IBM

# Flexible Self-Service Application Patterns Using WebSphere and Process Choreography on z/OS

- Build solutions using WebSphere BI Server for z/OS V5.1
- Explore self-service applications and the Decomposition pattern
- Follow the sample scenarios

Franck Injey
Mohammed Ajab
Pedro Duarte Meira
Andrew Gardner
Bart Verboven
Frances Williams
Rica Weller

Redbooks

ibm.com/redbooks

International Technical Support Organization

# Flexible Self-Service Application Patterns Using WebSphere and Process Choreography on z/OS

December 2005



SG24-6356-00

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



**First Edition (December 2005)**

This edition applies to the following products for use with z/OS V1R5, program number 5694-A01:

- WebSphere Business Integration Server Foundation for z/OS V5.1, program number 5655-L85
- CICS Transaction Server V2R2, program number 5697-E93
- CICS Transaction Gateway V5.1, program number 5724-D12
- DB2 Universal Database for z/OS V8R1, program number 5625-DB2
- IMS Transaction Manager V8R1, program number 5655-C56
- IMS Connect V2.1 program number 5655-K52
- WebSphere MQ for z/OS V5R3 Modification Level 1, program number 5655-F10

# 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:

AIX®
Cloudscape™
CICS Connection®
CICS®
CICSPlex®
Distributed Relational Database Architecture™
Domino®
DB2 Universal Database™
DB2®
DRDA®
Encina®
Enterprise Systems Architecture/390®
ESCON®
@server®
@server®
FICON®
HiperSockets™
IBM®
IMS™
Lotus®
MVS™
OS/2®
OS/390®
OS/400®
Parallel Sysplex®
PR/SM™
Redbooks (logo) ™
Redbooks™
RACF®
RMF™
S/360™
S/370™
S/390®
Tivoli®
VisualAge®
VTAM®
WebSphere®
z/Architecture™
z/OS®
zSeries®

The following terms are trademarks of other companies:

EmbeddedJava, Enterprise JavaBeans, EJB, Java, Java Naming and Directory Interface, JavaBeans, JavaOS, JavaScript, JavaServer, JavaServer Pages, JDBC, JDK, JSP, JVM, J2EE, Solaris, Sun, Sun Microsystems, 100% Pure Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

ActiveX, Microsoft, PowerPoint, Visio, Windows NT, 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 role of an IT architect has become more challenging by several orders of magnitude in recent years. Today an IT architect is responsible for working with lines of business staff and capturing business needs. This person must also translate those requirements into an IT solution that satisfies the current business goals and exhibits several additional characteristics. To rise to these challenges, the IT architect needs to develop increased competencies in several areas including the business domain, architecture methodologies, and current software, hardware, networking, and storage technologies.

To improve this process, the IT architect needs to identify and leverage existing assets. This IBM® Redbook shows you how to leverage one key set of assets, the IBM Patterns for e-business. It explains how you can use the IBM Patterns for e-business in combination with key architectural elements to deliver on demand and autonomic architectures that deliver best-in-class qualities of service.

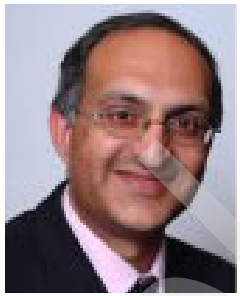
This book is designed as a resource for IT architects whose primary area of expertise is enterprise or mainframe systems. In the book, you learn about business scenarios that an IT architect is likely to come across. You also learn how to implement these solutions using best practice design guidelines.

Part 1 provides an overview of the IBM Patterns for e-business. It focuses on describing Self-Service business patterns and how they are transformed into an implemented infrastructure. Part 2 introduces concepts, design considerations, technologies, and architectural elements that are key in helping the IT architect in the task of designing an architecture based on the IBM @server zSeries® platform. Finally Part 3 provides a technical overview about the implementation of Self-Service business patterns on the zSeries platform.

# 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 (ITSO), Poughkeepsie Center.

**Frank Injey** is a project leader at the ITSO, Poughkeepsie Center. He has 30 years of experience working on S/390® and @server zSeries hardware and system performance. Franck is a Senior I/T Architect from France.



**Mohammed Ajab** is a consulting IT specialist for the IBM Software business in the United Kingdom (U.K.). He has over 21 years of experience in the area of application and systems development, application development management, architecting complex systems and consulting. His areas of expertise are enterprise application integration and cross-brand software advocacy. Mohammed holds both a Bachelor of Science and a Master of Science degree in Computer Science from the University of London.

**Pedro Duarte Meira** is a senior certified engineer and an IT architect. He is currently the vice president of Capital IT, SGPS. He writes extensively and teaches on enterprise architecture, business process modeling, workflow, enterprise content management, and related topics. Pedro holds several formal IBM professional certifications including certifications for DB2®, WebSphere® MQ, WebSphere MQ Workflow, IBM DB2 Content Manager, and IBM DB2 Content Manager OnDemand. He is a member of Institute of Electrical & Electronic Engineers (IEEE) and Association for Computing Machinery (ACM).

**Andrew Gardner** is a support specialist at the IBM Support Center in Australia. He has 16 years of experience in both mainframe and distributed systems. His areas of expertise include z/OS®, CICS® Transaction Server, WebSphere Application Server and WebSphere MQ. Andrew holds a degree in science from the University of Sydney.

**Bart Verboven** is an IBM Certified IT Specialist, working for IBM Global Services in Belgium. After a brief academic career in applied mathematics at the universities in Leuven, Belgium, and Bielefeld, Germany, he moved to an IT support role in the insurance and medical sector. In 1997, Bart joined IBM, and he now has over 13 years experience in supporting several database and transaction systems on diverse platforms. Bart has long been devoted to CICS, as well as the WebSphere for z/OS portfolio. He has also contributed to the redbook *Revealed! Architecting Web Access to CICS,* SG24-5466.

**Frances Williams** is a senior consultant with the IBM @server zSeries e-business Services in the United States. She has over 20 years of experience in application development from design to implementation. She provides consulting and educational services to zSeries customers. Her areas of expertise include the WebSphere suite of products, DB2, and application design.

**Rica Weller** is a Project Manager at the International Technical Support Organization (ITSO), working out of New Zealand and the U.S. She was a Systems Engineer for S/390® for two years and as a Senior Consultant for IBM WebSphere® Business Integration on z/OS in the Competence Center with IBM Germany for three years. She also taught classes, presented at several conferences, and coauthored several Redbooks™ about WebSphere on z/OS and zScholar Basics. Rica holds a degree in Business Administration from TU Dresden, Germany, and a Master's Degree in Business from Massey University, New Zealand.

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

Rich Conway
Tamas Vilaghy
ITSO, Poughkeepsie Center

Martin Keen
Carla Sadtler
ITSO, 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 Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

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

# Part 1

# Patterns for e-business

IBM Patterns for e-business have resulted in the creation of a significant volume of assets. Part 1 provides an overview of these assets, which are the Business patterns, Integration patterns, Composite patterns, and custom designs.

In Chapter 1, “Overview of Patterns for e-business” on page 3, we explain the purpose of patterns and assets and how to categorize them. We introduce you to the asset model and describe the process of translating patterns into business solutions.

Then in Chapter 2, “Business pattern assets” on page 19, we describe the use of Business patterns, including the Self-Service, Collaboration, Information Aggregation, and Extended Enterprise business patterns. We associate these patterns with the Application and Runtime patterns and their variations.

Access Integration patterns and Application Integration patterns support access to multiple applications, databases, and services. We describe these Integration patterns in more detail in Chapter 3, “Integration pattern assets” on page 37. In this chapter, we also explain how they are used in conjunction with Business patterns to create solutions that support complex business requirements.

In Chapter 4, “Composite patterns and custom design assets” on page 53, we describe in more detail Composite patterns, which are a combination of Business and Integration patterns. We especially describe these patterns for such business scenarios involving Electronic Commerce, e-Marketplace, Portal, and Account Access. We also mention custom design assets.

In Chapter 5, “Selected patterns for e-business on zSeries” on page 65, we introduce the Application and Runtime patterns that we use to highlight the strengths of the zSeries platform in Part 2, “Methodology, design, and WBISF on z/OS” on page 81, and in Part 3, “Preparing WBISF run time and Workload Manager on z/OS” on page 137. We describe some high function runtime nodes that are specific to the zSeries and provide the product mappings for the runtime patterns.

1

# Overview of Patterns for e-business

IBM's work on Patterns for e-business has resulted in the creation of a significant volume of assets. These assets are Business patterns, Integration patterns, Composite patterns, and custom designs.

The objective of this chapter is to provide an overview of these assets based on the most current material available at the time of writing. We explain the purpose of patterns and assets and how you categorize them. We introduce you to the asset model and describe the process of translating patterns into business solutions considering variations.

## 1.1 Purpose of patterns and assets

The role of the IT architect is to evaluate business problems and build solutions to solve them. The architect begins by gathering input on the problem, the outline of the desired solution, and any special considerations or requirements that need to be factored into that solution. Using this input, the architect then designs the solution, which includes computer applications that address business problems by supplying necessary business functions.

To improve the process over time, we need to capture and reuse the experience of the IT architects in such a way that future engagements can be made simpler and faster. We do this by capturing knowledge gained from each engagement and using it to build a repository of assets. IT architects can then build future solutions based on these proven assets. This reuse saves time, money, and effort, as well as helps to ensure delivery of a solid, properly architected solution.

IBM Patterns for e-business help facilitate this reuse of assets. Their purpose is to capture and publish e-business (On Demand Business) artifacts that have been used, tested, and proven to be successful. The information captured by them is assumed to fit the majority of situations. IBM Patterns for e-business are further augmented with guidelines and related links for better use.

## 1.2 Pattern categories and the layered asset model

The IBM Patterns for e-business assets are organized into four main pattern categories:

- Business pattern

  This pattern describes the interaction between users, business entity (or systems), and data.

- Integration pattern

  This pattern describes whether the emphasis is on integration at the user interface level or integration at a middleware level between processes, people, applications, and information. Integration patterns tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern.

- Composite pattern

  This pattern represents commonly occurring combinations of Business patterns and Integration patterns to deliver the business functionality required.

- Custom design

  A custom design is similar to Composite patterns, except that the patterns have been implemented to a lesser extent.

Each pattern category contains its own patterns, called *pattern assets*, as shown in Table 1-1.

*Table 1-1 Pattern categories and assets*

| Pattern categories | Pattern assets |
|---|---|
| Business patterns | Self-Service<br>Collaboration<br>Information Aggregation<br>Extended Enterprise |
| Integration patterns | Access Integration<br>Application Integration (including Data and Process Integration) |
| Composite patterns | Electronic Commerce<br>e-Marketplace<br>Portal<br>Account Access |
| Custom design | For example integrating the Self-Service and Collaboration patterns using WebSphere and Domino®, or integrating WebSphere Application Server with SAP R/3 |

These assets are used as a high-level guide during the process of translating business requirements for a solution into the IT solution that will service those business requirements.

The Patterns for e-business approach enables architects to implement successful solutions for On Demand Business through the reuse of components and solution elements from proven successful experiences. The Patterns for e-business approach is based on a set of layered assets that can be exploited by any existing development methodology. These layered assets are structured in a way that each level of detail builds on the last, as illustrated in Figure 1-1.

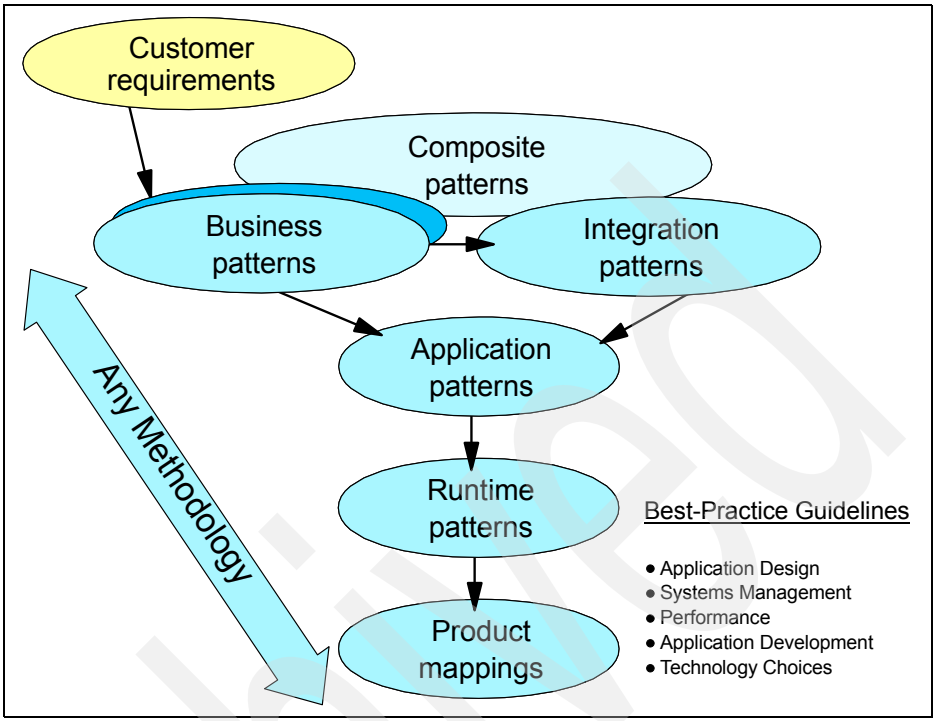



*Figure 1-1 The Patterns for e-business layered asset model*

At the highest layer are Business patterns, which describe the entities involved in the solution for On Demand Business. Composite patterns appear in the hierarchy above the Business patterns. However, Composite patterns are made up of a number of individual Business patterns and at least one Integration pattern. The Patterns for e-business layered asset model includes these additional elements:

- Application pattern

  This pattern describes the conceptual layout of application components and data within a Business pattern or Integration pattern interact.

- Runtime pattern

  This type of pattern defines the logical middleware structure that supports an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.

- Product mapping

  Product mapping entails using a selection of products (proven and tested software implementations) to provide the logical functions defined in the Runtime pattern.

- Design guidelines

  Design guidelines represent best practice guidance for working with the products selected and using these products to build applications, specifically guidelines for design, development, deployment, and management of e-business applications.

The Patterns Web site provides more information and an easy way to navigate through the layered Patterns assets to determine the most appropriate assets for a particular engagement. You can find this site on the Web at:

http://www.ibm.com/developerWorks/patterns/

## 1.3 Translating patterns into business solutions

When faced with the challenge of designing a solution for a business problem, you need a high-level view of the goals that you are trying to achieve. Describe the proposed business scenario, and match each element to an appropriate IBM Pattern for e-business. IBM Patterns for e-business are methodology agnostic, so you can use them to guide any architecture definition process.

To apply the layered asset model to design an architecture for a business problem, you use the following general process:

1. Identify the Pattern category.
   a. Choose the Business patterns that are required to provide the business function.
   b. Choose the Integration pattern that determines whether the emphasis is on integration at the user interface level (that is, portal), at the middleware level, or at both levels.
2. Select the Application pattern by selecting the major logical components and the interaction styles between them.
3. Select the Runtime pattern by selecting the logical middleware structure necessary to support the Application pattern selected in step 2. Decide where these logical functions will be placed in the overall network structure.
4. Select the products that will provide the functions specified in the Runtime pattern.
5. Apply the design guidelines to implement custom logical functions.

The following sections describe the process in more detail.

### 1.3.1 Selecting a pattern category

Suppose an insurance company wants to reduce the amount of time and money spent on call centers that handle customer inquiries. By allowing customers to view their policy information and request changes online, the company can cut back significantly on the resources spent on handling queries by phone. The objective allows policy holders to view their policy information stored in existing databases. This example is used throughout this chapter to explain patterns and their translation process better.

First you must choose the pattern category. You may find that the total solution requires multiple Business and Integration patterns, or that it fits into a Composite pattern or custom design. A Business pattern describes the relationship between the users, the business organizations or applications, and the data to be accessed. Table 1-2 lists the primary Business patterns and provides examples of them.

*Table 1-2 The primary Business patterns*

| Business patterns | Description | Examples |
|---|---|---|
| Self-Service (User-to-Business) | Applications that provide users with direct access to business applications and data via the Web | Simple Web site applications |
| Information Aggregation (User-to-Data) | Applications that allow users to extract useful information from large volumes of data, text, images, etc. | Business Intelligence, Knowledge Management, Web crawlers |
| Collaboration (User-to-User) | Applications in which the Internet supports collaborative work between users | E-mail, community, chat, videoconferencing, etc. |
| Extended Enterprise (Business-to-Business) | Applications that link two or more business processes across separate enterprises | Electronic data interchange (EDI), supply chain management, and so on |

It would be convenient if all problems fit nicely into these four slots, but reality is usually more complicated. The patterns assume that most problems, when broken down into their basic components, fit more than one of these patterns. Patterns for e-business provide Integration patterns if a problem requires multiple Business patterns.

Integration patterns allow you to tie together multiple Business patterns to solve a business problem. Table 1-3 outlines the Integration patterns.

*Table 1-3 Integration patterns*

| Integration patterns | Description | Examples |
|---|---|---|
| Access Integration | Integration of a number of services through a common entry point | Portals |
| Application Integration | Integration of multiple applications and data sources without the user directly invoking them | Message brokers, workflow managers, data propagators, data federation engines |

The Access Integration pattern maps to user integration. This pattern refers to the integration of a process, which is the functional flow of processing between the applications, or data, that is the information used by applications.

These Business and Integration patterns can be combined to implement installation-specific business solutions. We call this a *custom design*. We illustrate the use of a custom design to address a business problem through an iconic representation as shown in Figure 1-2.

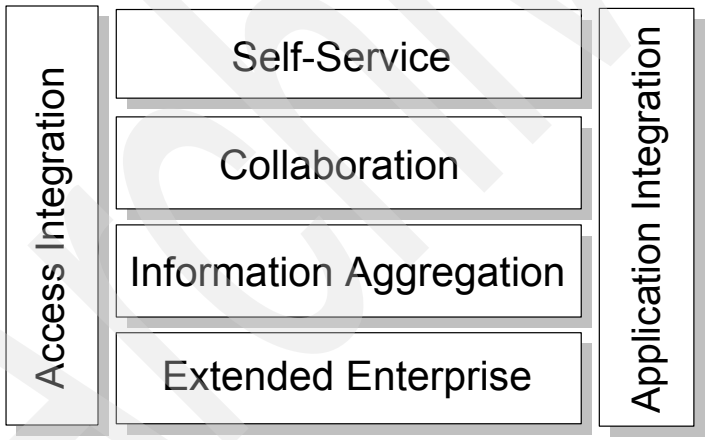



*Figure 1-2 Patterns that represent a custom design*

If any of the Business or Integration patterns are not used in a custom design, we can show the unused patterns as lighter blocks than those that are used. For example, Figure 1-3 shows a custom design that does not apply a Collaboration business pattern nor an Extended Enterprise business pattern for a specific business problem.


*Figure 1-3 Custom design without Collaboration or Extended Enterprise business patterns*

Common uses of Business and Integration patterns are identified and formalized into composite patterns which are shown in Table 1-4. A custom design may also be a Composite pattern if it recurs many times across domains with similar business problems. For example, the iconic view of a custom design in Figure 1-3 can also describe a Sell-Side Hub composite pattern.

*Table 1-4 Composite patterns*

| Composite pattern | Description | Examples |
|---|---|---|
| Electronic Commerce | Wholesale and retail via the Internet | Online warehouse, online bookstore |
| Portal | Users access aggregated applications and information from multiple sources in a uniform, seamless, and personalized way | ► Enterprise intranet portal: self-service functions such as payroll, benefits, and travel expenses<br>► Collaboration providers: e-mail or instant messaging |

| Composite pattern | Description | Examples |
|---|---|---|
| Account Access | Customers access their account information anytime they want | Online brokerage trading applications; telephone company account manager functions; bank, credit card, and insurance online applications |
| Trading Exchange | Buyers and sellers trade goods and services on a public site. | Any commodity market:<br>► Buyer's side: Interaction between buyer's procurement system and commerce functions of e-Marketplace<br>► Seller's side: Interaction between procurement functions of the e-Marketplace and its suppliers |
| Sell-Side Hub (Supplier) | The seller owns the e-Marketplace and uses it as a vehicle to sell goods and services on the Web | Car seller's Web site |
| Buy-Side Hub (Purchaser) | The buyer owns the e-Marketplace and uses it as a vehicle to leverage the buying or procurement budget in soliciting the best deals for goods and services from prospective sellers across the Web. | WorldWide retail exchange |

The makeup of these patterns is variable in that there will be basic patterns present for each type, but the Composite pattern can be extended easily to meet additional criteria. For more information about Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos[1].

