimal Effort** - Supports transactions with rollback, but no data isolation.
- **Best Effort** - Supports transactions with rollback and with data isolation checking.

- **Stringent** - Supports transactions with rollback and with data locking during isolation check.

In a transactional collaboration, each scenario is a transactional scenario, whose start implicitly begins a transaction and whose successful completion implicitly commits the transaction. Hence, WBIS has no support for the BizTalk Supported and Required New options. WBIS supports only the None and Required options. The actual transaction level is determined by the lowest transaction level of the collaborations and adapters that participate in the transaction.

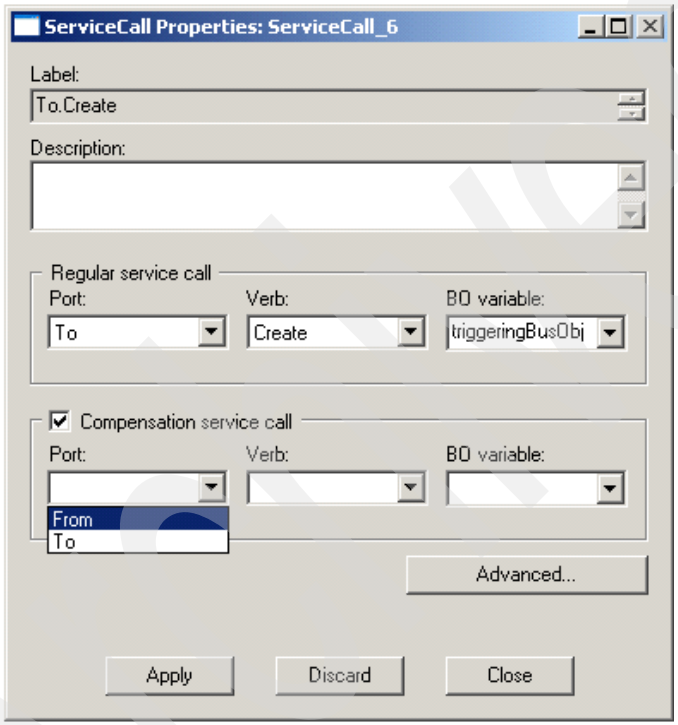

Figure 13-5 Specifying compensation actions for a service call

WBIS allows you to provide roll back action at the collaboration's service call level (Figure 13-5). During rollback, WBIS steps backward through the execution path and executes the compensation action for each service call.

13.3.1 Data isolation

The WBIS transactional scenario cannot lock data between execution steps, so data isolation cannot be guaranteed between these execution steps. WBIS supports the concept of isolation checking. Isolation checking allows you to

check for data consistency before changing or committing new data. Isolation checking is issued for Best Effort and Stringent transaction levels.

At the Best Effort level, WBIS checks for data changes between successive steps before applying the changes. The update operation results with an error if WBIS finds inconsistencies in the data. The Stringent level allows WBIS to lock the data between the data consistency check and the data update, thereby providing an even higher level of consistency. Few applications currently provide APIs for supporting such operations.

13.4 Migrating the transaction properties

There are many differences between the BizTalk transactional model and the WBIS transactional model. While WBIS opens a new transaction for every service call, BizTalk lets you group several implementation shapes in a single transaction. This grouping requires a set of properties for controlling the allocation and release of transaction resources.

WBIS allows you to set properties that span multiple transactions. By default, these properties are the scenario properties. (A scenario is usually composed of multiple transactions.) BizTalk uses the concept of long-lived transactions to allow the release of resources between transactions that span a long duration. There is no data consistency check in BizTalk terminology.

Table 13-1 summarizes the correlations between BizTalk and WBIS transaction support properties and suggests the required migration actions.

Table 13-1 Correlation between BizTalk and WBIS transaction support properties

Biz Talk transaction	**WBIS transaction**	**Description and migration action**
None	None	No support for transaction. **Migration action**: Set Transaction Level Support in the collaboration's General properties tab to None.
Supported, Required, Required New	Any transactional support level other then none	WBIS has no direct equivalent to these BizTalk properties. Any step in WBIS collaboration will start a new transaction. Any called process will be part of the caller transaction. The transaction level will be downgraded to the highest common level supported by all of its participants (lowest common denominator). **Migration action**: Set Transaction Level Support in the collaboration's General properties tab according to the required data isolation level (see the following rows for details).

Biz Talk transaction	WBIS transaction	Description and migration action
Read Uncommitted, Read Committed	Minimal Effort	BizTalk Read Uncommitted and Read Committed are not directly equivalent to WBIS Minimal Effort. Nevertheless, both assume that no changes to data occur between successive accesses. This might be the case if this process is the only one to access this data. **Migration action**: Set Transaction Level Support in the collaboration's General properties tab to Minimal Effort.
Repeatable Read	Best Effort	BizTalk Repeatable Read allows you to lock resources between successive transactions. While WBIS does not support this capability, you can still use the Best Effort level to ensure that no changes to data occur between successive accesses. **Migration action**: Set Transaction Level Support in the collaboration's General properties tab to Best Effort.
Serialized	Best Effort, Stringent	BizTalk Serializable allows you to lock all database resources while executing successive transactions. WBIS does not allow you to lock external resources between successive service calls. You can still use Best Effort or Stringent (if supported by the application) to ensure data consistency. **Migration action**: Set Transaction Level Support in the collaboration's General properties tab to Stringent (if the application API supports atomic test and set) or to Best Effort.

13.4.1 Migrating compensation steps

In BizTalk Orchestration, you use a dedicated business process tab to specify compensation steps. BizTalk sets failure actions as well as compensation steps in a separate process tab. These tabs behave like a WBIS sub-diagram and are actually executed in line with the main process.