### 1.3.2 Selecting Application patterns

After you identify the Business pattern, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the *Application pattern*. A Business pattern can have multiple Application patterns. An Application pattern may have logical components that describe a presentation tier for interacting with users, an application tier, and a back-end application tier.

[1] Adams, Jonathan; Galambos, George; Koushik, Srinivas; and Vasudeva, Guru. *Patterns for e-business: A Strategy for Reuse*. MC Press, October 2001. ISBN 1-931182-02-7

Application patterns break the application down into the most basic conceptual components, identifying the goal of the application. In our insurance example, the application falls into the Self-Service business pattern, and the goal is to build a simple application that allows users to access back-end information. The Self-Service::Directly Integrated Single Channel application pattern, shown in Figure 1-4, fulfills this requirement.

The Self-Service business pattern should be used in situations in which users need direct access to business applications and data.

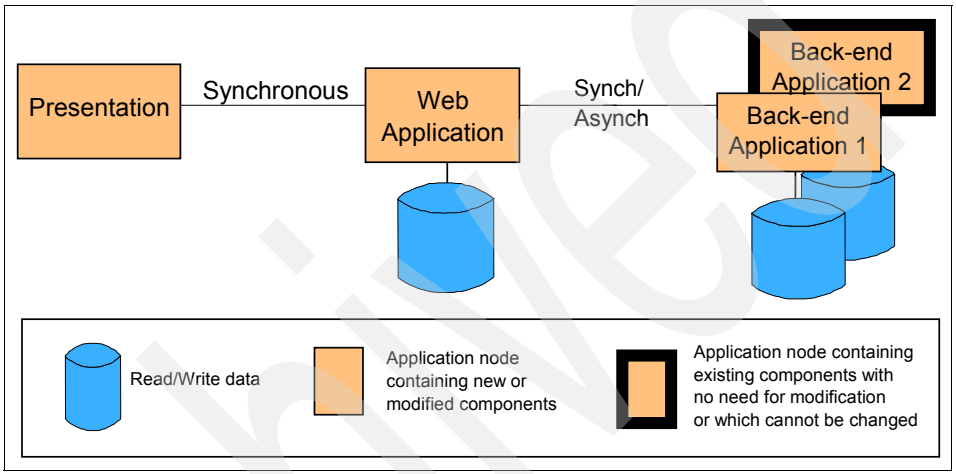



*Figure 1-4 Self-Service::Directly Integrated Single Channel*

This Application pattern consists of a presentation tier that handles the request/response to the user. The application tier represents the component that handles access to the back-end applications and data. The multiple application boxes on the right represent the back-end applications that contain the business data. The type of communication is specified as synchronous (one request/one response, then the next request/response) or asynchronous (multiple requests and responses intermixed).

Let's assume a more complex situation in which the automobile policies and the homeowner policies are kept in two separate and dissimilar databases. The user request would need data from multiple, disparate back-end systems. The request needs to be broken down into multiple requests (decompose the request) to be sent to the two different back-end databases. Then the information that is sent back from the requests needs to be gathered and put into the form of a response (recompose). The Application pattern in Figure 1-4 would no longer be sufficient.

Instead, we could use the Self-Service::Decomposition Application pattern shown in Figure 1-5. This Application pattern extends the idea of the application tier that accesses the back-end data by adding decomposition and recomposition capabilities.

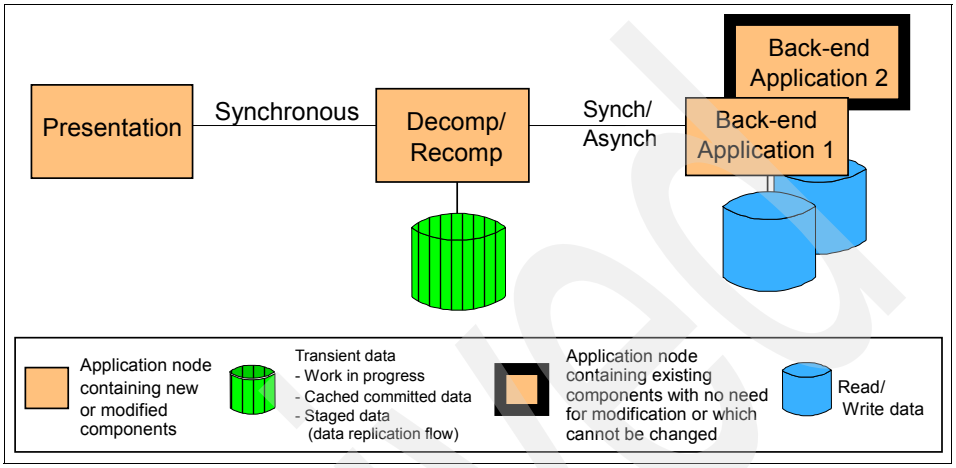



*Figure 1-5 Self-Service::Decomposition*

The concept of variations in the IBM Patterns for e-business is used to describe possible alternatives for Application patterns. This is due the fact that the architect has to consider several different design considerations which impact Application patterns, such as:

- Styles of interaction between applications

  Interaction may occur directly or via memory, via a call or message, serially or in parallel, via human interaction, or using combinations of these methods.

- Degree of functional richness provided by the interaction, transformation, or mediation service

  Some examples include Broker, Router, and Process Choreographer.

- Location of the interaction, transformation, or mediation service

  The service could be privately hosted by an external third-party. It could be private between two organizations, or it could be shared between multiple organizations.

- Approach to creating or accessing data/information resources

  The approach may involve direct access to source data-sets, access to new data-sets following aggregation, or use of an operational data store.

### 1.3.3 Identifying Runtime patterns

The Application pattern can be further refined with more explicit functions to be performed. Each function is associated with a runtime node. In reality these functions, or nodes, can exist on separate physical machines or can coexist on the same machine. In the Runtime pattern, this is not relevant. The focus is on the logical nodes required and their placement in the overall network structure.

As an example, let's assume that our client has determined that their solution fits into the Self-Service business pattern and that the Directly Integrated Single Channel pattern is the most descriptive of the situation. The next step is to determine the Runtime pattern that is most appropriate for their situation.

The client knows that they have users on the Internet accessing their business data, and therefore, require a measure of security. Security can be implemented at various layers of the application. However, the first line of defense is almost always one or more firewalls that define who and what can cross the physical network boundaries into their company network. They also need to determine the functional nodes required to implement the application and security measures. The Runtime pattern shown in Figure 1-6 is one option for the client.

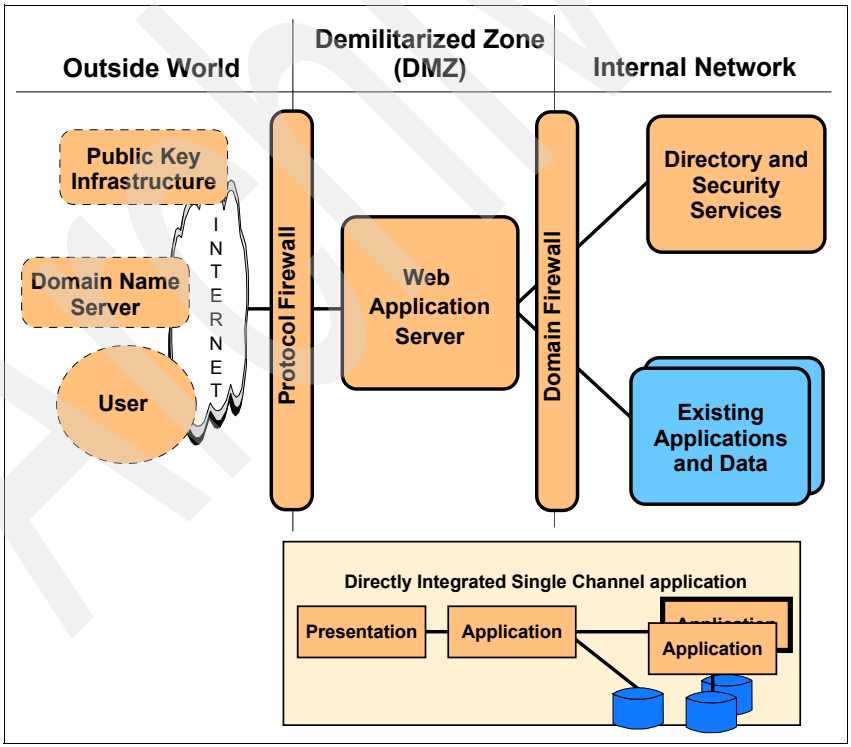



*Figure 1-6 Directly Integrated Single Channel application pattern::Runtime pattern*

By overlaying the Application pattern on the Runtime pattern, you can see the roles that each functional node will fulfill in the application. The presentation and application tiers will be implemented with a Web application server, which combines the functions of an HTTP server and an application server. It handles both static and dynamic Web pages.

Application security is handled by the Web application server through the use of a common central directory and security services node.

A characteristic that makes this Runtime pattern different from others is the placement of the Web application server between the two firewalls. The Runtime pattern shown in Figure 1-7 is a variation. It splits the Web application server into two functional nodes by separating the HTTP server function from the application server. The HTTP server (Web server redirector) serves static Web pages and redirects other requests to the application server. It moves the application server function behind the second firewall, adding more security and complexity.

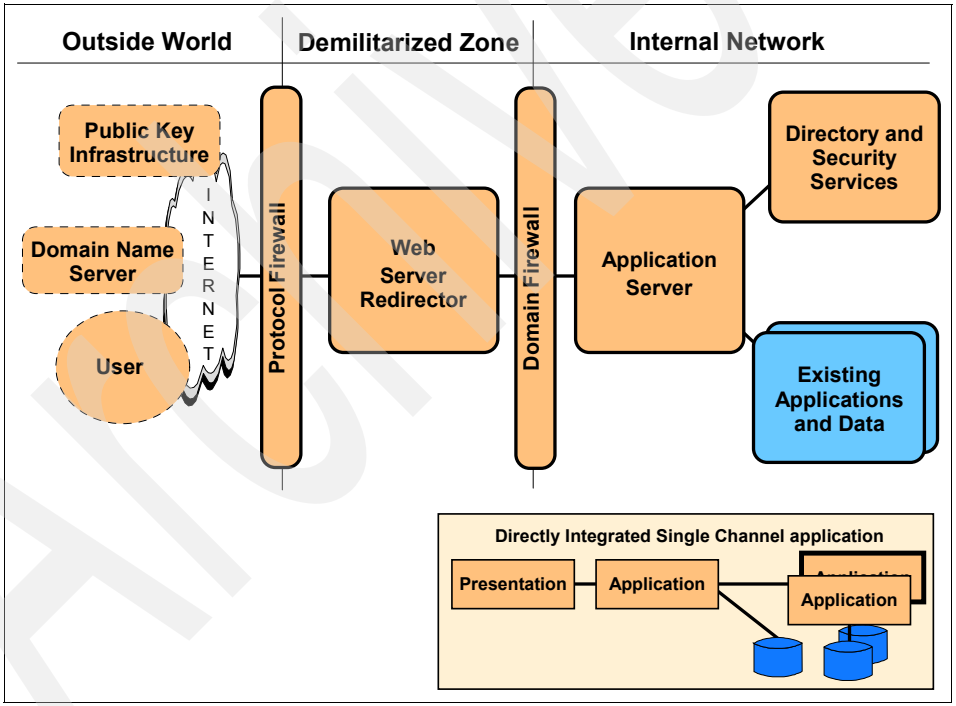



*Figure 1-7 Directly Integrated Single Channel application pattern::Runtime pattern: Variation 1*

These are just two examples of the Runtime patterns that are available. Each Application pattern can have one or more Runtime patterns. These can be modified to suit the customer's needs. For example, the customer may want to add a load-balancing function and multiple application servers.

The concept of variations in the IBM Patterns for e-business is used to describe possible alternatives for Runtime patterns. This is due the fact that the architect has to take into account a number of different design considerations, which impact Runtime patterns, such as:

- Choice of placement for functions/nodes within the overall network structure

  The nodes can be placed on the internal network, within the DMZ of the internal network, or outside the internal network.

- Non-functional requirements

  These may include high availability or high performance.

### 1.3.4 Mapping products

The next step in designing the solution (defining the network structure for the application) is to correlate real products with one or more runtime nodes. The Patterns Web site shows each Runtime pattern with products that have been tested in that capacity. The Product mappings are oriented toward a particular platform, though it is more likely that the customer will have a variety of platforms involved in the network. In this case, it is simply a matter of mix and match.

For example, the runtime variation in Figure 1-7 could be implemented using the product set illustrated in Figure 1-8.

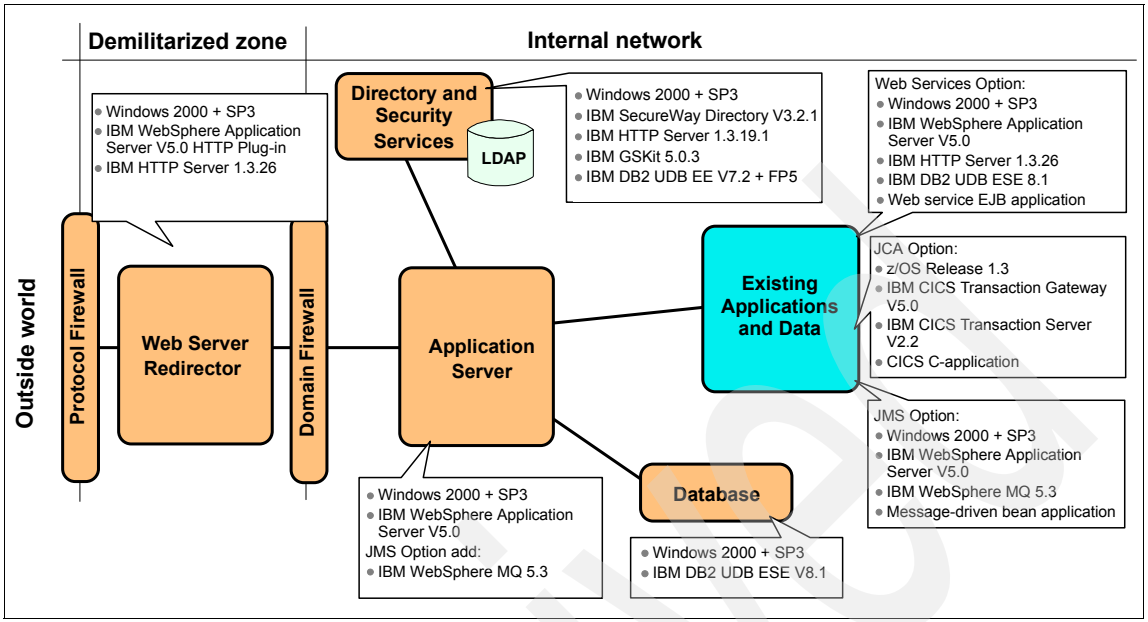



*Figure 1-8 Directly Integrated Single Channel application pattern: Microsoft® Windows® 2000 product mapping*

## 1.3.5 Reviewing guidelines and related links

The Application patterns, Runtime patterns, and product mappings are intended to guide you in defining the application requirements and the network layout. The application development has not been addressed yet. The Patterns Web site (see the end of 1.2, “Pattern categories and the layered asset model” on page 4) provides guidelines for each Application pattern, including techniques for developing, implementing, and managing the application, based on:

- Design guidelines that instruct you on tips and techniques for designing the applications
- Development guidelines that take you through the process of building the application, from the requirements phase through the testing and rollout phases
- System management guidelines that address the day-to-day operational concerns, including security, backup and recovery, application management, and so forth
- Performance guidelines that give information about how to improve the application and system performance

## 1.4 Summary

IBM Patterns for e-business are a collection of proven architectures. This repository of assets can be used by companies to facilitate the development of Web-based applications. They help an organization understand and analyze complex business problems. They break them down into smaller, more manageable functions that can then be implemented.

The layers of patterns, along with their associated links and guidelines, allow the IT architect to start with a problem and a vision for the solution, and then find a pattern that fits that vision. Then, by drilling down using the patterns process, the architect can further define the additional functional pieces that the application needs to succeed. Finally, the application can be built using coding techniques that are outlined in the associated guidelines.

2

# Business pattern assets

When describing the process of translating patterns into business solutions, in 1.3.1, “Selecting a pattern category” on page 8, we mentioned the primary Business patterns:

- Self-Service
- Collaboration
- Information Aggregation
- Extended Enterprise

This chapter summarizes the assets associated with each of these Business patterns by describing the Application and Runtime patterns and their variations.

Business and IT drivers of an organization govern the selection of a specific Business pattern and its associated Application and Runtime pattern. For information about a common set of Business and IT drivers, refer to the IBM Patterns for e-business Web site at:

http://www.ibm.com/developerWorks/patterns

This Web site provides additional information for each Application pattern, including a solution overview, guidelines for use of the Application pattern, the benefits of the Application pattern, and its limitations.

## 2.1 Self-Service business pattern assets

The Self-Service business pattern allows a customer of an organization to have access to information and processes that are necessary for conducting business with the organization. The access is through a user interface that could be a Web site, a personal digital assistant (PDA), or any other client interface.

### 2.1.1 Self-Service business application patterns

Seven Application patterns are associated with the Self-Service business pattern. Figure 2-1 provides a key to explain the main semantics associated with the Application patterns.

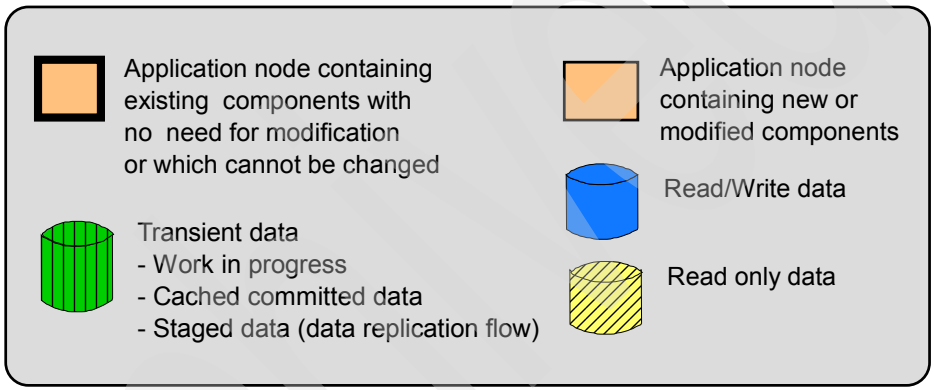



*Figure 2-1 Key for Application patterns for the Self-Service business pattern*

#### Stand Alone Single Channel

The Stand Alone Single Channel pattern, as illustrated in Figure 2-2, provides for stand-alone applications that have no need for integration with existing applications or data.

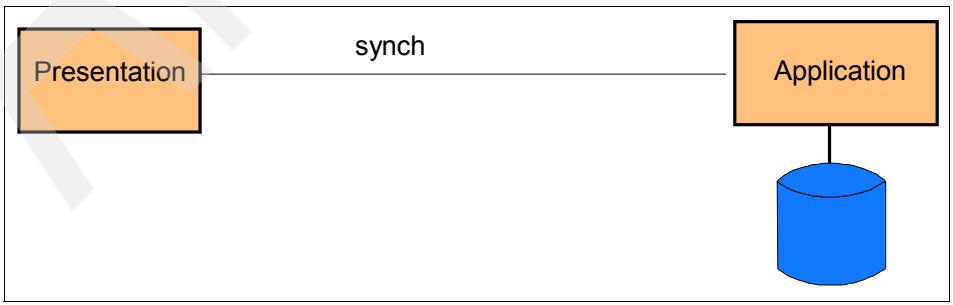



*Figure 2-2 Self-Service::Stand Alone Single Channel pattern*

## Directly Integrated Single Channel

As illustrated in Figure 2-3, the Directly Integrated Single Channel pattern provides point-to-point connectivity between the user and the existing back-end applications.

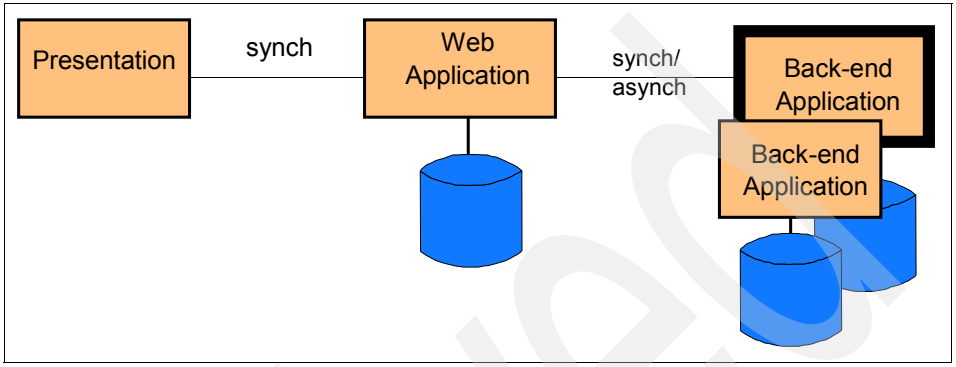



*Figure 2-3 Self-Service::Directly Integrated Single Channel pattern*

## As-Is Host application pattern

The As-Is Host application pattern (see Figure 2-4) provides simple, direct access to existing host applications. The application is unchanged, but the user access is translated from green-screen type access to Web browser-based access.


*Figure 2-4 Self-Service::As-Is Host application pattern*

## Customized Presentation to Host application pattern

For the Customized Presentation to Host application pattern, as illustrated in Figure 2-5, the back-end host application remains unchanged, but a Web application now translates the presentation from the back-end host application into a more user-friendly graphical view. The back-end host application is not aware of this translation.

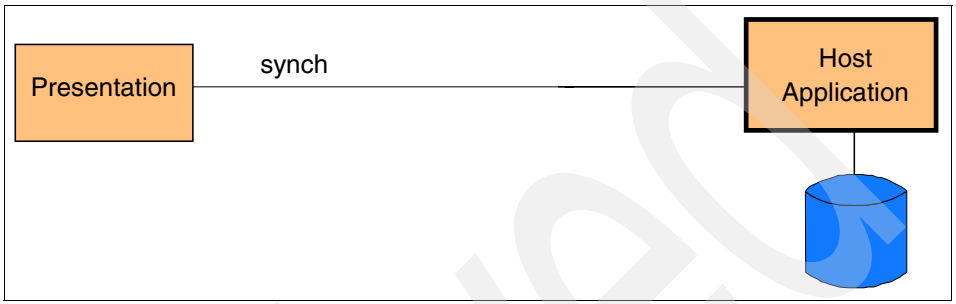



*Figure 2-5 Self-Service::Customized Presentation to Host application pattern*

## Router application pattern

As shown in Figure 2-6, the Router application pattern provides intelligent routing from multiple channels to multiple back-end applications using a hub-and-spoke architecture. The interaction between the user and the back-end application is a one-to-one relation, meaning the user interacts with applications one at a time.

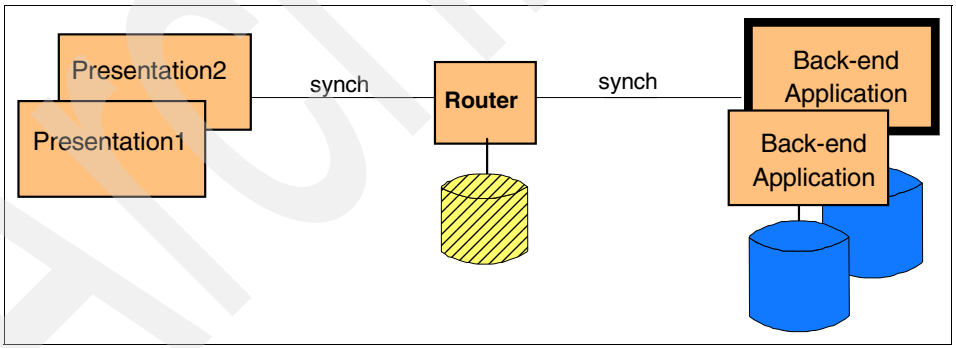



*Figure 2-6 Self-Service::Router application pattern*

## Decomposition application pattern

As illustrated in Figure 2-7, the Decomposition application pattern expands on the Router application pattern. It provides the ability to take a user request and decompose it into multiple requests to be routed to multiple back-end applications. The responses are recomposed into a single response for the user. This moves some of the business logic into the decomposition tier, but the primary business logic still resides in the back-end application.

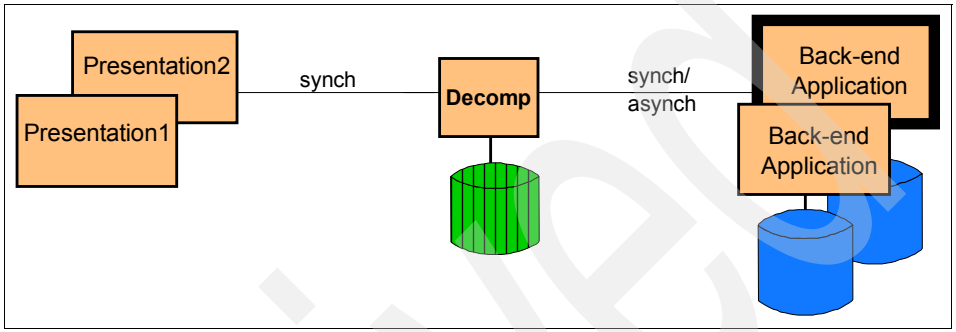



*Figure 2-7 Self-Service::Decomposition application pattern*

## Agent application pattern

As shown in Figure 2-8, the Agent application pattern structures an application design to provide a unified client-centric view that can be exploited for mass customization of services and for cross-selling purposes.

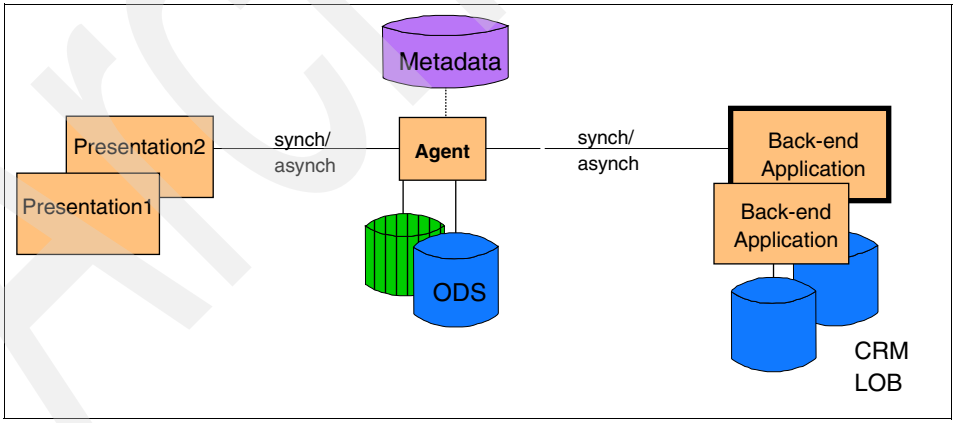



*Figure 2-8 Self-Service::Agent application pattern*

### Pattern differences

The differences in the patterns that we have mentioned depend on:

- Whether the application uses a stand-alone data source or accesses a back-end data source

  This accounts for differences between Application patterns as seen in Figure 2-2 on page 20 and Figure 2-3 on page 21.

- The degree to which the representation of the user interface of existing applications is changed prior to being presented to the client

  This accounts for the differences between the Application patterns in Figure 2-4 on page 21 and Figure 2-5 on page 22.

- The degree of functional richness provided by the interaction, transformation, or mediation service that mediates between applications and users

  This accounts for the differences between the Application patterns in Figure 2-6 on page 22, Figure 2-7, and Figure 2-8.

## 2.1.2 Self-Service business runtime patterns

There are essentially two Runtime patterns. Variation is due to the placement or location of the Web application server components within an organization's network structure. The first Runtime pattern, shown in Figure 2-9, has all the Web server components located in the demilitarized zone (DMZ).

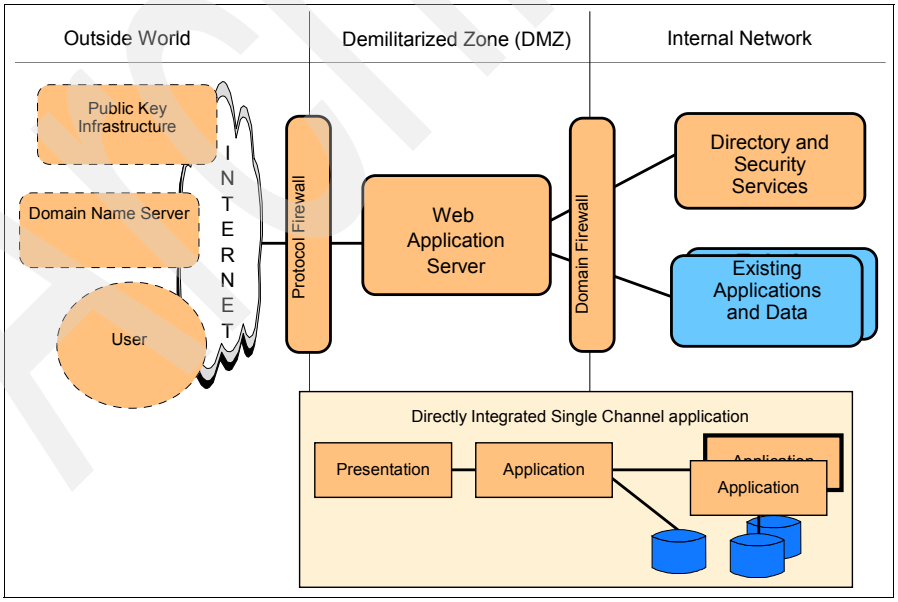



*Figure 2-9 Self-Service::Directly Integrated Single Channel::Runtime pattern*

The second Runtime pattern, shown in Figure 2-10, has only the Web server redirector in the DMZ. The remaining Web server components are located on the internal side of an organization's network structure.

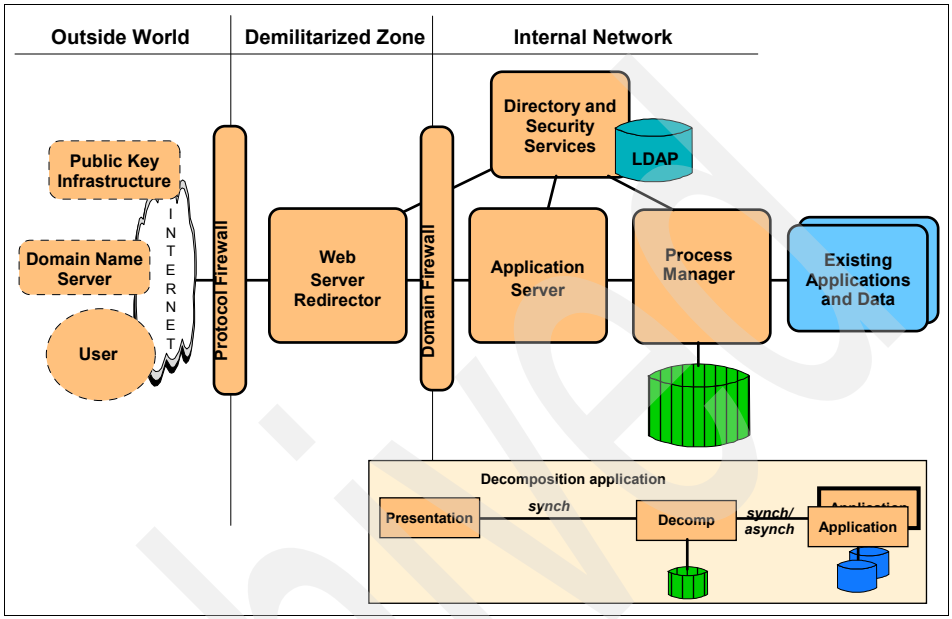



*Figure 2-10 Self-Service::Decomposition::Runtime variation 1*

Other variations to these Runtime patterns can be introduced to address non-functional requirements such as high availability and high performance.

Although the first Runtime pattern was historically used as an entry-level footprint, the proliferation of hacker attacks has caused it to be regarded as an anti-pattern. However for the moment, we have kept it on the Patterns Web site because it is used in the IBM Redbook *IBM WebSphere V5 Edge of Network Patterns,* SG24-6896, as the simplest base design to which various high availability and high performance nodes can be added:

http://www-106.ibm.com/developerworks/patterns/edge/index.html

## 2.2 Collaboration business pattern assets

The Collaboration business pattern, also known as the User-to-User (U2U) business pattern, allows users to communicate and share data and information with other users or groups of users in the network.

## 2.2.1 Collaboration business application patterns

Four Application patterns are associated with the Collaboration business pattern. Variations between these patterns depend on:

- Whether the collaboration is synchronous (users have to be online at the same time) or asynchronous (users can be off-line)
- Whether the collaboration is assisted by using a collaboration server on which the users have to register

The four Application patterns are explained and illustrated in the following sections.

### Point-to-Point application pattern

As illustrated in Figure 2-11, the Point-to-Point application pattern is used when collaboration is synchronous. Often, initiating the collaboration requires knowing the direct physical address of the other user, such as a TCP/IP address or a fully specified machine name. An example of such a collaboration is one supported by such applications as Telnet.

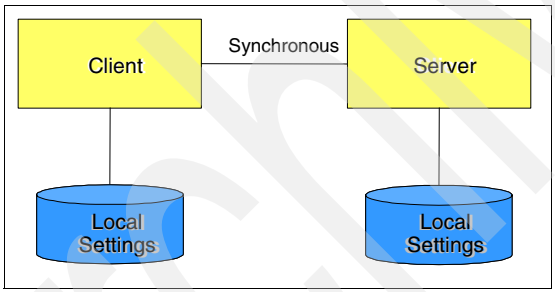



*Figure 2-11 Collaboration::Point-to-Point application pattern*

## Store and Retrieve application pattern

The Store and Retrieve application pattern, as illustrated in Figure 2-12, allows users to collaborate with others in the network interactively. Unlike the Point-to-Point application pattern, this pattern does not require both partners to be online at the same time. Nor does it require the customer to know the physical or direct address of other users of the solution.

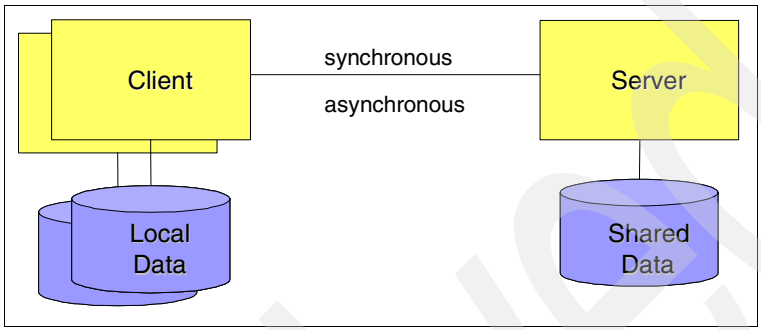



*Figure 2-12 Collaboration::Store and Retrieve application pattern*

## Directed Collaboration application pattern

As shown in Figure 2-13, the Directed Collaboration application pattern allows users to collaborate with others in the network interactively. This Application pattern requires the two users who need to interact to be online simultaneously. It also requires users to register with a server. In this pattern, all of the users are peers. There are no client-server or master-subordinate relationships between the tiers in the pattern.

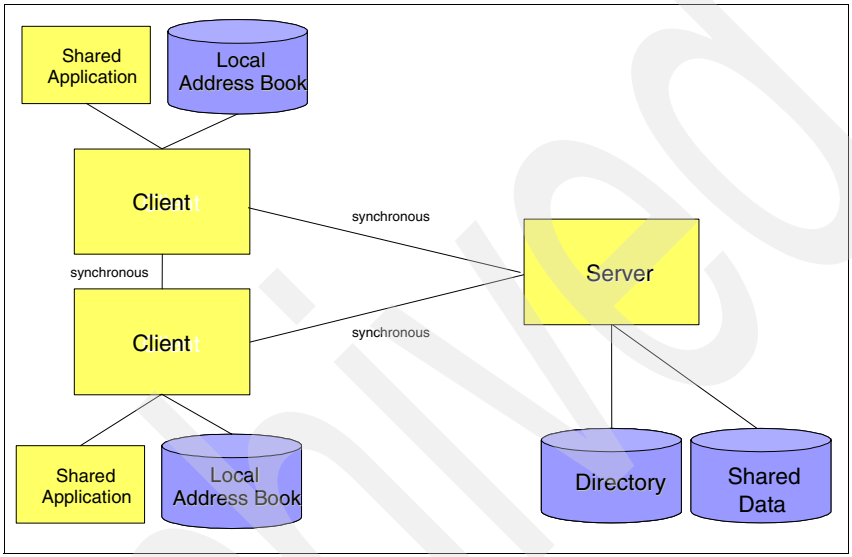



*Figure 2-13 Collaboration: Directed Collaboration application pattern*

### Managed Collaboration application pattern

The Managed Collaboration pattern builds on the Directed Collaboration pattern by implementing workflow rules to manage the collaboration between users of the solution. See Figure 2-14. It supports both synchronous and asynchronous collaboration.

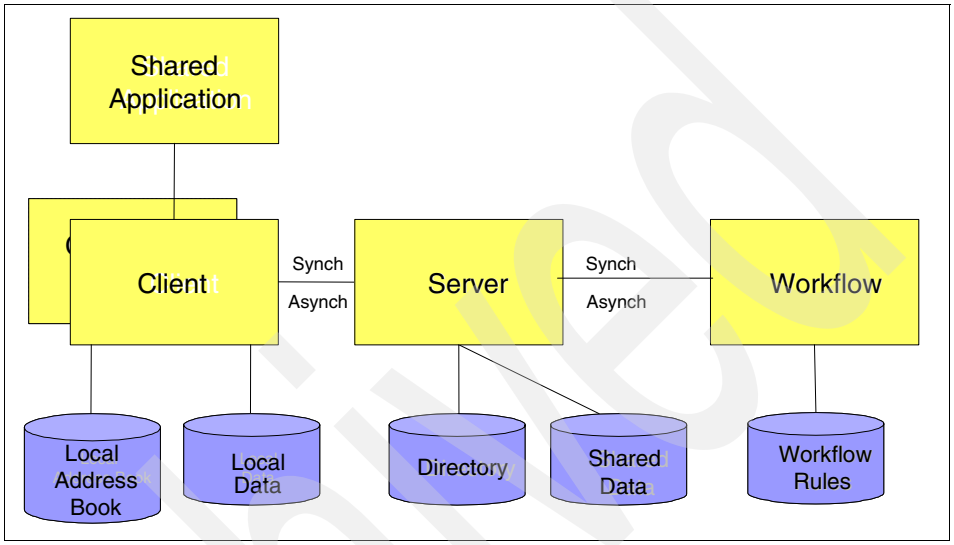



*Figure 2-14 Collaboration::Managed Collaboration application pattern*

## 2.2.2 Collaboration business runtime patterns

Currently there are no Runtime patterns for the Collaboration business pattern.

# 2.3 Information Aggregation business pattern assets

The Information Aggregation pattern, also known as the User-to-Data pattern, facilitates user access to and manipulation of data that is aggregated from multiple sources. Users employ tools to extract useful information from large volumes of data, text, images, video, and so forth. This gathered data often needs to be transformed, combined, and stored in a new data environment, for example for Business Intelligence and Knowledge Management. Both the User Information Access application pattern and the Population application pattern are typically combined together to meet this business requirement.

**Note:** The Data Population application patterns have been moved into the Application Integration pattern category and grouped as “Data Integration”.