In WBIS you specify compensation steps with each collaboration service call. You cannot specify other compensation actions inside your collaboration because WBIS runtime actually executes your process backward in case of a transaction failure. You can use a service call to other collaborations to execute your compensation steps if a simple rollback using the original port is not enough.

13.5 Correlation IDs

Business integration systems use Correlation IDs to execute several instances of the same business process simultaneously in an asynchronous fashion.

13.5.1 BizTalk correlation

BizTalk uses the instance ID as its unique ID for correlation purposes. BizTalk uses different correlation mechanisms for correlating messages over MSMQ or BizTalk Messaging.

BizTalk uses the orchestration instance ID (GUID that uniquely identifies a schedule instance) as the correlation ID when communicating over MSMQ. This ID is automatically added to every message sent or received by the orchestration. The ID is added to the message as a message label. MSMQ has support for message labeling. This message label must be added to every message sent to or received from that queue, both by the orchestration and the peer application. The peer application can extract the correlation ID from the queue using the MSMQ API.

BizTalk uses a local queue instance ID (GUID that uniquely identifies a local queue instance) as a correlation ID when communicating over BizTalk messaging. This local queue is dynamically created for each schedule instance. The queue's instance ID is communicated to the peer application as the correlation ID. The peer application must use this ID when submitting a response. The BizTalk runtime automatically removes this local queue after receiving the required response.

In summary, BizTalk uses an instance ID automatically generated by its runtime as the correlation ID. Peer applications must be able to find this ID and use it while sending the response.

13.5.2 WBIS correlation

WBIS uses a combination of BO attributes as its unique ID for correlation purposes. WBIS allows different correlation IDs for different service calls. You will need to define a template variable to hold the collaboration's correlation ID. You can initialize this variable at the start of the scenario or before calling the service call.

On receiving an asynchronous inbound service call, WBIS runtime correlates those defined BO attributes with the orchestration template variable. If WBIS finds a match, it continues with the scenario execution.

13.5.3 Migrating correlation IDs

If your BizTalk application uses correlation IDs, you need to manually define the unique attributes in your BO and set the template variable at the relevant scenarios to hold this ID. You then need to define the correlation ID at the relevant asynchronous inbound service call.

A

BizTalk functoids and WebSphere Business Integration activities

This appendix compares BizTalk functoids and WBIS activities. The following tables summarize the available BizTalk functoids and their WBIS counterparts. They are grouped by type, as follows:

- A-1, BizTalk string functoids
- A-2, BizTalk logical functoids
- A-3, BizTalk mathematical functoids
- A-4, BizTalk date/time functoids
- A-5, BizTalk conversion functoids
- A-6, BizTalk scientific functoids
- A-7, BizTalk cumulative functoids
- A-8, BizTalk database functoids
- A-9, BizTalk advanced functoids

Table A-1 BizTalk string functoids

BizTalk string functoids	WBIS activity component (General/string)	Description
Concatenate	Join (primitive)	Concatenates multiple strings into one.
String Extract	Substring by value	Extracts a string specified by the start and end positions of a super string.
String Find	No equivalent	Returns the position of the beginning of a string inside another string.
String Length	Text Length	Returns the length of a string.
String Left	Left String + Text Length	Returns a specific number of characters from the left side of a string. The WBIS Left String activity returns the left side of the string as a string, so Text Length activity has to be invoked to return the number of characters.
String Left Trim	Trim Left	Removes leading spaces from a string.
String Right	Right String + Text Length	Returns a specific number of characters from the right side of a string. The WBIS Right String activity returns the right side of the string as a string, so Text Length activity has to be invoked to return the number of characters.
String Right Trim	Trim Right	Removes trailing spaces from a string.
Lowercase	Lower Case	Returns the lowercase from a string.
Uppercase	Upper Case	Returns the uppercase from a string.

Table A-2 BizTalk logical functoids

BizTalk logical functoids	WBIS activity component (General/math)	Description
Greater Than	Greater Than	Returns **true** if the first parameter is greater than the second parameter.
Greater Than or Equal	Greater Than or Equal	Returns **true** if the first parameter is greater than or equal to the second parameter.
Less Than	Less Than	Returns **true** if the first parameter is less than the second parameter.

BizTalk logical functoids	WBIS activity component (General/math)	Description
Less Than or Equal to	Less Than or Equal to	Returns **true** if the first parameter is less than or equal to the second parameter.
Equal	Equal (Text Equal for Strings)	Returns **true** if the first parameter is equal to the second parameter.
Not Equal	Not Equal	Returns **true** if the first parameter isn't equal to the second parameter.
Logical String	No equivalent	Returns **true** if the expression is a string.
Logical Date	No equivalent	Returns **true** if the expression is a date.
Logical Numeric	Not a Number	Returns **true** if the expression is a number. WBIS returns true if the expression is not a number.
Logical OR	No equivalent	Returns the logical OR of the parameters.
Logical AND	No equivalent	Returns the logical AND of the parameters.
Logical Existence	No equivalent	Returns **true** if the input record of the field exists in the source document.

Table A-3 BizTalk mathematical functoids

BizTalk mathematical functoids	WBIS activity component (General/math)	Description
Absolute Value	Absolute Value	Returns the absolute value of a number.
Integer		Finds the integral portion of a real number.
Maximum Value	Maximum	Returns the maximum value of several numbers.
Minimum Value	Minimum	Returns the minimum value of several numbers.
Modulo	No equivalent	Returns the modulus after dividing the number by an integer.
Round	Round	Returns a number that is rounded to a specific number of decimal places. WBIS rounds to int and has no support for the specific number of decimal places.

BizTalk mathematical functoids	WBIS activity component (General/math)	Description
Square Root		Returns the square root of a number.
Addition	Plus	Returns the sum of two or more numbers. WBIS has no support for more than two inputs.
Subtraction	Minus	Subtracts one number from another.
Multiplication	Multiply	Returns the product of two or more numbers. WBIS has no support for more than two inputs.
Division	Divide	Divides one number by another.

Table A-4 BizTalk date/time functoids

BizTalk date/time functoids	WBIS activity component	Description
Add Day	Add Day	Returns the resulting date in the format YYYY-MM-DD after additional days have been added to the original date. WBIS receives the format as a parameter.
Date	Now	Returns a string representing the current date in the format YYYY-MM-DD. WBIS receives the format as a parameter.
Time	No direct equivalent	Returns a string representing the current time in the format hh:mm:ss.
Date and Time	Now	Returns a string representing the current date and time in the format YYYY-MM-DDTHH:MM:SS. WBIS receives the format as a parameter.

Table A-5 BizTalk conversion functoids

BizTalk conversion functoids	WBIS activity component	Description
ASCII from Character	No direct equivalent	Returns the ASCII value of a given character.
Character from ASCII	No direct equivalent	Returns the character for a given ASCII value.
Hexadecimal	No direct equivalent	Converts a decimal value to a hexadecimal value.
Octal	No direct equivalent	Converts a decimal value to an octal value.

Table A-6 BizTalk scientific functoids

BizTalk scientific functoids	WBIS activity component	Description
Sine	No direct equivalent	Returns the sine of a given number.
Cosine	No direct equivalent	Returns the cosine of a given number.
Tangent	No direct equivalent	Returns the tangent of a given number.
ArcTangent	No direct equivalent	Returns the arc tangent of a given number.
Natural Exponent Function	No direct equivalent	Returns the value of e to the given power.
Natural Logarithm	No direct equivalent	Returns the logarithm (base e) of a value.
Common Logarithm	No direct equivalent	Returns the logarithm (base 10) of a value.
Base Specified Logarithm	No direct equivalent	Returns the logarithm (base specific) of a value.
10 ^ X	No direct equivalent	Returns 10 to the X power.
X ^ Y	No direct equivalent	Returns X to the Y power.

Table A-7 BizTalk cumulative functoids

BizTalk cumulative functoids	WBIS activity component	Description
Cumulative Sum	No direct equivalent	Adds all values of the connected field by iterating over its parent record.
Cumulative Average	No direct equivalent	Calculates the average of all values for the connected field by iterating over its parent record.
Cumulative Minimum	No direct equivalent	Returns the minimum values of the nodes under its parent records.
Cumulative Maximum	No direct equivalent	Returns the maximum values of the nodes under its parent records.
Cumulative String	No direct equivalent	Returns the concatenated string of the string values for the concatenated field by iterating over its parent record.

Table A-8 BizTalk database functoids

BizTalk database functoids	WBIS activity component	Description
Database Lookup	No direct equivalent	Searches a database for a specific value and retrieves the record that contains the value. This functoid requires four parameters.
Value Extractor	No direct equivalent	Returns a value from a specific column in an ADO record set that has been retrieved by the Database Lookup functoid.
Error Return	No direct equivalent	Returns the error string, if any, returned by Open Database Connectivity (ODBC) when using the Database Lookup functoid.

Table A-9 BizTalk advanced functoids

BizTalk advanced functoids	WBIS activity component	Description
Scripting	No direct equivalent	Allows you to create customized VBScript or Jscript scripts to be executed at runtime.
Record Count	No direct equivalent	Returns a total count of records found in the instance.
Index	No direct equivalent	Returns the value of a record or a field at a specified index.
Iteration	No direct equivalent	Returns the iteration number (in a loop) of the source record.
Value Mapping	No direct equivalent	Returns the value of the second parameter if the value of the first parameter is true.
Value Mapping (Flattening)	No direct equivalent	Returns the value of the second parameter if the value of the first parameter is true, and flattens the source document hierarchy.
Looping	No direct equivalent	Creates multiple output records by iterating over each input record.

B

Java-COM bridge and Windows Script Components

This appendix provides an explanation of Java-COM bridge and the Windows Script components.

B.1 Java to COM bridge

Several vendors provide Java-COM bridges, and there is also an open source effort in this area (JACOB project - JAva COm Bridge). In our work, we use the IBM Rational Java-COM Bridge (RJCB), which supports bridging from Java components to COM and vice versa. The tooling for building RJCB, the Development Tool for Java-COM Bridge (DTJCB), integrates with the open source Eclipse IDE.

The RJCB technology uses the Java Native Interface (JNI) framework to bridge Java code and COM code. It allows users to execute COM components from Java. COM APIs are described in a special file called a type library. Type libraries can be stand-alone files (with a .tlb extension) or they can be embedded inside executable files (.exe, .dll, and .ocx). The RJCB technology reads a type library and generates the Java and C++ bridge code based on the API described

in that type library. COM supports two different method-call mechanisms: early-bound and late-bound.

- Early-bound calls know exactly which method is being called and the method's corresponding parameter types.
- Late-bound calls use a special super-interface called **IDispatch**.

IDispatch provides two methods: a lookup method that, given the name of a COM method, returns its corresponding method dispatch ID (dispid); and an invocation method that, given the dispid and an array of variants containing the call parameters, invokes the corresponding COM interface method. For performance reasons, the RJCB technology currently supports only COM APIs that provide early-bound interfaces. The RJCB-generated bridge code makes direct, interface-specific calls; it does not use the **IDispatch** super-interface. Using the Java-COM bridge technology is recommended when the COM functionality is complicated to implement or when it requires intensive development resources.

You can download the development tool for Java-COM Bridge from:
http://www.alphaworks.ibm.com/tech/dtjcb

You can find information about the JACOB project at:
http://danadler.com/jacob.