There are three Application patterns for the Information Aggregation business pattern. Differentiation between them is based on:

- Whether access is to structured, such as business intelligence data, or unstructured, such as knowledge search data
- Whether the Application pattern supports data access that is read-only or that is changeable (either immediately or in batch)

The three Applications patterns are explained in the following sections.

### 2.3.1 User Information Access application patterns

As illustrated in Figure 2-15, an additional “drill-through” capability may be provided in the User Information Access application pattern. Such a facility is needed when the data store has multiple levels.

For example, a data mart may contain summarized multi-dimensional data that is regularly used, while the data warehouse contains the details. This function provides the user with the ability to drill through to detailed data when required. This drill-though capability is implied in the Application pattern diagram as an inherent function of the source data DBMS or access method.

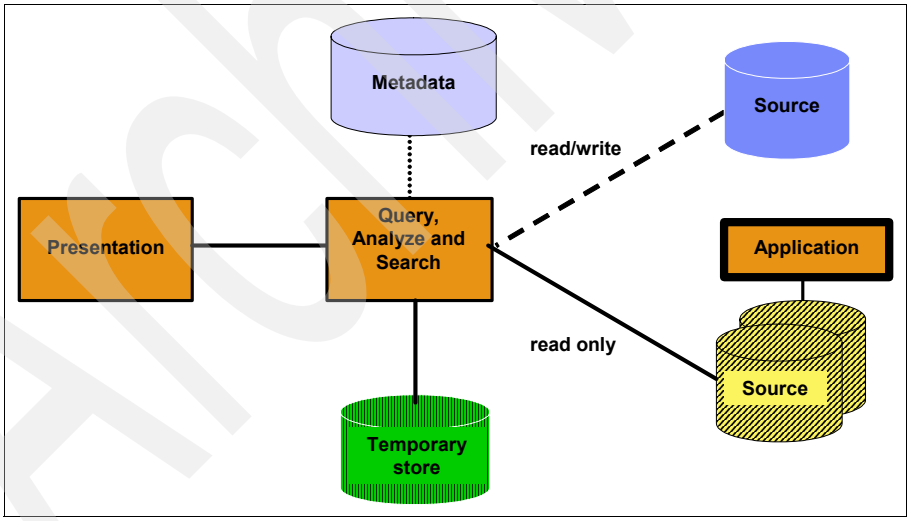



*Figure 2-15 User Information Access application pattern*

## 2.3.2 User Information Access=Write-back variation

As shown in Figure 2-16, use of this variation is indicated when a user needs in a largely read-only analysis type application (as supported by the User Information Access pattern) expand to updating or adding to existing information in the source systems such as operational systems. Typically, these needs are described by user needs such as “distributing what-if analyses for review” or “promoting planning scenarios to production”.

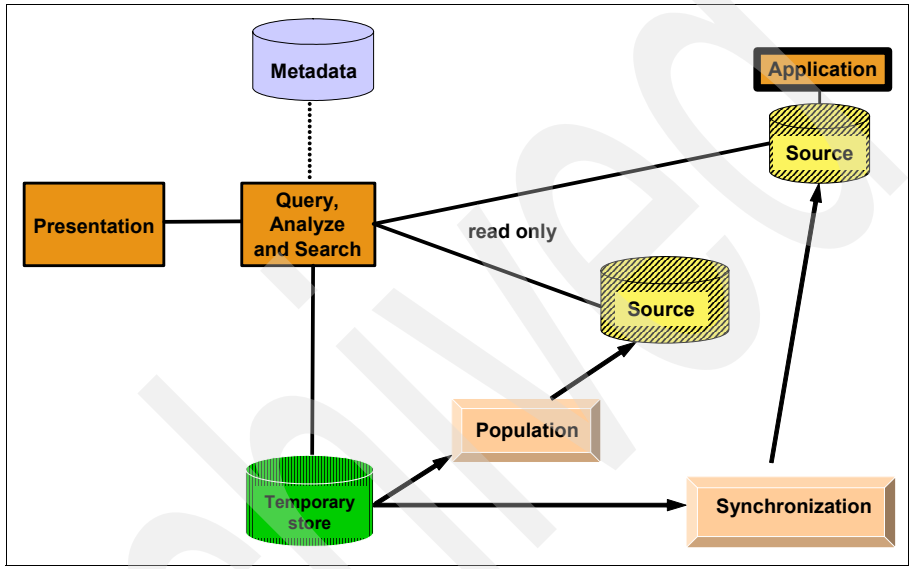



*Figure 2-16 User Information Access=Write-back variation*

## 2.3.3 User Information Access=Federation variation

Use of this variation pattern (see Figure 2-17) is indicated when multiple, diverse data sources exist that must be accessed in the same process. Federation provides both read-only and read/write access and allows access either directly to the data store or via an application programming interface (API) of the application.

The value provided by Federation here is in hiding the diverse access methods and data structures behind the single access method provided by the User Information Access pattern. Federation may also cache data as described earlier. This is omitted from Figure 2-17 for simplicity.

Federation also adds the possibility of writing to its data sources, and clearly the User Information Access: Federation variation can take advantage of this. While potential data integrity issues arise here and in the basic User Information

Access pattern, one advantage that Federation offers is the potential to do the write through of the API of the owning application rather than directly to the source data.

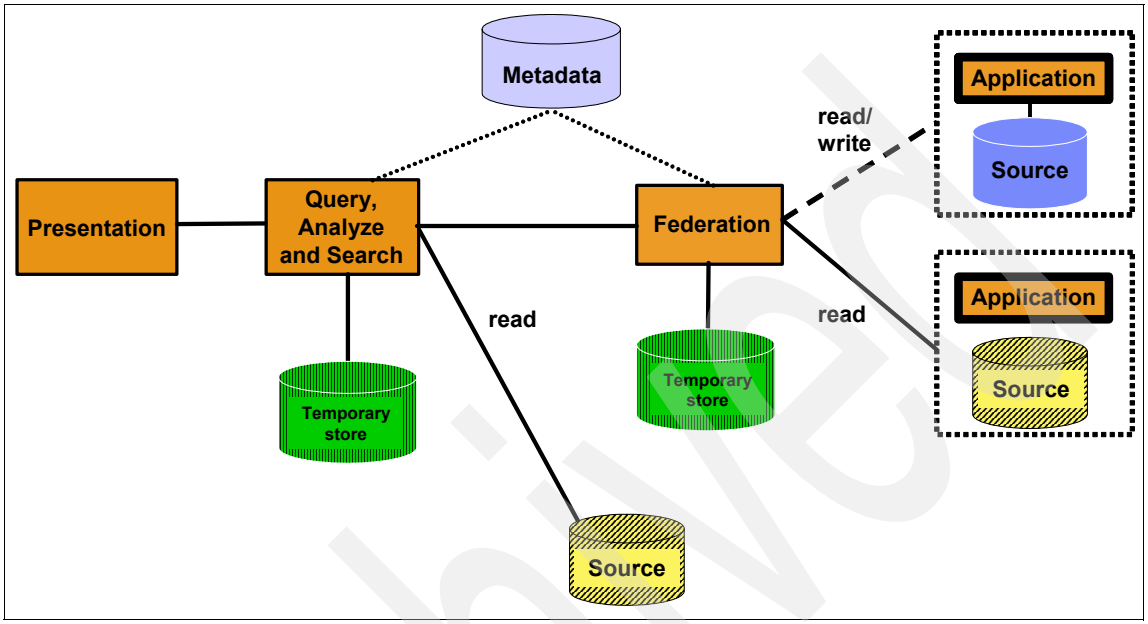



*Figure 2-17 User Information Access=Federation variation*

For additional details about the Population application patterns and Population runtime patterns, see the redbook *Patterns: Information Aggregation and Data Integration with DB2 Information Integrator*, SG24-7101.

## 2.4 Extended Enterprise business pattern assets

The Extended Enterprise business pattern, also called Business-to-Business (B2B) patterns, addresses the interactions and collaborations between business processes in separate enterprises. These separate enterprises are called *partners*.

Four application patterns are associated with the Extended Enterprise business pattern. Variations between these patterns depend on:

- How closely the partners are coupled by the connection between them
- The degree of functional richness of the collaboration service that connects the partners

The four application patterns are explained in the following sections.

## 2.4.1 Exposed Direct Connection application pattern

The Exposed Direct Connection application pattern represents the simplest interaction type based on a one-to-one topology (see Figure 2-18). The pattern allows a pair of applications to communicate with each other directly across organization boundaries. Interactions between a source and a target application can be arbitrarily complex. Generally, complexity can be addressed by breaking down interactions into more elementary interactions.

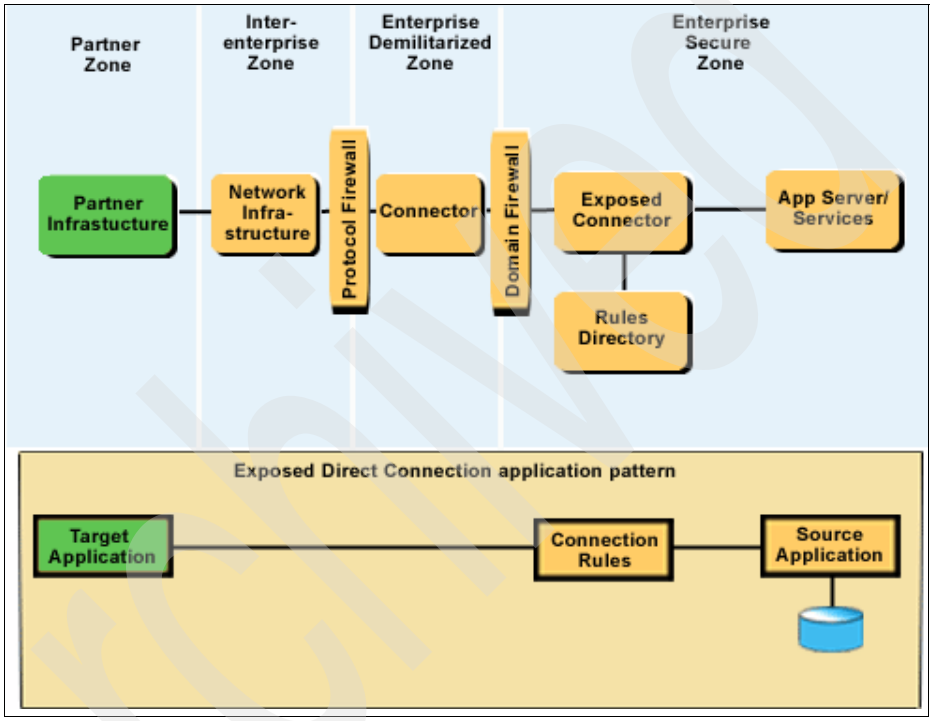



*Figure 2-18 Exposed Direct Connection application pattern::Runtime pattern*

## 2.4.2 Exposed Broker application pattern

The Exposed Broker application pattern, also known as *Exposed Business Services*, is based on a one-to-N topology that separates the distribution rules from the applications (see Figure 2-19). The pattern allows a single interaction from a partner's source application to be distributed to multiple target partner applications concurrently.

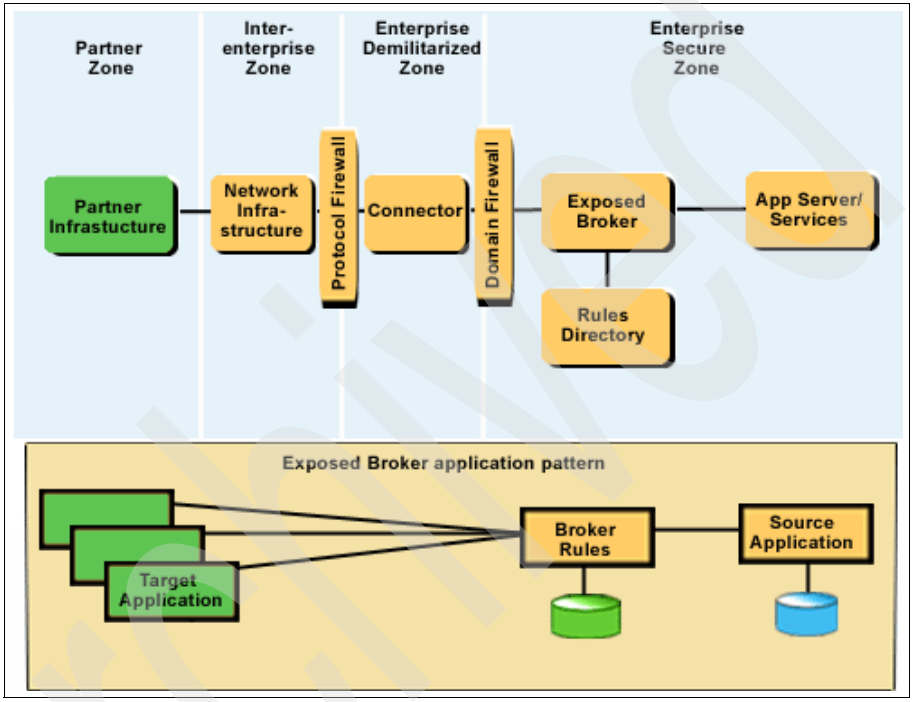



*Figure 2-19 Exposed Broker application pattern::Runtime pattern*

For more details, see the IBM Redbook *Patterns: Broker Interactions for Intra- and Inter-enterprise*, SG24-6075. This redbook introduces the Broker application pattern, and its Router variation. It provides Application and Runtime patterns, and product mappings with a focus on WebSphere Business Integration Message Broker and the WebSphere Application Server Web Services Gateway.

## 2.4.3 Exposed Router application pattern

As illustrated in Figure 2-20, the Router variation of the Exposed Broker application pattern applies to solutions where the partner's source application initiates an interaction that is forwarded to, at most, one of multiple target applications. The selection of the target application is controlled by the distribution rules that govern the functioning of the connector component.

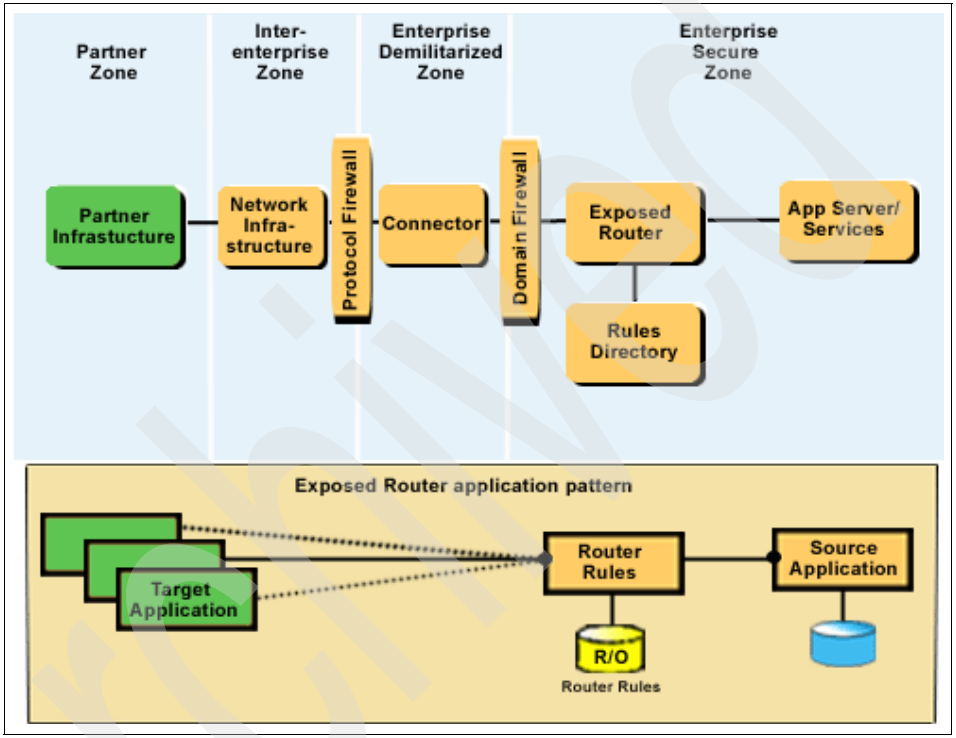



*Figure 2-20 Exposed Router application pattern::Runtime pattern*

### 2.4.4 Exposed Serial Process application pattern

The Exposed Serial Process application pattern extends the one-to-N topology provided by the Exposed Broker application pattern. It does this by facilitating the sequential execution of business services hosted by a number of target applications. It enables the orchestration of a serial business process across enterprise boundaries, in response to an interaction initiated by the source application.

For more information, refer to *Patterns: Serial Process Flows for Intra- and Inter-enterprise*, SG24-6305, and *Patterns: Serial and Parallel Processes for Process Choreography and Workflow,* SG24-6306. These IBM Redbooks consider key technologies that are relevant to the implementation of the Process-focused Application Integration pattern. They also consider flow languages, Web services, and Java™ Message Service.

3

# Integration pattern assets

Integration patterns are used in conjunction with Business patterns to create solutions that support complex business requirements. We can illustrate this by taking the business scenario of a bank customer (end-user) using an Internet interface to conduct business with the bank. The translation of this business scenario into an IT architecture is guided by the Self-Service business pattern as we mentioned in 1.3.2, “Selecting Application patterns” on page 11.

If the end-user experiences difficulties when using the application like a business process, such as bill payment or loan application, and is unable to resolve the problem, then frustration may lead the end-user to take their business elsewhere. In such a situation, providing an enhanced level of service to the end-user becomes important. This can be achieved by allowing the end-user to easily make contact with one of the bank’s customer services representatives (CSR) online. This extended capability of detecting online presence and initiating an online dialog is provided by the Collaboration business pattern.

This means that the solution now relies on two different Business patterns: Self-Service and Collaboration. The architect must have the means to determine how to tie these two Business patterns together. The Integration pattern assets guide the architect on how to achieve this.

As indicated in Table 1-3 on page 9, there are two Integration patterns categories:

- Access Integration pattern

  This pattern is designed to provide *users* with a seamless and consistent experience that combines *access to multiple applications, databases, and services*.

- Application Integration pattern

  This pattern brings together *multiple applications and information sources* without the user directly invoking them. There are two categories:

  – Data Integration patterns
  – Process Integration patterns

## 3.1 Access Integration pattern

The Access Integration pattern is designed to provide *users* with a seamless and consistent experience that combines *access to multiple applications, databases, and services*. Access Integration patterns are used to deliver an “integrated” user interface that often relies on functionality provided by products such as portals.

### 3.1.1 Access Integration application patterns

There are four Application patterns for the Access Integration pattern. Their differences are in the emphasis on the capability to be provided, which is security, personalization or pervasive device support.

#### Web Single Sign-On application pattern

The Web Single Sign-On application pattern provides a framework for seamless application access through unified authentication services. Single signon (SSO) allows the user to more quickly access information and applications without needing to sign onto each system individually. SSO can also lead to easier maintainability and a more consistent implementation of security policy by consolidating the administration of user ID entities.

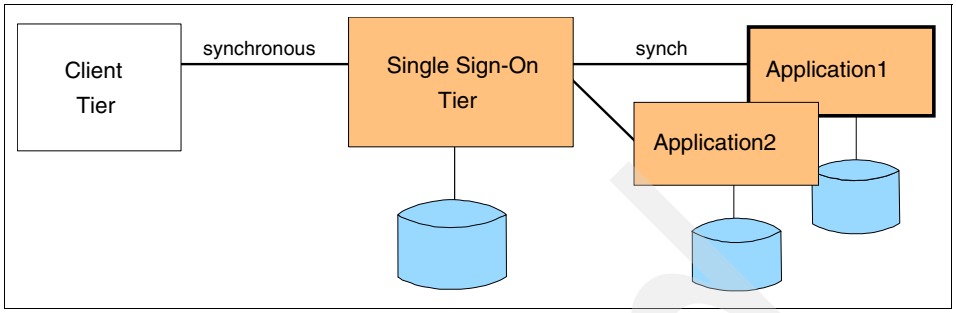



*Figure 3-1 Access Integration::Web Single Sign-On application pattern*

## Extended Single Sign-On application pattern

The Extended Single Sign-On application pattern provides additional capability on top of Web Single Sign-On by extending the authenticated identity context to a wider range of application systems. In Web Single Sign-On, the authenticated identity is normally visible only to the application tier that is directly adjacent to the SSO tier. Extended Single Sign-On adds a security integration tier that allows other application tiers to use the authenticated identity. This is known as *credential propagation*.

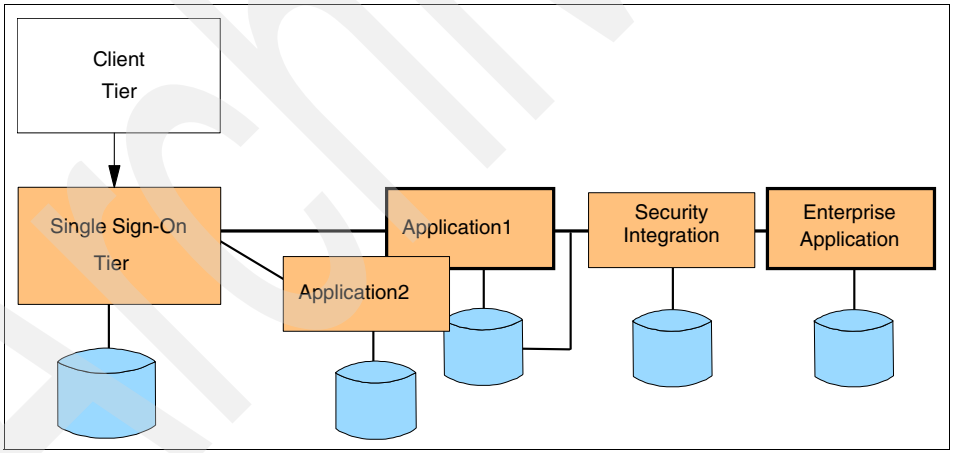



*Figure 3-2 Access Integration::Extended Single Sign-On application pattern*

## Pervasive Device Access application pattern

The Pervasive Device Access application pattern provides pervasive device access to an existing Business pattern by introducing a new tier into the architecture. This tier is responsible for the pervasive extensions to the original application and provides two key functions:

- A mapping mechanism between the client and the application so that the application can serve the correct markup to the client

  Advanced applications that are able to serve multiple markups can take advantage of this capability.

- Conversion of markup served by the application to a markup supported by the client, for example HTML to WML or WML to cHTML

  Basic applications that serve only one type of markup, such as HTML, can take advantage of this feature.

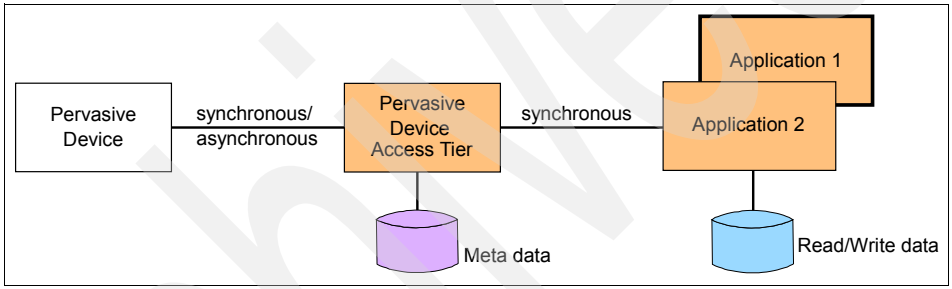



*Figure 3-3 Access_Integration::Pervasive Device Access application pattern*

## Personalized Delivery application pattern

The Personalized Delivery application pattern provides a framework for giving access to applications and information tailored to the interests and roles of a specific user or group. This Application pattern extends basic user management by collecting rich profile data that can be kept up-to-date by the user's current session. This Application pattern supports good design techniques because it provides for the separation of the business logic, business rules, and presentation layers.

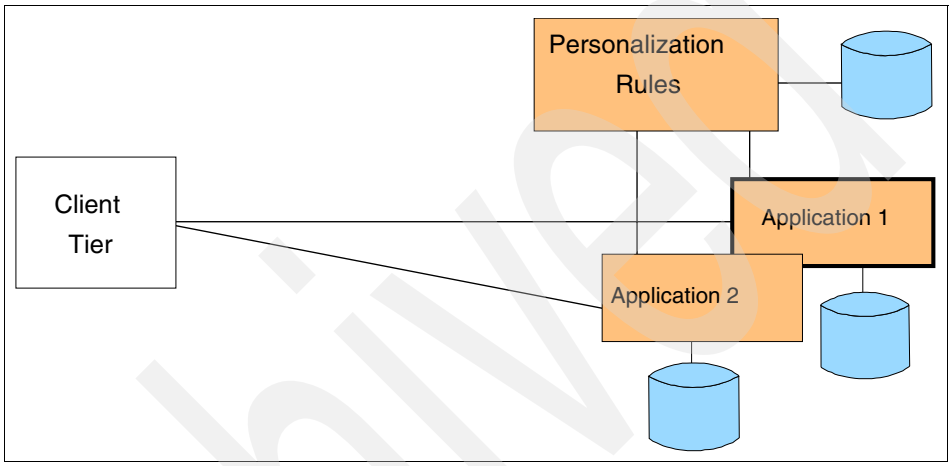



*Figure 3-4 Access Integration::Personalized Delivery application pattern*

Personalized delivery has three forms:

- **Participatory**: A user is allowed to tailor the content that they see.
- **Predictive**: A background application uses an inference engine to tailor the content to the user's behavior pattern automatically.
- **Prescriptive**: The organization specifies the content that the user can see.

For additional details relating to the personalized delivery application patterns, see *A Portal Composite site Pattern Using WebSphere Portal V5*, SG24-6087.

### 3.1.2 Access Integration runtime patterns

The Runtime patterns for the Access Integration pattern are:

- Web Single Sign-On to Homogenous application servers
- Web Single Sign-On to Heterogeneous application servers
- Extended Single Sign-On with Credential Propagation
- Extended Single Sign-On with Central Authorization Service
- Participatory personalization
- Predictive personalization
- Prescriptive personalization

However these Runtime patterns are often combined to form a Portal composite pattern which provides many of the capabilities of these Runtime patterns, as illustrated in Figure 3-5.

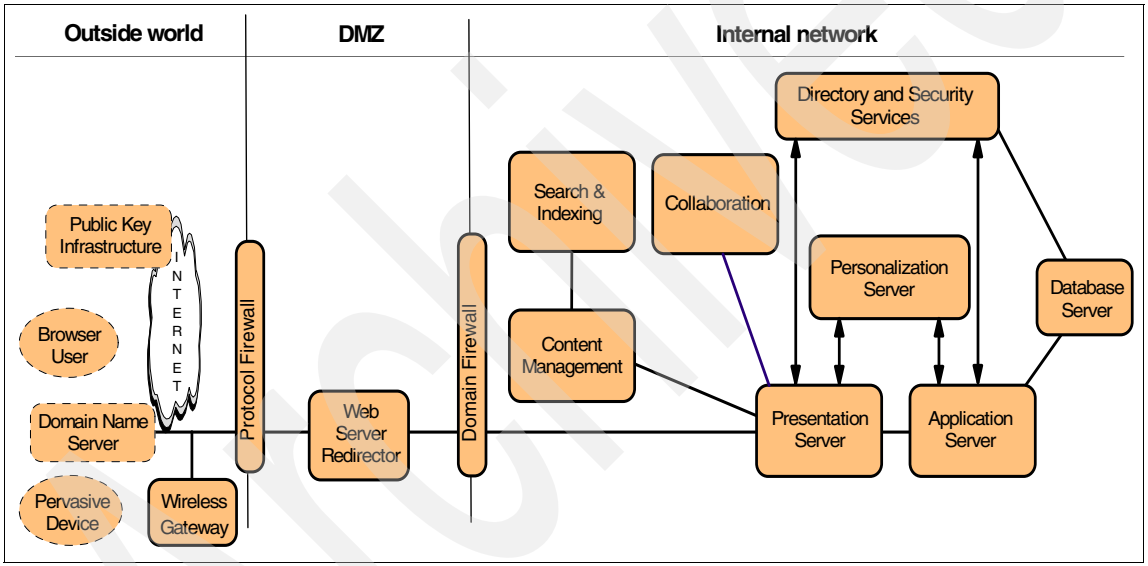



*Figure 3-5 Portal composite runtime pattern*

The Portal composite runtime pattern represents a starting point for most portal implementations, providing a way to identify those functional areas that will likely need to be addressed. It has a variation (Figure 3-6) which provides additional security capabilities through the inclusion of a reverse proxy node.

Note in Figure 3-6 the “presentation server” node has been renamed “portal server” in recognition of the fact that the presentation server has been enhanced to provide more functionality. The “database server” node has been replaced with an “existing applications and data” node in recognition of the fact that the portal now makes it possible to integrate far more resources than without the variation.

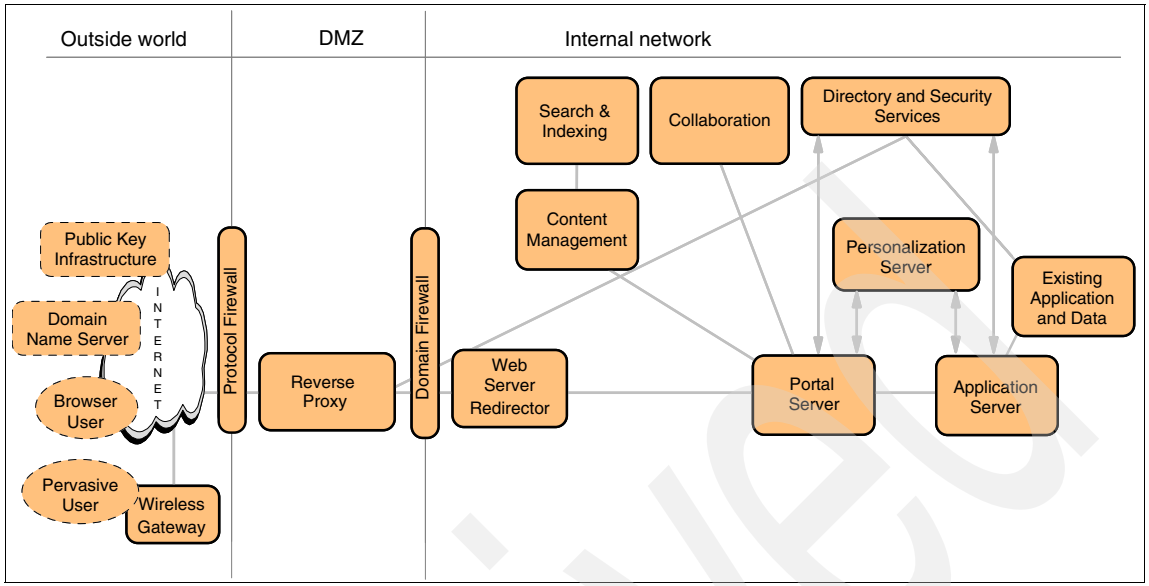



*Figure 3-6 Portal composite runtime pattern variation*

For more details about Runtime patterns, refer to these IBM Redbooks:

- *Access Integration pattern Using IBM WebSphere Portal Server,* SG24-6267
- *A Portal composite pattern Using WebSphere Portal V5,* SG24-6087

## 3.2 Application Integration pattern

The Application Integration pattern brings together *multiple applications and information sources without the user* directly invoking them. This pattern is most effectively applied when development efforts require the seamless execution of multiple applications and access to their respective data to automate a complex, new business function. To provide this capability, Application Integration patterns are divided into two categories.

- *Data Integration patterns* involve the integration of the information used by applications.
- *Process Integration patterns* involve the integration of the functional flow of processing between the applications.

## 3.2.1 Data Integration application patterns

Data Integration application patterns are more appropriate when applications need to share information rather than coordinate processing. There are two Data Integration application patterns as explained in the following sections.

### Federation application pattern

The Federation application pattern provides access to many diverse data sources and makes them appear as a single logical data store. Previously known as the *Federated Repository application pattern*, the Federation application pattern creates a unified query interface into isolated, structured and unstructured repositories, as illustrated in Figure 3-7.

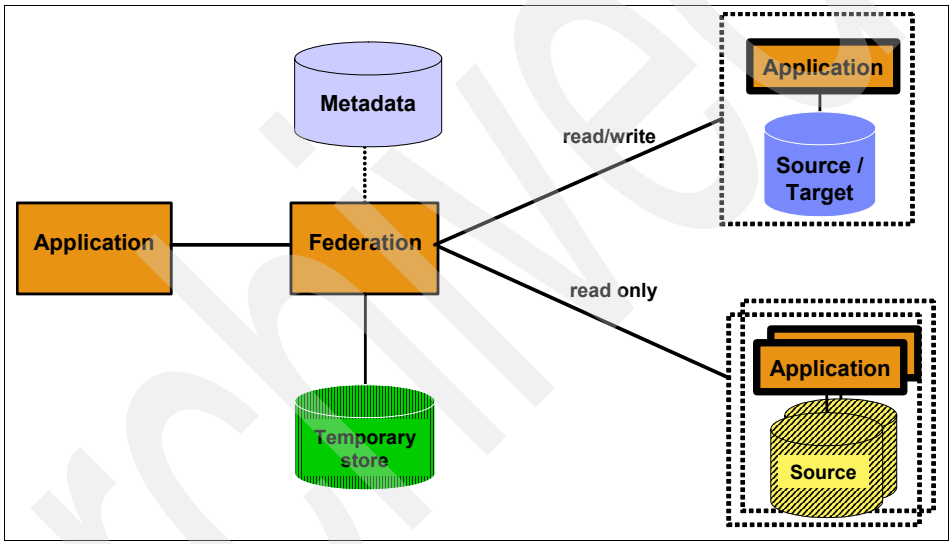
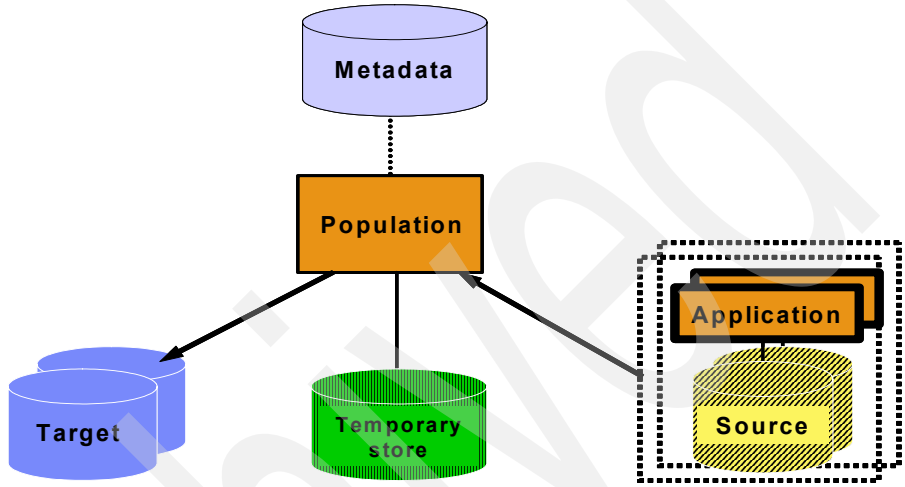



*Figure 3-7 Application Integration::Federation application pattern*

## Population application pattern

The Population application pattern gathers data from one or more sources and processes that data in an appropriate way. It also creates target data sets (store) with data from one or more sources. The extraction, transformation, and load rules may range from simple to complex. Figure 3-8 presents a legend for population application patterns to explain its four variations in more detail.




**LEGEND:**

Data sources are represented by disks in three different colors and shades:

- Blue/plain: Read/write
- Yellow/diagonal hatching: Read-only
- Green/vertical hatching: Temporary

Read/write and read-only refer only to the interaction between the overall pattern and that data source, as also indicated in some cases by annotation on the linkages. In general, we may assume that the application associated with a particular data source has read/write access.

A dotted box around an application and source data indicates that the source data may need to be accessed through the owning application via its API, or may be accessed directly via a database API. In general, a dotted box around a number of components indicates that we are not specifying which of those components we are interacting with.

A dashed line, arrow or component indicates an optional component.

*Figure 3-8 Population application pattern*

### *Population=Multi Step variation*

The building block provided by the Population application pattern is repeated several times to achieve the desired results. The intermediate target data created by one step acts as the source data for the subsequent step.

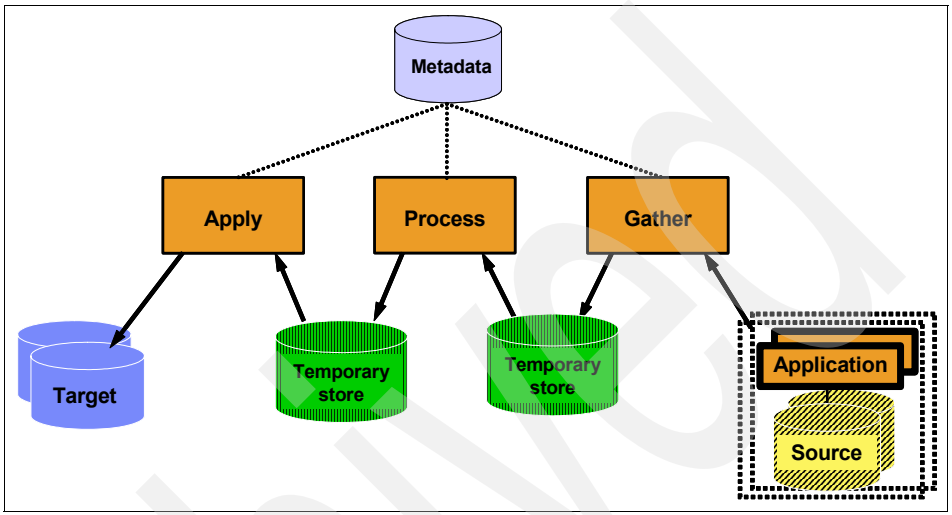



*Figure 3-9 Application Integration::Population=Multi Step variation*

### *Population=Multi Step Gather variation*

The transform and load steps from the Multi Step pattern have been combined into a single data analysis and cleansing stage. This stage does not so much transform the data, but rather validates and cleanses the data of errors.

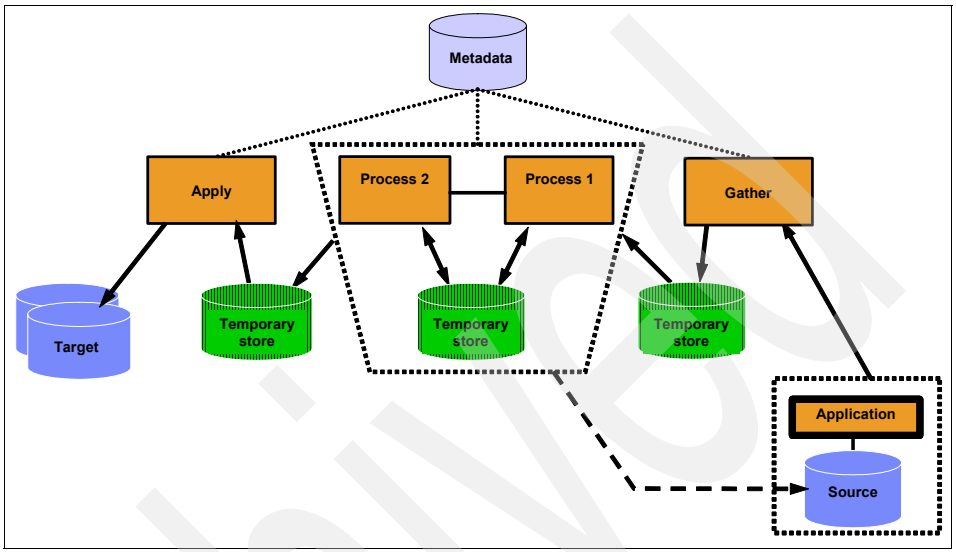



*Figure 3-10 Application Integration::Population=Multi Step Process variation*

### *Population=Multi Step Process variation*

The focus of this variation is on supporting population instances in the case of complex processing of the received data. It is impossible to perform in a single pass as shown in Figure 3-10.