B.2 Window Script Component interface

Windows Script Components (WSC) is a feature of Windows 2000 and Microsoft IIS 5.0. A WSC object is basically a COM object made of scripting code. It isn't a compiled binary file like most COM objects, but rather an XML file (with a .wsc extension) that exposes a set of methods and properties, which are implemented through scripting code (VBScript or JScript).

The executable module that implements WSC objects (the WSC runtime engine) is a system-provided dll (scrobj.dll), recognized by COM-aware clients. The WSC runtime engine acts like a proxy and makes WSC methods and properties available to clients; it resolves any method calls to .wsc scripts. From a COM API point of view, all WSC components provide late-bound interfaces. For performance reasons, the RJCB technology currently does not support these types of interfaces; thus, WSC components are not supported by the IBM Rational Java-COM bridge.

C

Fork Join migration

WBIS collaboration has no flowchart equivalent to Fork. However, by splitting the orchestration into multiple collaborations and adapters, and utilizing the parallel execution environment provided by the WBIS for triggering events, a substitute for Fork can be achieved.

Figure C-1 summarizes the full solution for migrating a BizTalk business process composed of Fork and Join operations into WBIS.

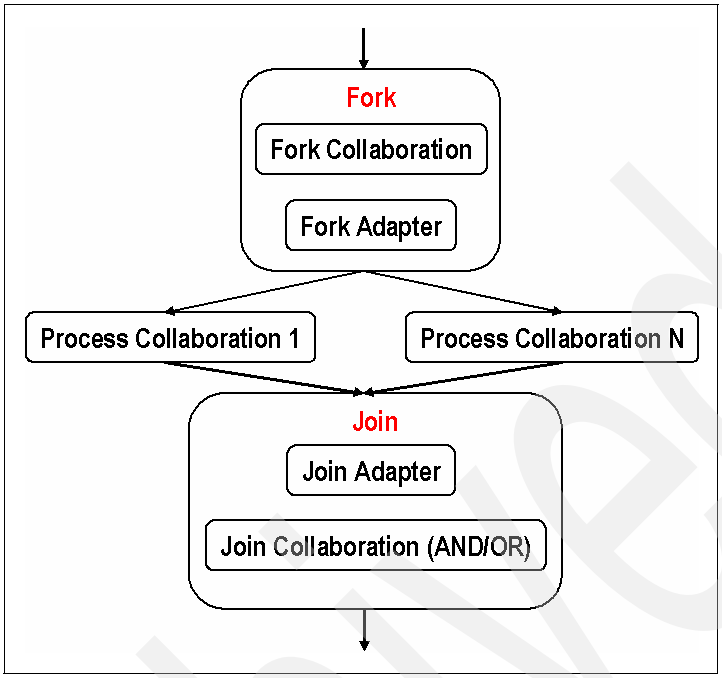

Figure C-1 Fork-Join migration process

The following section summarizes the basic components of this solution.

C.1 Business objects

In this section, we discuss the different business objects.

Input business object

The Input business object triggers the Fork process and contains the input data for the forked processes.

Result business object

The Result business object is the result of the processing done by the forked processes. We assume in this document that the Result business object is of the same type as the Input business object.

Wrapper business object

The Wrapper business object wraps the Input business object and the Result business object. It contains information that is needed for the execution of the Fork process (Figure C-2).

General | Attributes

	Pos	Name	Type	Key	Foreign	Requi red	Card	Maximu m Lengt	Default	App Spec Info	Comments
1	1	XMLDeclaration	String	☑	☐	☐		255		type=pi	
2	2	⊟ ROOT	Sample_Wrapper_CONTENT	☑	☐	☐	1			elem_name=sample_wrapper	
2.1	2.1	noNamespaceSchemaLocation	String	☐	☐	☐		255		attr_name=noNamespaceSchemaLocation;type=xsinoNSlocation	
2.2	2.2	id	String	☑	☐	☐		255		elem_name=id;type=pcdata	
2.3	2.3	process_id	String	☑	☐	☐		255		elem_name=process_id;type=pcdata	
2.4	2.4	⊞ input	Sample	☐	☐	☐	1			elem_name=input	
2.5	2.5	⊞ result	Sample	☐	☐	☐	N			elem_name=result	
2.6	2.6	ObjectEventId	String								
3	3	ObjectEventId	String								
4	4			☐	☐	☐		255			

Figure C-2 Wrapper business object

- The ID attribute is the unique identifier of the fork process.
- The process_id attribute is the unique identifier of the forked process.
- The input attribute is the input business object.
- The result attribute is an array of size NUM_OF_PROCESSES of business objects of the same type as the result business object. This attribute enables you to save the results of each forked process.

C.2 Collaboration templates

In this section, we discussed the collaboration templates.

C.2.1 ForkCollaboration

The ForkCollaboration template handles the forked process (Figure C-3).

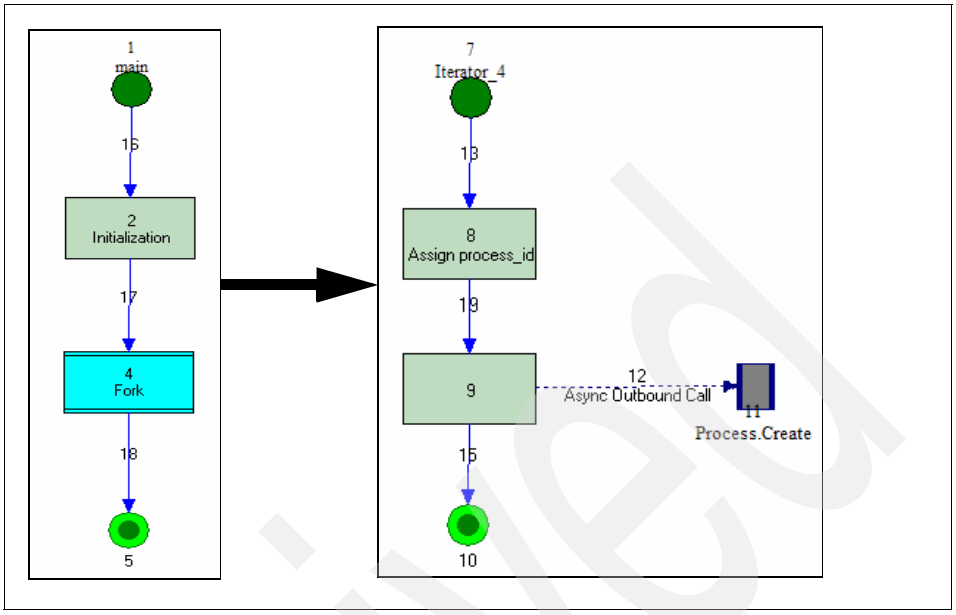

Figure C-3 Forked process

Properties

The following property is defined for the collaboration template:

1. **NUM_OF_PROCESSES** - The number of forked processes.

Ports

Two ports are defined for the collaboration template, as shown in Figure C-4.

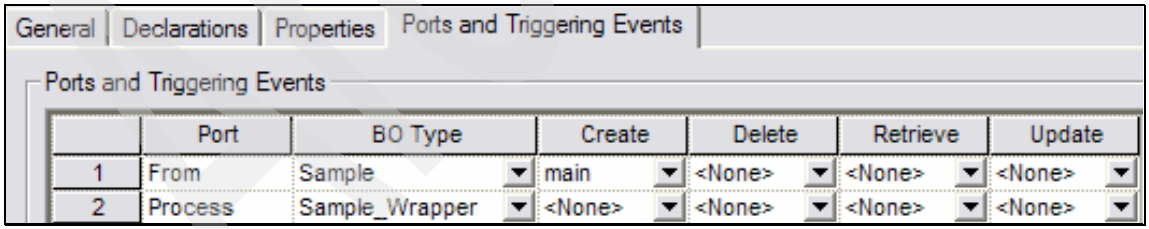

Figure C-4 Ports and Triggering Events

- **From** - The port connected to the originator of the data that will be delivered to the forked processes. The business object that is handled is the Input business object.
- **Process** - The port connected to the forked processes. The business object that is delivered to the forked processes is the Wrapper business object.

Workflow

The workflow of the collaboration template is as follows:

- **Initialization** - The Wrapper business object is initialized according to the number of processes, the Input business object, and the Result business object.
- **Fork** - for each process, the process_id attribute is set (starting from 1 to NUM_OF_PROCESSES) and an asynchronous service call is made to the Process port with the Wrapper business object.

C.2.2 ProcessCollaboration

The ProcessCollaboration template handles the logic of the forked process.

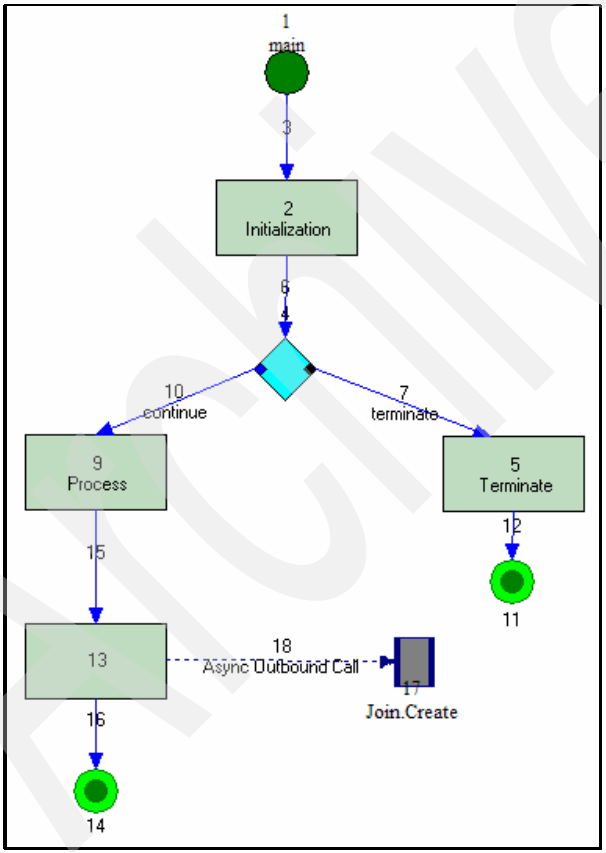

Figure 13-6 Forked process

Properties

The following property is defined for the collaboration template:

- **PROCESS_ID** - The unique identifier of the forked process.

Ports

Two ports are defined for the collaboration template (Figure C-5).

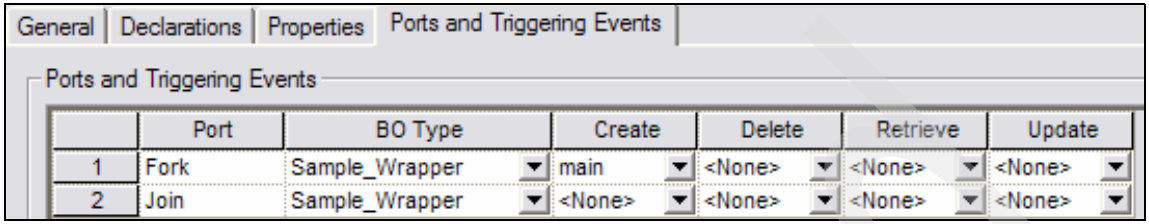

Figure C-5 Ports and Triggering Events

- **Fork** - The port bounded to the Fork process. The business object handled is the Wrapper business object.
- **Join** - The port connected to the Join process. The business object delivered to the Join process is the Wrapper business object.