### *Population=Multi Step Federated Gather variation*

This variation provides a unified query that accesses data in separate or remote, structured and unstructured repositories in real time. Figure 3-11 shows how the Population application pattern can be composed with the Federation application pattern as a means to gather data from one or more sources.

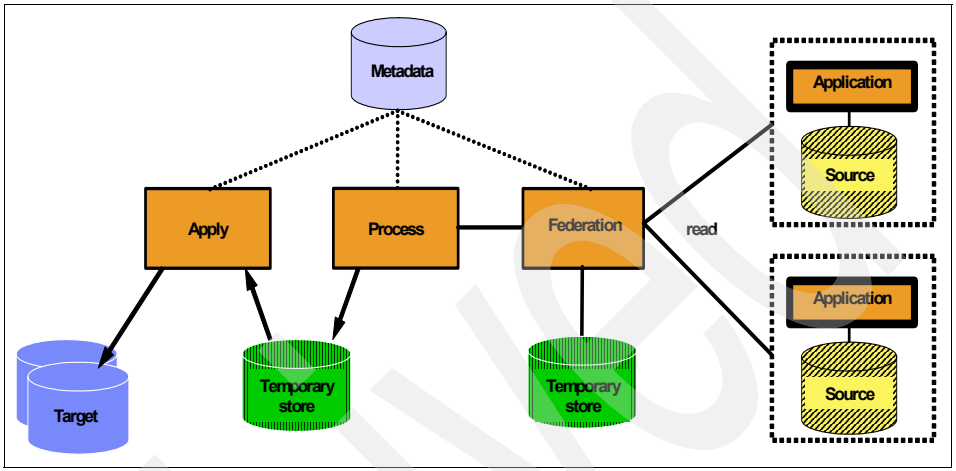



*Figure 3-11 Application Integration::Population=Multi Step Federated Gather variation*

### *Data sets*

The Population application pattern is designed to help the architect develop solutions that are appropriate for the creation of different types of target data-sets. These different data sets include data repositories such as:

- Data marts

  A data mart is a data store defined and designed to meet the information needs of a department or group of users. It contains the necessary data that is detailed or summarized, and structured, according to the query or reporting needs of the user. The data mart is the primary source of information for end users. The data in the data mart is accessed most often in read-only formats, but occasionally in a read-write format. Data marts come in a variety of forms, including relational databases, dimensional databases, and spreadsheets.

- Web marts

  An increasingly important form of data a mart is the Web mart. It stores data in a Web page or similar format directly usable by Web browsers.

- Business data warehouse

  The business data warehouse is a data store that contains detailed, reconciled, and historical data. It is structured according to an enterprise data model and designed to be the single, consistent source for all management

information. If end-users have access to this data source, then the data is available in a read-only format.

- Operational data stores

  An operational data store is a data store that contains detailed, partially reconciled, and nearly current data used for immediate reporting needs. Users can often write additional data to this form of data store.

Data is often moved between these data sets. These Application pattern variations differ due to:

- Use of single or multiple extract, transformation and load steps
- Use of data discovery techniques for structured and unstructured data
- Enhancement of data with additional (summary) information

### 3.2.2 Data Integration runtime patterns

The Data Integration patterns provide three types of data manipulation:

- Data access in place using the Federation application pattern
- Data movement using Population application patterns
- Data update:
  - Bidirectional, using the Two-way Synchronization application pattern
  - Update of target, using the Collect application pattern

Variations on these three data manipulation styles are based on data integration nodes and a degree of parallelism in accessing the source data. This results in many Runtime patterns. However, they all follow the general form shown in Figure 3-12.

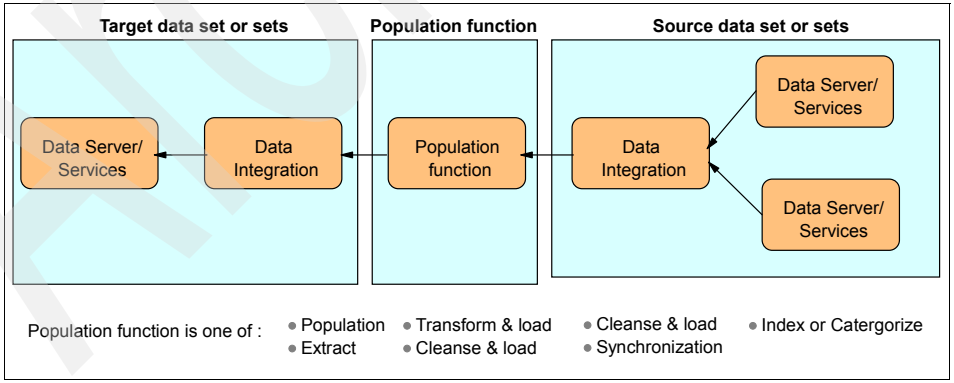



*Figure 3-12 Runtime patterns for Data Integration*

Note that this is a generalized Runtime pattern. The Population function node can be interspersed between the Data Server/Services nodes and the Data Integration nodes as often as required. The Data Integration nodes are optional.

For additional details about the Data Integration application patterns and Runtime patterns, see *Patterns: Information Aggregation and Data Integration with DB2 Information Integrator*, SG24-7101.

### 3.2.3 Process Integration application patterns

The major Application patterns in the Process Integration category are:

- Application Integration::Direct Connection=Call
- Application Integration::Direct Connection=Message
- Application Integration::Broker
- Application Integration::Broker=Router
- Application Integration::Serial Process
- Application Integration::Serial Workflow
- Application Integration::Parallel Process
- Application Integration::Parallel Workflow

The difference between these patterns is based on the functional richness of the integration service that provides the connectivity/mediation between applications.

Figure 3-13 shows the general form of the Process Integration pattern.

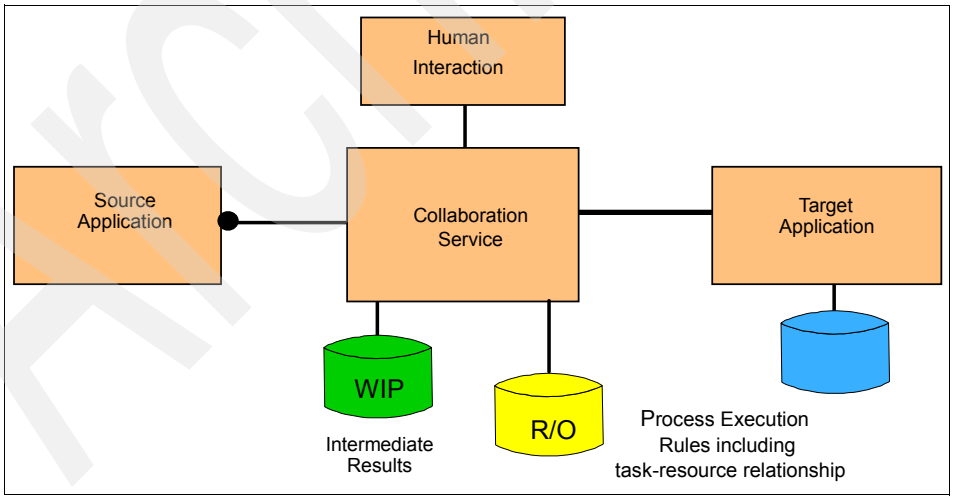



*Figure 3-13 Application Integration::Generic application pattern*

You can find details about the current approach regarding these Application patterns in the following IBM Redbooks:

- *Patterns: Direct Connections for Intra- and Inter-enterprise, SG24-6933*
- *Patterns: Broker Interactions for Intra- and Inter-enterprise*, SG24-6075
- *Patterns: Serial Process Flows for Intra- and Inter-enterprise*, SG24-6305
- *Patterns: Serial and Parallel Processes for Process Choreography and Workflow*, SG24-6306
- *Patterns: Information Aggregation and Data Integration with DB2 Information Integrator*, SG24-7101

At the time of writing this redbook, the most sophisticated Application pattern for Process Integration is the Application Integration::Parallel Workflow pattern, which is illustrated in Figure 3-14.

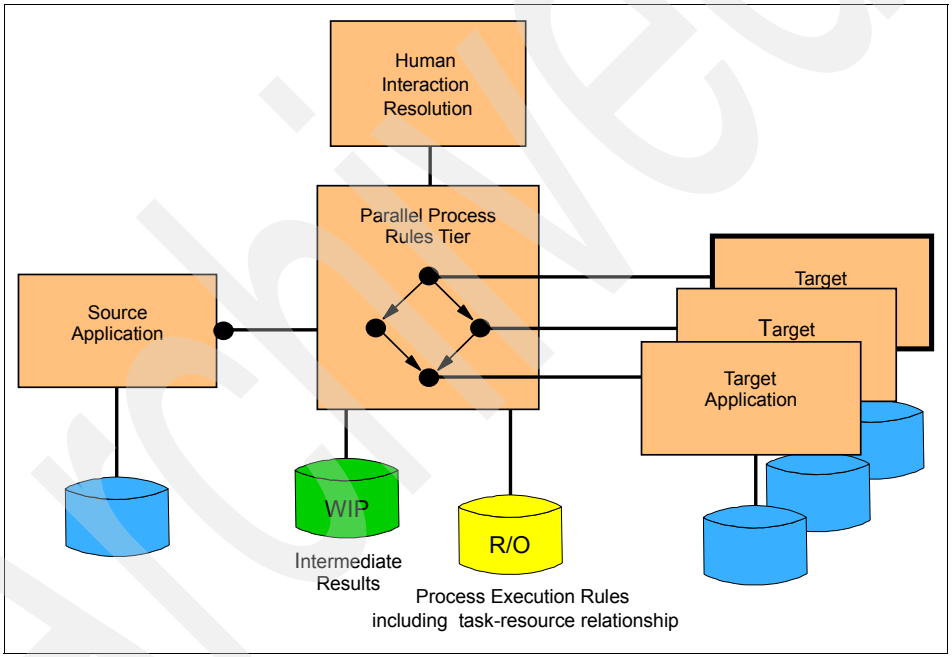



*Figure 3-14 Application Integration::Parallel Workflow application pattern*

The Parallel Workflow application pattern improves the flexibility and responsiveness of an organization by implementing end-to-end process flows that externalize process logic from individual applications. Further flexibility is introduced by externalizing task-resource resolution rules.

This Application pattern supports the reduction of cycle time by supporting parallel execution of portions of a process flow. It also provides a foundation for

automated support for Business Process Management that enables monitoring and measurement of the effectiveness of business processes.

The Application patterns that were introduced in this chapter are designed to meet the need for integrating processes that are internal to an organization. These patterns are said to support intra-enterprise process integration. If there is a need to allow access to these processes by an external organization, then the Extended Enterprise application patterns should be used.

### 3.2.4 Process Integration runtime patterns

Since there are eight Process Integration application patterns, there is also a large number of Runtime patterns. For details about individual Runtime patterns, refer to the IBM Redbooks and Redpapers mentioned in 3.2.3, “Process Integration application patterns” on page 50. For your appreciation of the functional capabilities provided by such a Runtime pattern, we include the Parallel Workflow runtime pattern in Figure 3-15.

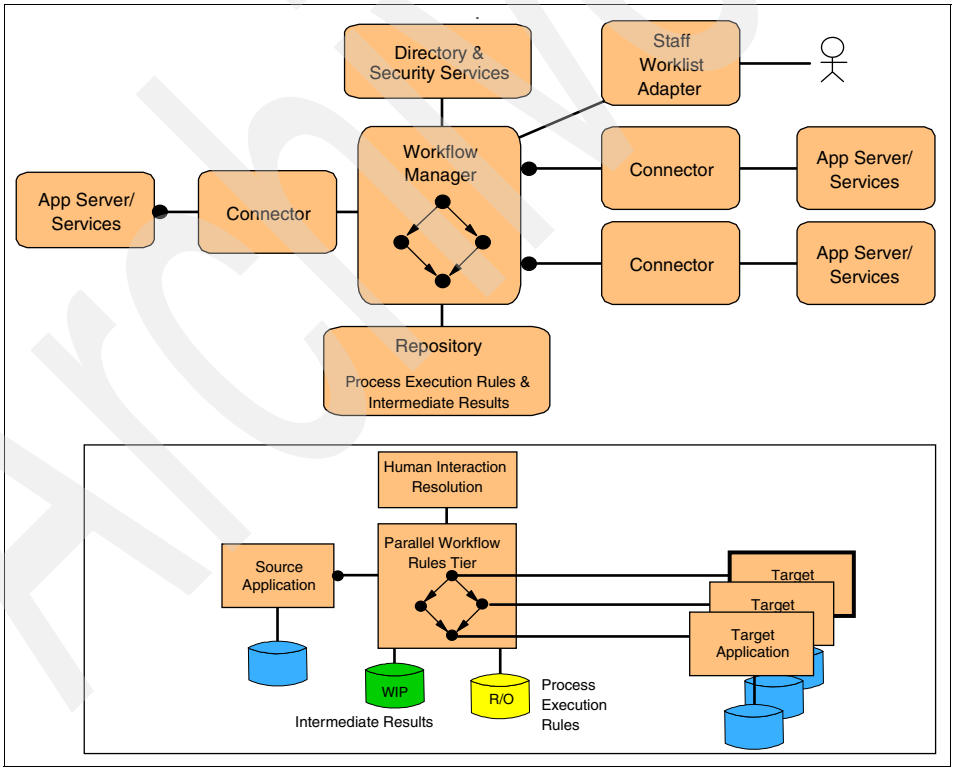



*Figure 3-15 Application Integration::Parallel Workflow runtime pattern*

4

# Composite patterns and custom design assets

Composite patterns are combinations of Business and Integration patterns. The most common Composite patterns, which are explained in more detail in this chapter, are:

- Electronic Commerce
- e-Marketplace
- Portal
- Account Access

As their names suggest, custom design assets are specialized, customized patterns. Their explanations are best left to the documents in which they are covered extensively. Two of these assets are:

- Integrating the Self-Service and Collaboration patterns using WebSphere and Domino in *Applying the Patterns for e-business to Domino and WebSphere Scenarios,* SG24-6255
- Integrating WebSphere Application Server with SAP R/3 in *Integrating WebSphere Application Server with SAP R/3*, a presentation by Roland Schütze et. al., on the Web at:

  `ftp://www6.software.ibm.com/software/developer/patterns/ws-sap-integration.pdf`

## 4.1 Electronic Commerce composite patterns

Electronic Commerce combines the Self-Service and Information Aggregation business patterns with the Application Integration pattern. Additionally, an electronic commerce site can integrate supply chain management functionality into the solution through the Extended Enterprise pattern, or e-mail confirmation of orders to clients through the Collaboration pattern.

The Electronic Commerce composite pattern has two Application pattern variations:

- The Web-up application pattern, which defers updates to back-end systems
- The Enterprise-out application pattern, which provides online access to back-end systems

### 4.1.1 Application patterns for the Web-up variant

The Web-up application pattern for the Electronic Commerce composite pattern is applicable if you are building a new application and there is no need to interface with legacy or third-party applications or data. All of the required data is handled by the new e-business application, as illustrated in Figure 4-1.

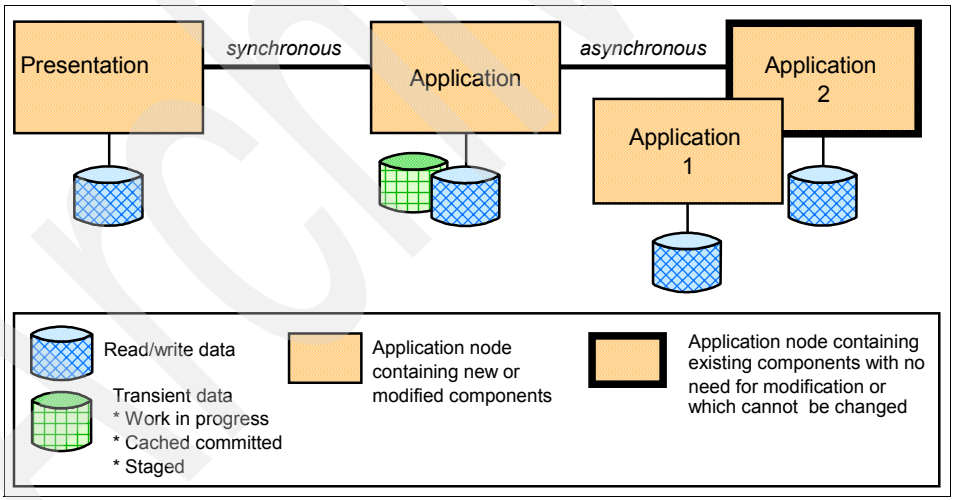



*Figure 4-1 Application pattern for the Web-up variant*

## 4.1.2 Runtime patterns for the Web-up variant

There are three variations for the Runtime pattern. Their differences are based on the placement of the commerce server within different segments of an organization's network structure and the degree of security represented by the number of protocol firewalls.

A common Runtime variation is illustrated in Figure 4-2. For additional details about Web-up runtime patterns, see the IBM Patterns for e-business Web site:

http://www.ibm.com/developerWorks/patterns

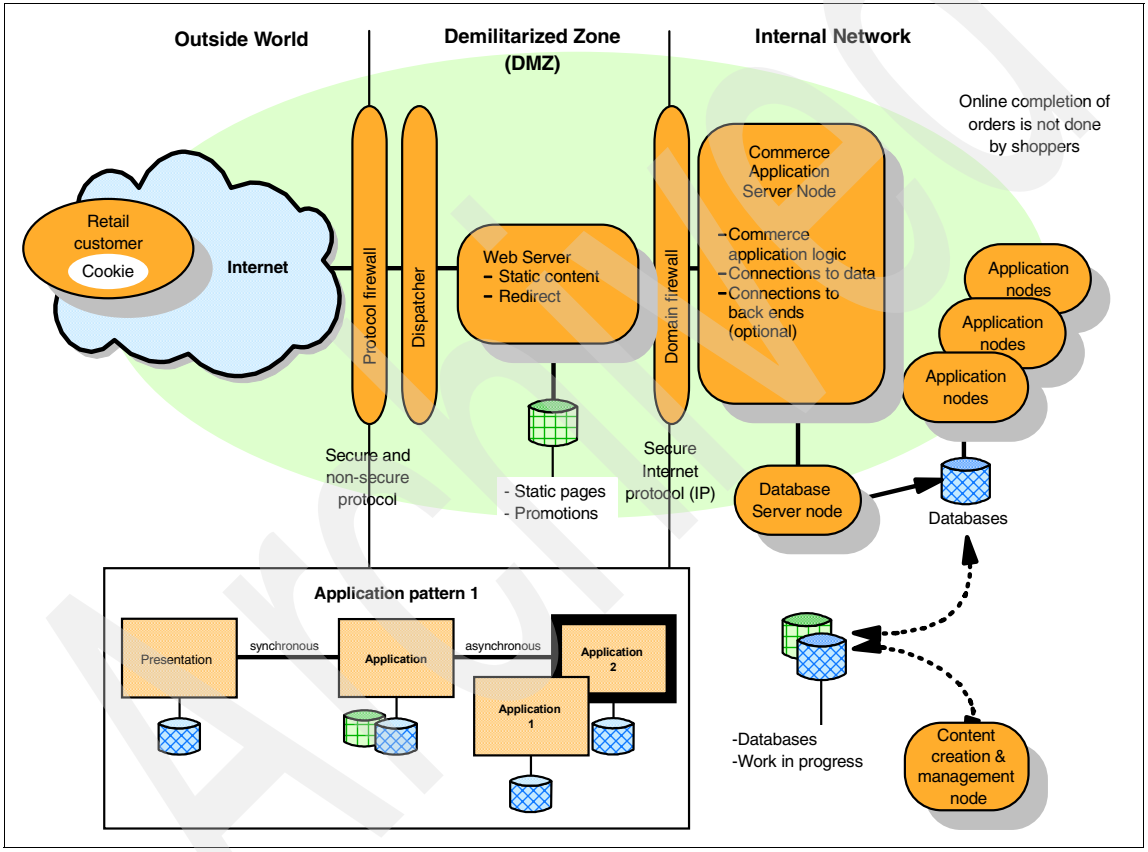



*Figure 4-2 Runtime pattern for the Web-up variant*

## 4.1.3 Application pattern for the Enterprise-out variant

The Enterprise-out application pattern applies to scenarios in which you need integration of existing or third-party applications. The new application is not built as a stand-alone solution, but instead needs to integrate with existing applications, as illustrated in Figure 4-3. The integration can be achieved on a functional basis or a data basis using a transaction.