Workflow

The workflow of the collaboration template is as follows:

1. **Initialization** - Extracts the process_id attribute from triggeringBusObj.
2. **Decision point** - Checks whether to continue the process. Continue this process only if the **process_id** attribute is the same as the **PROCESS_ID** property.
3. **Process** - The business process of the forked process. At the end of the process, copy the Wrapper business object that contains the result of the processing to the JoinBusObj.

   Use the following utility methods:

   BusObj getInputBO(BusObj pWrapperBO) - Returns the Input business object from triggeringBusObj. The returned business object contains the input for the forked process.

Example: C-1 Returned business object

```
public BusObj getInputBO(BusObj pWrapperBO){
  BusObj inputBO = null;
  try{
    inputBO = pWrapperBO.getBusObj("ROOT.input");
  }
  catch(Exception ex){
    System.err.println
      ("Exception occurred " + ex.getMessage());
```

```
  }
  return inputBO;
}
```

BusObj getWrapperBOWithResult(BusObj pWrapperBO, BusObj pResultBO) - Returns a Wrapper business object that contains the Result business object in the result array, at the index corresponding to the process that generated the result.

Example: C-2 Result business object

```
public BusObj getWrapperBOWithResult(BusObj pWrapperBO, BusObj pResultBO){
  BusObj wrapperBOWithResult = null;
  try{
    String processeID = pWrapperBO.get("ROOT.process_id").toString().trim();
    int processIndex = 0;
    try{
      processIndex = (Integer.parseInt(processID) - 1);
    }
    catch(NumberFormatException ex) {}

    BusObjArray result = pWrapperBO.getBusObjArray("ROOT.result");
    if((null != result) && (result.size() > processIndex)){
      result.setElementAt(processIndex, pResultBO);
    }
    wrapperBOWithResult = new BusObj(pWrapperBO.getType());
    wrapperBOWithResult.copy(pWrapperBO);
  }
  catch(Exception ex){
    System.err.println("Exception occurred " + ex.getMessage());
  }
  return wrapperBOWithResult;
}
```

4. **End** - Executes an asynchronous service call to the Join port with the JoinBusObj.

C.2.3 JoinCollaboration

The JoinCollaboration template handles the **Join** process (Figure C-6).

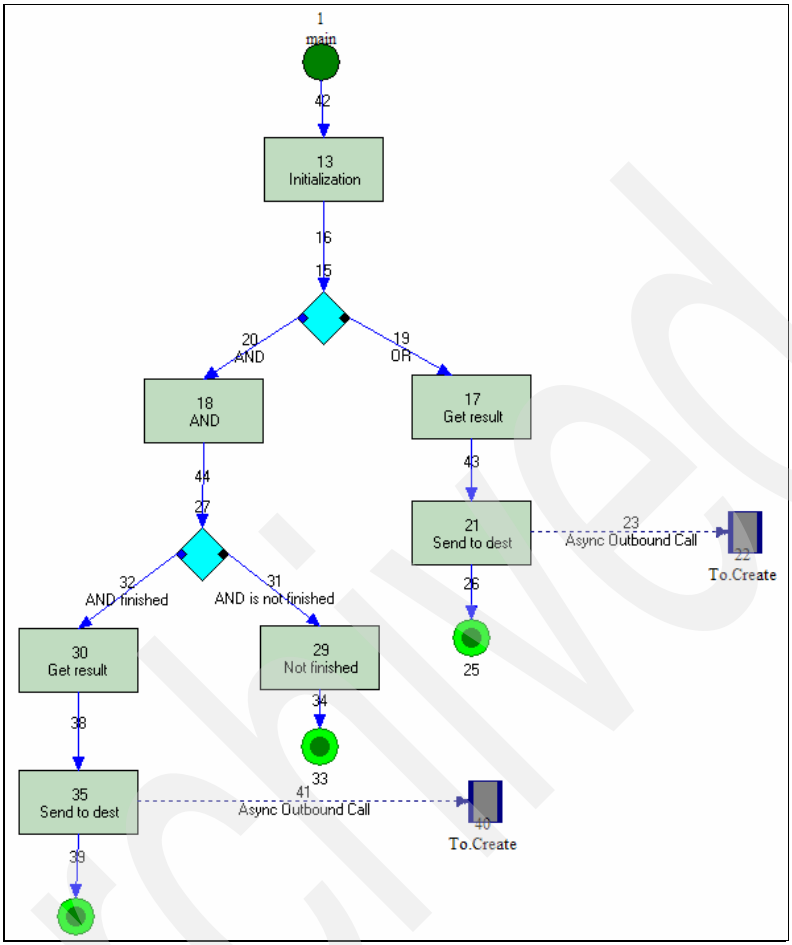

Figure C-6 Join process

Properties

The following property is defined for the collaboration template:

- **JOIN_OP** - the Join operation: **AND** or **OR**.

Ports

Two ports are defined for the collaboration template (Figure C-7).

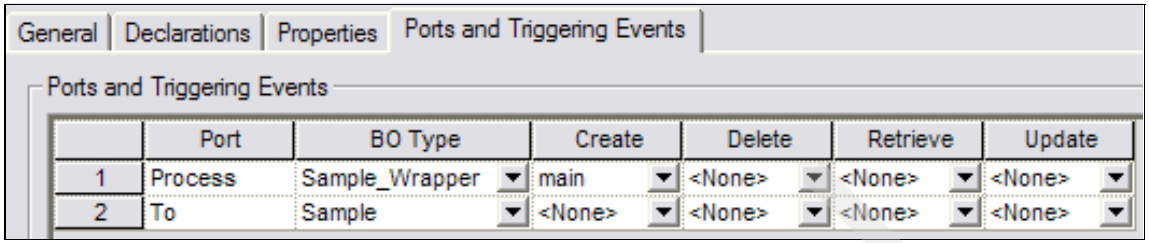

Figure C-7 Ports and Triggering events

- **Process** - The port connected to the forked processes. The business object handled is the Wrapper business object.
- **To** -The port connected to the destination. The business object delivered to the destination is the Result business object.