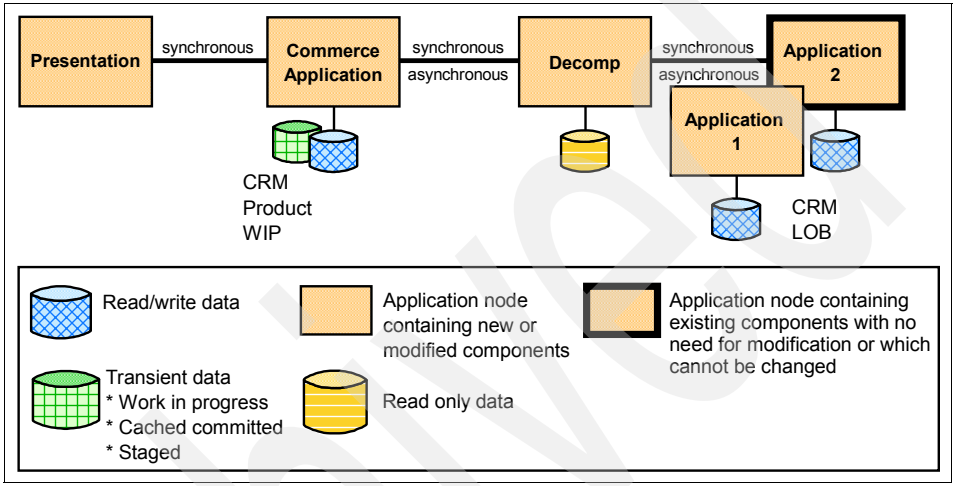



*Figure 4-3 Application pattern for the Enterprise-out variant*

## 4.1.4 Runtime patterns for the Enterprise-out variant

There are three variations for the Enterprise-out Composite runtime pattern. Their differences are based on the placement of the commerce server within an organization's network structure and the degree of security represented by the number of protocol firewalls.

A common runtime variation is shown in Figure 4-4. For additional details about this and other Web-up runtime patterns, see the IBM Patterns for e-business Web site.

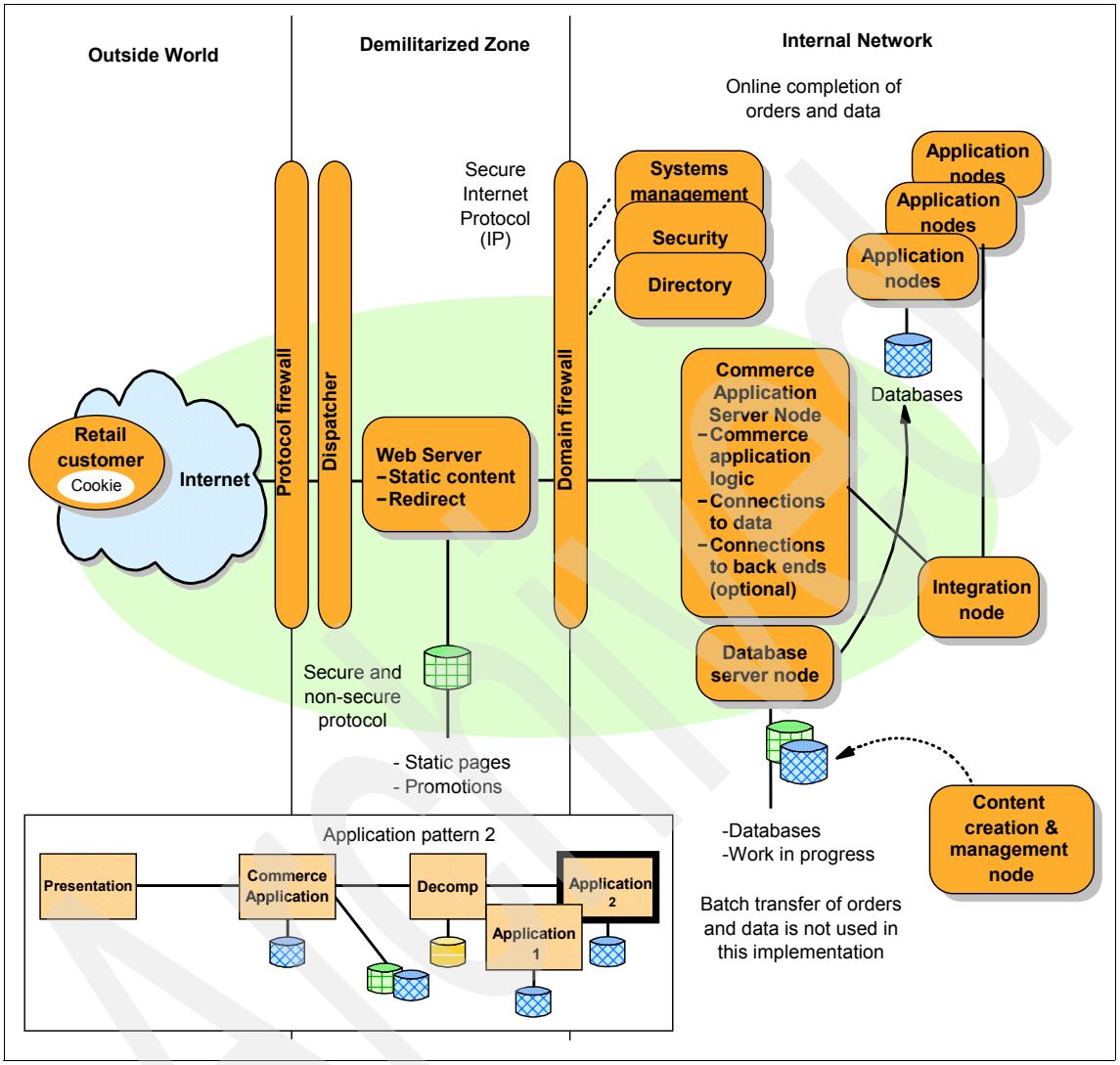



*Figure 4-4 Runtime pattern for the Enterprise-out variant*

## 4.2 e-Marketplace composite patterns

e-Marketplaces are trading exchanges that facilitate and promote buying and selling among trading partners within an industry. Three types of e-Marketplaces have been defined by the IBM Patterns for e-business:

- *Trading Exchange* allows buyers and sellers to trade goods and services on a public site.
- *Sell-Side Hub* is an e-Marketplace owned by the seller, who uses it to sell goods and services to prospective buyers across the Internet either through a browser, a pervasive device, or an interface with a client's enterprise resource planning (ERP) or Buy-Side Hub system.
- *Buy-Side Hub* is an e-Marketplace owned by the buyer of the goods or services, who uses it for procurement and to solicit the best deals from prospective clients across the Internet, either through a browser, a pervasive device, or an interface with a client's ERP or Sell-Side Hub system.

All of these e-Marketplaces leverage the following patterns:

- Access Integration pattern, by providing a unified client interface, which is mandatory
- Self-Service business pattern, by allowing users to browse through a catalog and place an order with the hub
- Information Aggregation business pattern, by creating a catalog from the supplier's product files, pricing files, and marketing collateral
- Application Integration pattern, by integrating Business patterns that make up the Sell-Side Hub

The e-Marketplaces composite patterns may also leverage the following additional patterns:

- Collaboration patterns, by enabling dynamic trading mechanisms, including auctions
- Extended Enterprise business patterns, by providing additional capabilities such as payment gateways

Figure 4-5 shows a Composite Sell-Side Hub runtime pattern.

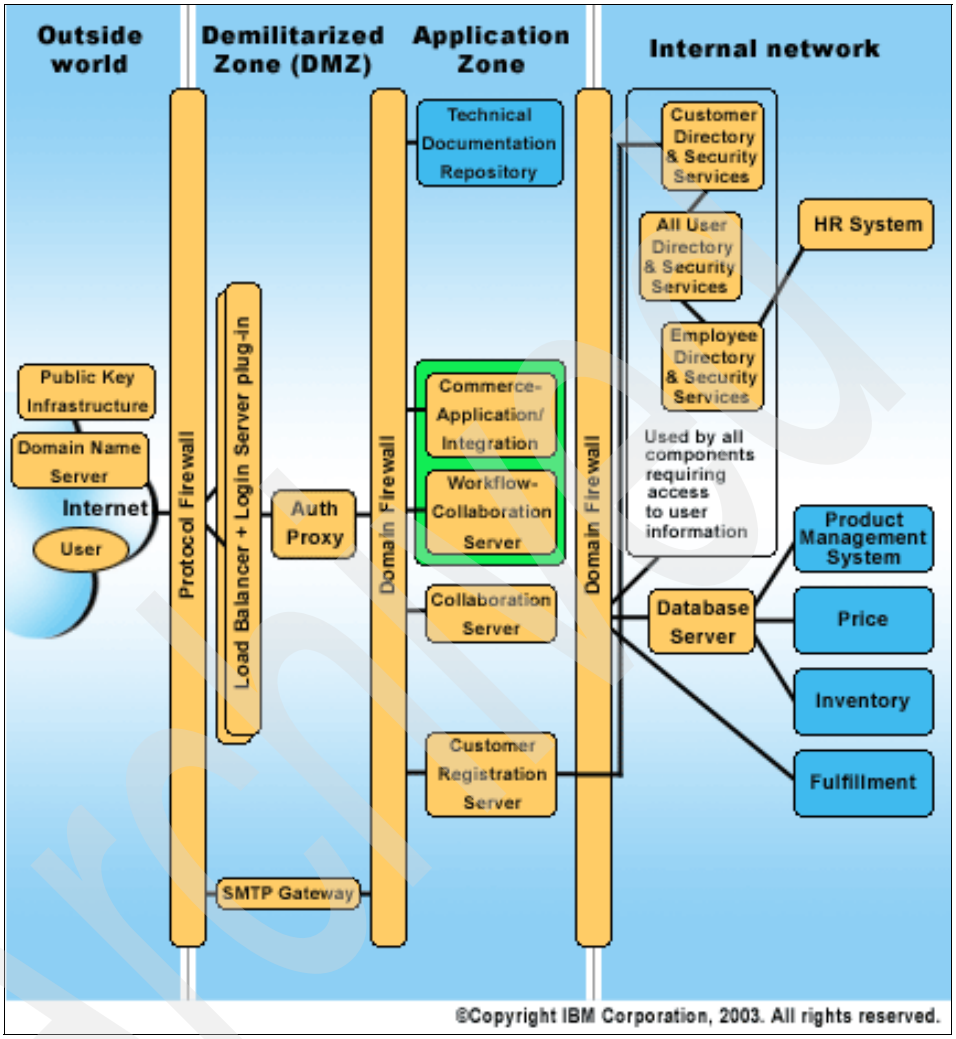



*Figure 4-5 e-Marketplace composite pattern:: Sell-Side Hub runtime pattern*

Figure 4-6 shows another Runtime pattern for e-Marketplace.

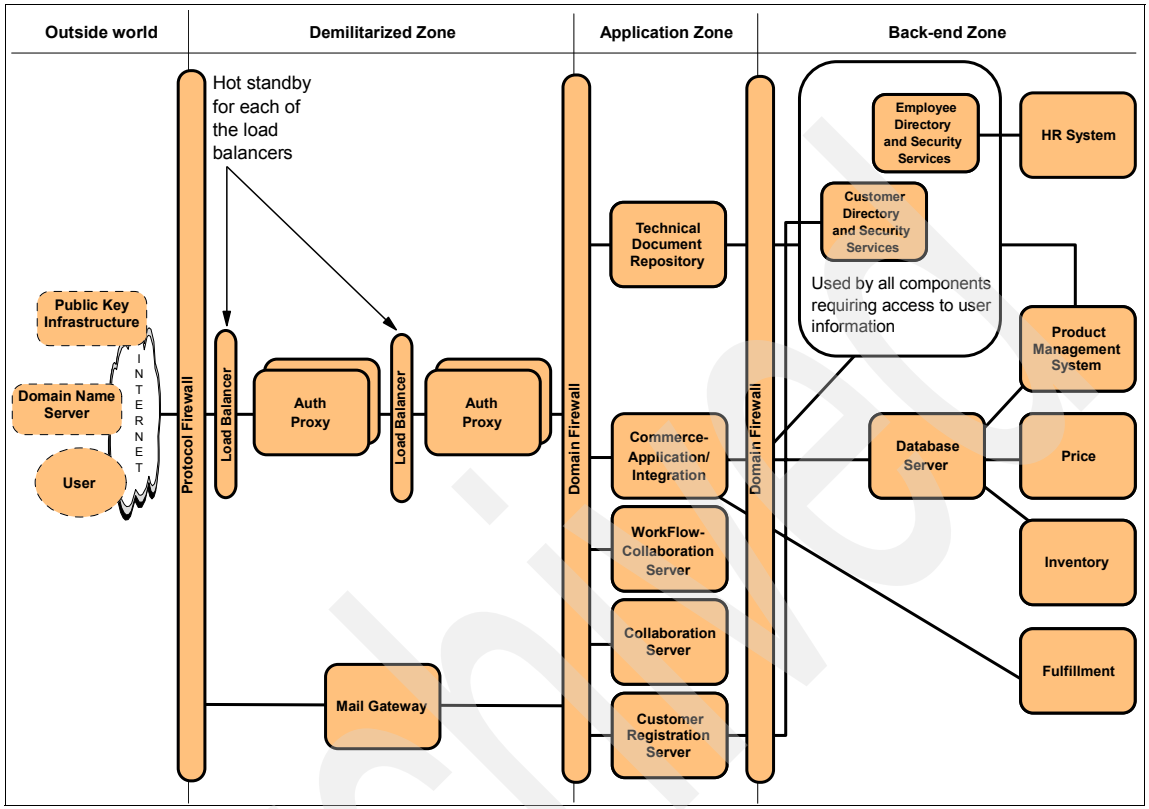



*Figure 4-6 Runtime pattern for e-Marketplace*

## 4.3 Portal composite patterns

The Portal composite patterns specify all the capabilities required to provide users with a seamless and consistent user experience by allowing integrated access to multiple applications, databases, and services.

The Portal composite patterns incorporate:

- Three Business patterns
  - Self-Service
  - Collaboration
  - Information Aggregation
- Two Integration patterns
  - Access Integration
  - Application Integration

### 4.3.1 Portal composite application patterns

The Portal composite application patterns can be used to develop highly complex portal applications that offer multiple services to users at one centralized location. Since the Portal composite patterns use so many Business and Integration patterns, it is understandable that the Business pattern is supported by a large number of Application patterns, which are briefly outlined here for easy reference:

- Access Integration
  - Access Integration::Web Single Sign-On

    This pattern provides a framework for seamless application access through unified authentication services.
  - Access Integration::Pervasive Device Access (optional)

    This pattern provides consistent access to various applications using multiple device types.
  - Access Integration::Personalized Delivery

    This pattern provides a framework for giving access to applications and information tailored to the interests and roles of a specific user or group.
- Self-Service::Directly Integrated Single Channel

  This pattern provides a structure for applications that need one or more point-to-point connections with back-end applications but only need to focus on one delivery channel.

- Collaboration
  - Collaboration::Store and Retrieve

    This pattern allows users to collaborate with others in the network interactively. Unlike the Point-to-Point application pattern, this pattern does not require both partners to be online at the same time.
  - Collaboration::Directed Collaboration (optional)

    This pattern allows users to collaborate with others in the network interactively. It requires the two interacting users to be online simultaneously.
- Information Aggregation::User Information Access

  This pattern allows users to distill meaningful information from a vast amount of structured and unstructured data.
- Application Integration
  - Application Integration::Population

    This pattern structures the population of a data store with data that requires minimal transformation and restructuring.
  - Application Integration::Population=Multi Step (optional)

    This pattern structures the population of a data store with structured data that requires extensive reconciliation, transformation, and restructuring.
  - Application Integration::Population=Multi Step Gather

    This pattern provides a structure for applications that retrieve and parse documents and create an index of relevant documents that match a specified selection criteria.

### 4.3.2 Portal composite runtime patterns

There is a large number of Runtime patterns for the Portal composite pattern. However, they have been combined to form the Composite runtime pattern for portal shown in Figure 4-7.

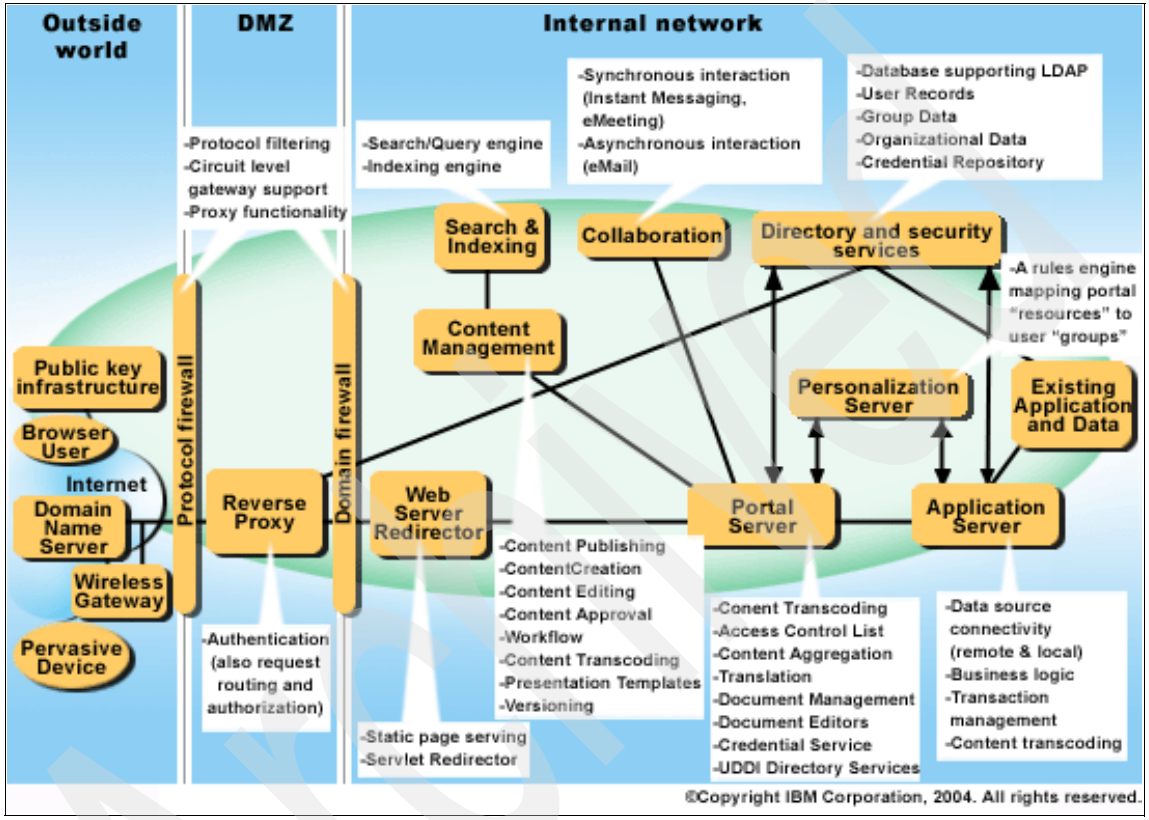



*Figure 4-7 Portal composite pattern::Functional mappings*

## 4.4 Account Access composite pattern

The Account Access solution provides clients with around-the-clock access to their account information. It also allows users to inquire, update, and delete information about their individual accounts.

The Account Access composite pattern does not have any Application patterns documented yet.

5

# Selected patterns for e-business on zSeries

In this chapter, we introduce the Application and Runtime patterns that we use to highlight the strengths of the zSeries platform in Part 2, “Methodology, design, and WBISF on z/OS” on page 81, and Part 3, “Preparing WBISF run time and Workload Manager on z/OS” on page 137.