Workflow

The workflow of the collaboration template is implemented by using the **ForkAndJoinHandler** class. This class is the implementation of the interface shown in Example C-3.

Example: C-3 IForkAndJoinHandler interface

```
public Interface IForkAndJoinHandler{

  public BusObj getInputBO(BusObj pWrapperBO);

  public BusObj getWrapperBOWithResult(BusObj pWrapperBO, BusObj pResultBO);

  public void updateWrapperBOWithResult(String pJoinOP, BusObj pWrapperBO);

  public BusObj getResultBO(String pJoinOP, BusObj pWrapperBO);

  public boolean isAllProcessesFinished(String pJoinOP, BusObj pWrapperBO);

  public BusObj getResultBO(BusObj[] pResultBOs);
}
```

This class gives a default implementation for the interface methods:

- BusObj getInputBO(BusObj pWrapperBO) - Returns the Input business object from a Wrapper business object.
- BusObj getWrapperBOWithResult(BusObj pWrapperBO, BusObj pResultBO) - Returns a Wrapper business object that contains the Result business object in the result array, at the index corresponding to the process that generated the result.

- void updateWrapperBOWithResult(String pJoinOP, BusObj pWrapperBO)
  - For the OR operation, returns immediately with no additional actions.
  - For the AND operation, creates a new Wrapper business object for this fork process with empty results for all the forked processes (if it hasn't been created yet). Retrieves the Result business object from the given Wrapper business object and stores it in the Wrapper business object associated with this fork process.
- BusObj getResultBO(String pJoinOP, BusObj pWrapperBO)
  - For the OR operation, returns the Result business object that is stored in the result array of the given Wrapper business object at the index corresponding to the calling process.
  - For the AND operation, invokes the abstract method getResultBO(BusObj[] pResultBOs) on the Wrapper business object that is stored for this fork process and returns the returned business object.
- Boolean isAllProcessesFinished(String pJoinOP, BusObj pWrapperBO)
  - For the OR operation, returns true.
  - For the AND operation, returns true if all the forked processes have finished their processing; otherwise, returns false.
- abstract BusObj getResultBO(BusObj[] pResultBOs) - Needs to be implemented according to the required behavior for the AND operation. It should create a Result business object from all the Result business objects that were generated by the forked processes.

The workflow of the collaboration template is as follows:

1. **Initialization** - Calls the updateWrapperBOWithResult() method with the value of the JOIN_OP property and the triggeringBusObj.
2. **Decision point** - Checks whether the value of the JOIN_OP property is AND or OR:

   OR

   - Retrieves the Result business object by invoking the getResultBO() method with the value of the JOIN_OP property and the triggeringBusObj
   - Copies the returned Result business object to the ToBusObj
   - End - Executes an asynchronous service call to the To port with the ToBusObj

AND

– Decision point - Checks whether all the forked processes have finished their processing:

  • Finished

    * Retrieves the Result business object by invoking the getResultBO() method with the value of the JOIN_OP property and the triggeringBusObj.

    * Copies the returned Result business object to the ToBusObj.

    * End - executes an asynchronous service call to the To port with the ToBusObj.

  • Not Finished

    * End

C.3 Collaboration objects

This section discusses collaboration objects.

C.3.1 ForkCollaboration_N_Processes

This collaboration object is generated from the ForkCollaboration template (Figure C-8).

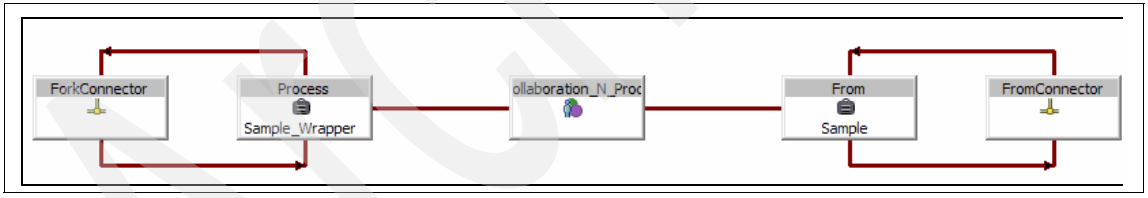

Figure C-8 Collaboration object

- The property NUM_OF_PROCESSES needs to be set to the number of the forked processes.
- The From port can be bound directly to the adapter that originates the flow or to another collaboration, if the data needs to be processed.
- The Process port cannot be bound directly to a collaboration object because this would cause the service call to automatically become synchronous. Therefore, this port is bound to a well-defined adapter, the ForkAdapter, which is explained in Appendix C.4.3, “ForkAdapter” on page 177.

C.3.2 ProcessCollaboration_1 - ProcessCollaboration_N

These collaboration objects are generated from the ProcessCollaboration template (Figure C-9).

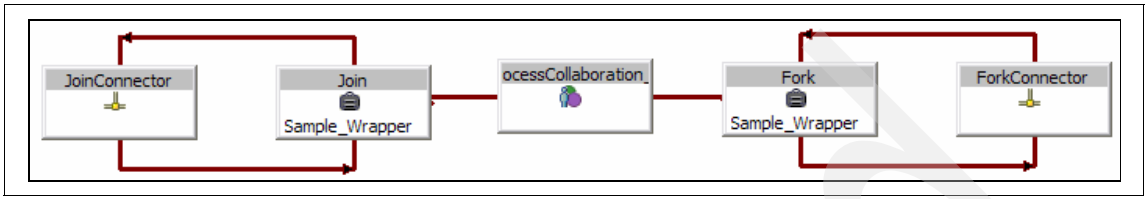

Figure C-9 Collaboration objects

- The property PROCESS_ID needs to be set for each collaboration object, from 1 to NUM_OF_PROCESSES, which was defined in the ForkCollaboration_N_Processes.
- The Fork port is bounded to the ForkAdapter. As explained previously, the adapter is necessary for the asynchronous execution.
- The Join port cannot be bound directly to the collaboration object that implements the join process because it can't be bound simultaneously to several collaboration objects. Therefore, this port is bound to a well-defined adapter, the JoinAdapter, which is covered in the next section.

13.5.4 JoinCollaboration_AND/JoinCollaboration_OR

These collaboration objects are generated from the JoinCollaboration template (Figure C-10).

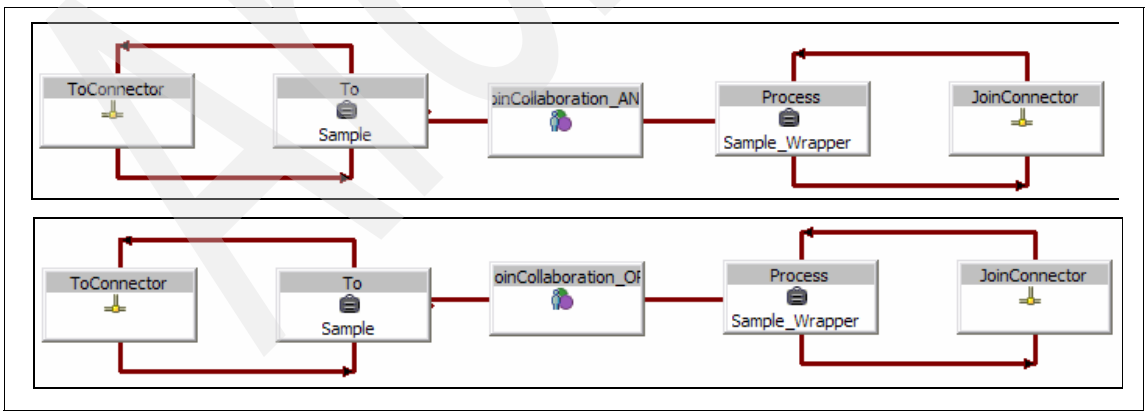

Figure C-10 Collaboration objects

- The property JOIN_OP needs to be set to AND or OR, depending on the merging condition.
- The Process port is bounded to JoinAdapter. As explained previously, the adapter is necessary for connecting several collaboration objects.
- The To port can be bounded directly to the destination adapter or to another collaboration, if the data continues to be processed.

C.4 Adapters

This section discusses adapters.

C.4.1 FromAdapter

The FromAdapter is the adapter that originates the flow. It supports the Input business object.

C.4.2 ToAdapter

The ToAdapter is the destination adapter. It supports the Result business object.

C.4.3 ForkAdapter

The ForkAdapter is a lightweight adapter that acts as a proxy. It sends all the messages it receives from the ICS back to the ICS. This adapter is necessary for an asynchronous execution of the forked processes. It supports the Wrapper business object.

C.4.4 JoinAdapter

The JoinAdapter works in the same way as the ForkAdapter. It sends all the messages it receives from the ICS back to the ICS. This adapter is necessary for binding the forked processes to the Join process. It supports the Wrapper business object.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

Other publications

This publication is relevant as further information source:

- BizTalk Server 2002 Design and Implementation, by Xin Chen

Online resources

These Web sites and URLs are also relevant as further information sources:

- WebSphere Business Integration Server

  http://www-306.ibm.com/software/integration/wbiserver/
- Microsoft BizTalk

  http://www.microsoft.com/biztalk/
- Mainsoft

  http://www.mainsoft.com
- Stryon

  http://www.stryon.com
- The Mono project

  http://www.go-mono.com
- Alphaworks

  http://www.alphaworks.ibm.com/tech/dtjcb
- Danadler

  http://danadler.com/jacob

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

O

P

R

S

T

U

V

W

X

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Microsoft BizTalk to WebSphere Business Integration Server Express Migration

Best migration process to WebSphere Busines Integration

Understand the similarities and differences between BizTalk and WBIS

Learn the techniques to migrate to WBIS

The purpose of this IBM Redbook is to help the reader understand what is involved in migrating a BizTalk application into the WebSphere Business Integration (WBIS) environment. It deals specifically with migrating a BizTalk 2002 application into the WebSphere Business Integration Server Express (WBI SE) environment.

It begins with a general discussion of business integration platforms, and migration drivers and issues relevant in the small to medium business environment. Several migration options are described and some basic migration techniques are presented.

A comprehensive description of the BizTalk and WBIS platforms is provided, highlighting the similarities and differences between the two environments. A migration path and a set of best practices are proposed to migrate a BizTalk application into the WebSphere Business Integration environment. Finally, the detailed steps for migrating the functional and architectural components of BizTalk to the WBIS environment are presented.

Readers are assumed to be familiar with the concepts of business integration and distributed applications, including multi-tiered architecture, synchronous and asynchronous messaging, transformations, content-based routing, and workflows. This redbook also assumes knowledge of cross-platform open standards, such as eXtensible Markup Language (XML) syntax and methods, Web Services Description Language (WSDL), and Business Process Execution Language (BPEL).

This redbook is intended for both internal IBM employees and external IT personnel who intend to migrate a BizTalk-based business integration application into the WebSphere Business Integration environment.

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

SG24-7142-00 ISBN 0738496170