We used the following pattern process to explain our approach in more detail:

1. Identify the Business pattern.
2. Select the Application pattern.
3. Select the Runtime pattern.
4. Map products to the Runtime pattern.

We describe some high function runtime nodes that are specific to the zSeries and provide the product mappings for the Runtime patterns.

## 5.1 Identification of the Self-Service business pattern

Businesses have traditionally invested a lot of resources into making information available to clients, vendors, and employees. These resources took the form of call centers, mailings, and so on. They have also maintained information about their clients in the form of client profiles. Updates to these profiles were handled over the phone or by mail.

The concept of self-service puts this information at the fingertips of the clients through a user interface, whether that interface is a Web site, a personal digital assistant (PDA), or some other client interface. An e-business application makes the information accessible to the right audience in an easy-to-access manner, reducing the need for human interaction and increasing user satisfaction.

Key elements of an application that provides self-service for a client include clear navigational directions, extended search capabilities, and useful links. A popular aspect is a direct link to the online representatives who can answer questions and offer a human interface if needed.

The following list provides examples of self-service applications:

- An insurance company makes policy information available to users and allows them to apply for a policy online.
- A mortgage company publishes information about its loan policies and load rates online. Clients can view their current mortgage information, change their payment options, or apply for a mortgage, all online.
- A bank allows customers to access their accounts and pay bills online.
- A well-known and respected group of technical writers makes its work available online. The group recruits technical participants for its projects by listing the upcoming projects online and allowing possible participants to apply online.
- A company allows its employees to view current human resource policies online. Employees can change their medical plan, tax withholding information, stock purchase plan, and so on online, without having to call the human resources office.

We mentioned only common usage areas for the Self-Service business pattern. Refer to 2.1, “Self-Service business pattern assets” on page 20, which describes them in general and identifies their variations.

## 5.2 Selection of the Decomposition application pattern

The Self-Service::Decomposition application pattern (Figure 5-1) provides intelligent routing from multiple channels to multiple back-end applications using a hub-and-spoke architecture and recomposing/decomposing capability. It provides the ability to decompose a user request into multiple requests to be routed to multiple back-end applications. The responses are recomposed into a single response for the user. This moves some of the business logic into the decomposition tier, but the primary business logic still resides in the back-end application tier.

From a service-oriented architecture (SOA) perspective, the decomposition tier of this Application pattern facilitates the invocation of business services hosted by a number of back-end applications. In doing so, the Decomposition application pattern fully leverages the integration capabilities described by the Application Integration::Broker pattern.

Consider a case where an interaction initiated by a user requires the execution of an end-to-end business process or workflow, where process and workflow rules are better externalized. The Decomposition application pattern would leverage the integration capabilities of more advanced process integration alternatives such as the Application Integration::Serial Process or Serial Workflow and the Application Integration::Parallel Process or Parallel Workflow.

Since the end result is the decomposing and recomposing capability, these variations are not documented as Decomposition application pattern variations. Instead, they are captured as different Runtime patterns where applicable.

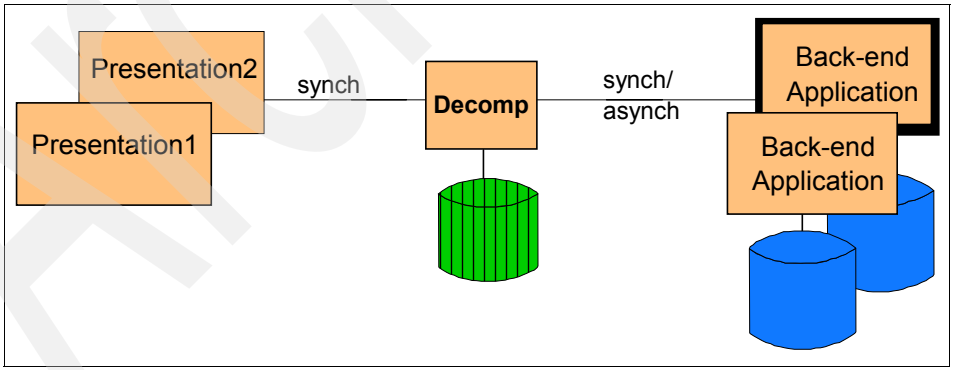



*Figure 5-1 Self-Service::Decomposition application pattern*

The Decomposition application pattern supports the following business and IT drivers to:

- Improve organizational efficiency
- Reduce the latency of business events
- Adapt easily during mergers and acquisitions
- Integrate across multiple delivery channels
- Unify client view across lines of businesses (LOB)
- Minimize total cost of ownership (TCO)
- Leverage existing skills
- Leverage legacy investment
- Integrate back-end applications
- Minimize enterprise complexity
- Be maintainable
- Be scalable

All business and IT drivers listed under the Router application pattern apply here as well. Additional drivers relate to the fact that many organizations have back-end applications that are focused on certain product lines. For example, insurance companies use different systems for supporting health insurance policies and life insurance policies. Such product-specific silos were necessary due to different business logic and data requirements for various products. For the same reason, many companies plan to continue to use separate systems for separate product lines.

These same companies, however, want to provide a unified customer view when customers visit the self-service Web sites or contact the call centers, rather than providing a fragmented, product-specific view. Similarly when changes are made to customer information in one system, they should be automatically reflected in other systems. In the example of the insurance company selling health insurance and life insurance policies, changes to address information should be automatically reflected in both the systems. Such features are increasingly important since customers often ask for a consolidated view of their multiple accounts.

As shown in Figure 5-1, this Application pattern is divided into three logical tiers: presentation, decomposition, and back-end application.

- The *presentation tier* is responsible for all the user interface-related logic, including data formatting and screen navigation. It can support many different presentation styles, including the Internet, call centers, kiosks, and voice recognition units.
- The *decomposition tier* supports most of the services provided by the router tier in the Router application pattern, including intelligent routing of requests, protocol conversion, security, and session concentration. In addition, it implements the intelligence to break down a single request received from a

presentation client into several, simpler requests that it routes to multiple back-end applications. It typically uses a local Work In Progress (WIP) database to store routing, decomposition, and recomposition rules and to cache the results from multiple back-end applications until a recomposition of the desired response has been generated. The decomposition tier implements significantly more business logic than a router tier. Such business logic focuses on providing a unified client-centric view.

**Router tier functions:** A router tier receives requests from multiple presentation components and intelligently routes them to the appropriate back-end transactions. In doing so, this tier may use a read-only database to look up routing rules. In addition, the router may be responsible for message transformation, protocol conversion, management of different levels of security, and session concentration. In most cases, the router tier implements minimal business logic. This routing capability can also be used for routing requests from one back-end system to another back-end system.

- The majority of the product and function-specific business logic is concentrated in the back-end application tier. Some of these back-end applications are highly available, and scalable online transaction processing systems and others are batch applications.

The requester interaction between the presentation and decomposition tier is synchronous. The requester interaction between the decomposition tier and the back-end application tier can be either synchronous or asynchronous.

A synchronous requester interaction is required when the presentation customer expects an immediate answer, as in the case with the insurance company and its customers. In the insurance company example, a customer logs on to the self-service Web site and asks to view a consolidated bill. This request is decomposed into multiple synchronous requests that are targeted towards multiple product-specific billing systems. The decomposition tier waits for responses from these systems and combines the results and displays a consolidated billing view to the customer.

An asynchronous requester interaction between the decomposition tier and back-end applications is appropriate when the presentation customer does not expect an immediate response. For example, consider the customer who initiates an electronic transfer of funds to pay for monthly bills using a self-service Web site. This request can be decomposed into two separate requests. The first request is targeted towards a confirmation engine that synchronously provides a confirmation number to the customer for tracking purposes. At the same time, an asynchronous request can be sent to a batch system that transmits an electronic

funds transfer request to a local bank using electronic data interchange (EDI) technology.

A variation on this pattern includes caching on the second logical tier to avoid a high volume of accesses to the back-end application. Another variation is to use fast asynchronous communications so that multiple parallel requests can be sent to the third tier to improve response times over serial requests.

### Considerations

There are a number of possible approaches to doing business request decomposition into multiple back-end transactions and recombining the responses into a single business response. These include:

- Roll-your-own (RYO) programming that issues multiple asynchronous requests to back-end systems and combines the responses
- Using a message broker plus a rules engine, possibly with a two-phase commit (2-PC) or compensations mechanism
- Using an enterprise service bus with broker capability

### Guidelines for use

To increase the flexibility of the solution and responsiveness to changing business requirements, we recommend that you pay particular attention to the definition of reusable messages/services that pass through the decomposition tier. Use robust transaction processing systems to implement the back-end applications to ensure availability, scalability, and performance.

A decomposition implementation (one source call to multiple target calls) requires state persistence and re-composition of the response messages. Use standards where possible to minimize future changes required to the source and target applications.

### Benefits

The benefits of this Application pattern are:

- It allows the integration of multiple, diverse applications.
- It minimizes the impact to existing applications.
- It provides routing services, relieving the source application from being aware of the target application.
- It provides transformation services that allow the source and target to use different communication protocols.
- It provides a holistic client-centric view rather than a fragmented product-centric view of information.

Executing transactions in batch mode, when appropriate, provides several benefits including the ability to free up systems resources for more important tasks at hand. The Decomposition application pattern allows you to distinguish those transactions and use asynchronous mode for communication under such circumstances. This is particularly true for updates that do not need to be reflected into the appropriate data stores immediately.

## Limitations

The focus of this Application pattern is providing a consolidated client-centric view of information. However, such information is gathered only when required and is discarded at the end of the transaction. Because of this, this Application pattern does not accumulate all the necessary information in an operational data store (ODS) that can be used for mass customization of services and for cross-selling purposes.

## Using the Application pattern

In the example of the insurance company that we mentioned earlier, the company wants to provide a unified customer view when customers visit the self-service Web sites or contact the call centers. It doesn't want to provide a fragmented, product-specific view. When changes are made to customer information in one system, they should be automatically reflected in other systems. Such features are increasingly important since customers often ask for a consolidated view of their multiple accounts. To achieve these goals, the company can apply the Decomposition application pattern.

## 5.3 Selection of the Runtime pattern

Runtime patterns describe how the logical components in the Application pattern are further broken down into more specific or detailed logical functions. The Runtime pattern also specifies the placement of these logical functions in the overall network structure.

We chose the Self-Service::Decomposition application pattern::Process Manager runtime pattern, shown in Figure 5-2, as the basis for our self-service application in this redbook.

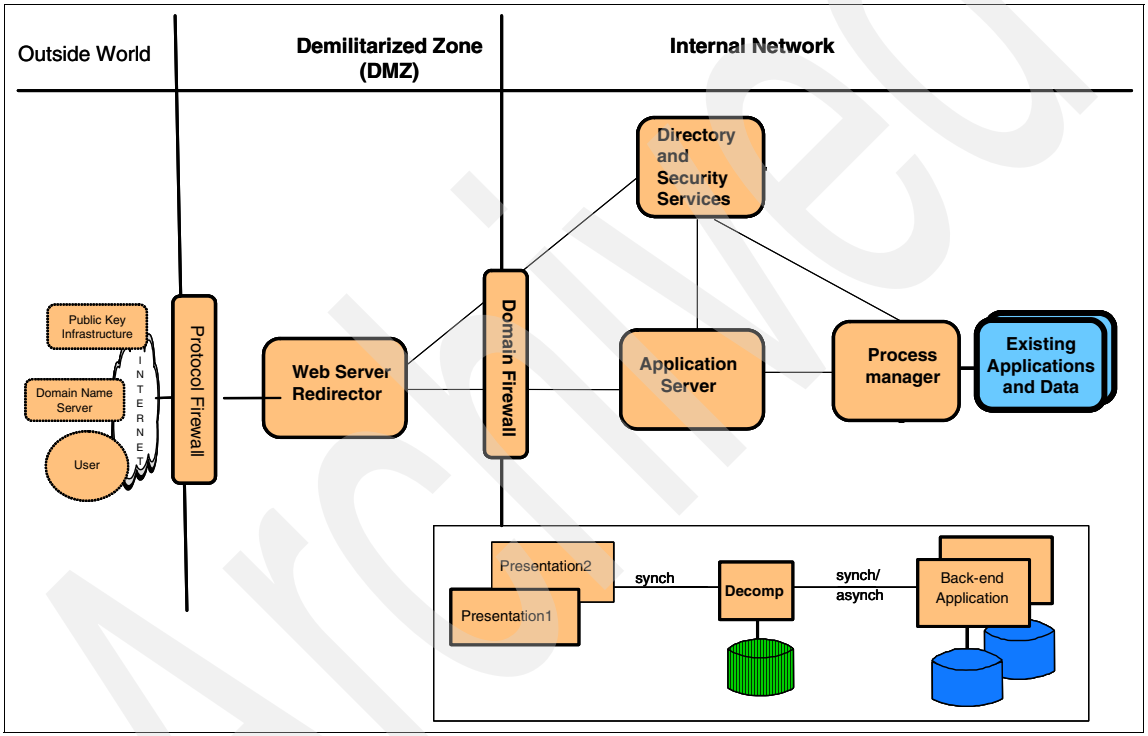



*Figure 5-2 Self-Service::Decomposition::Process Manager runtime pattern*

### 5.3.1 Runtime variations

Depending on nonfunctional requirements such as availability, performance, and security, several variants can be developed for a Runtime pattern. Figure 5-3 shows one variant, which is designed to provide increased availability and performance.

This variant provides increased performance by distributing the network traffic, Web content, and computational workload among the different runtime nodes.

- The Load balancer, Web server redirector, and the Workload distributor effectively distribute the network traffic.
- The Caching proxy nodes hold content that can be delivered to the client without the request having to go to the Web or application server.
- The Workload balancer distributes the workload between the application servers.

Each runtime node that has a “double outline” in Figure 5-3 represents multiple instances of that runtime node. Connected runtime nodes indicate cross coupling achieved by connections.

**Note:** Some functional nodes, for example the Workload balancer node, may only be available on some platforms as products off the shelf (though customization may be involved).

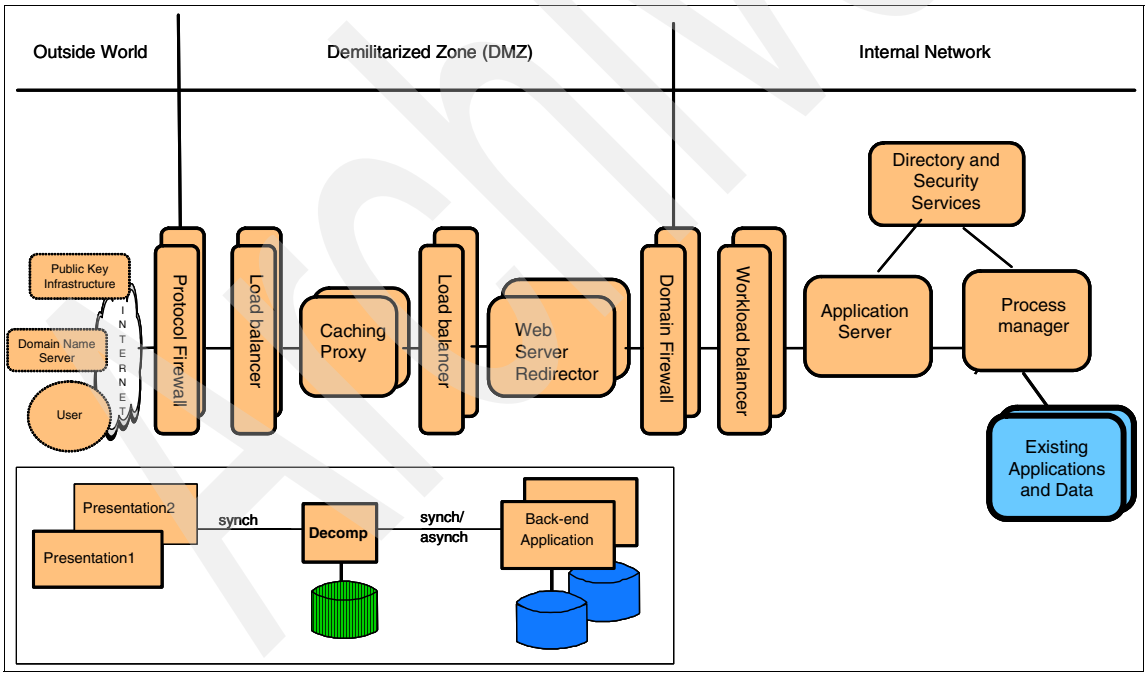



*Figure 5-3 Self_Service::Decomposition::Process Manager runtime pattern variant*

## 5.3.2 Decomposition runtime pattern in this redbook

The Runtime patterns we tested for this redbook use a simplified infrastructure. The differences stem from two simplifications:

- We accessed the application server from within the internal network.
- We did not put in place a separate Web Server Redirector node.

Figure 5-4 shows the simplified Runtime patterns used in this redbook.

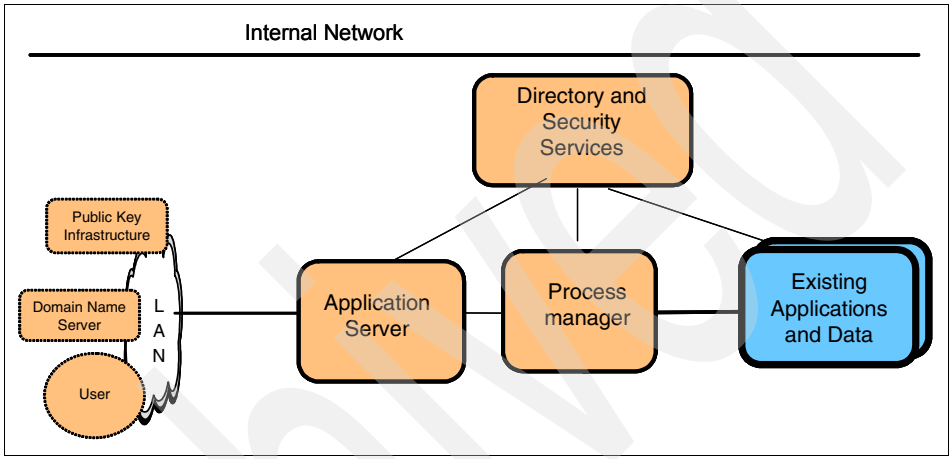



*Figure 5-4 Self-Service::Decomposition::Process Manager runtime pattern*

## 5.3.3 Runtime node descriptions

To map runtime nodes onto products, you must understand the logical function provided by the runtime nodes. We describe some of the common runtime nodes that are used in the Runtime patterns discussed in this book.

### Web application server node

The Web application server node is an application server that includes an HTTP server (also known as a Web server). It is typically designed for access by HTTP clients and to host both presentation and business logic.

The Web application server node is a functional extension of the informational (publishing-based) Web server. It provides the technology platform and contains the components to support access to both public and user-specific information by users employing Web browser technology. For the latter, the node provides robust services to allow users to communicate with shared applications and databases. In this way, it acts as an interface to business functions, such as banking, lending, and human resources (HR) systems.

The node can contain these data types:

- HTML text pages, images, multimedia content to be downloaded to the client browser
- Servlets, JavaServer™ Pages™ (JSP™)
- Enterprise beans
- Application program libraries, such as Java applets for dynamic download to client workstations.

### User node

The user node is most frequently a personal computing device (PC) supporting a commercial browser, such as Netscape Navigator and Internet Explorer. The browser is expected to support Secure Sockets Layer (SSL) and some level of DHTML. Increasingly, designers need to consider that this node might be a pervasive computing device, such as a PDA.

### Public key infrastructure

Public key infrastructure (PKI) is a system for verifying the authenticity of each party involved in an Internet transaction. It protects against fraud or sabotage. For nonrepudiation purposes, it helps consumers and retailers protect themselves against denial of transactions. Trusted third-party organizations called *certificate authorities* (CA) issue digital certificates, which are attachments to electronic messages that specify key components of the user's identity.

During an Internet transaction using signed and encrypted messages, the parties can verify that the other's certificate is signed by a trusted certificate authority before proceeding with the transaction. PKI can be embedded in software applications or offered as a service or a product. Leaders in On Demand Business agree that PKI is critical for transaction security and integrity, and the software industry is moving to adopt open standards for their use.

### Domain Name System node

The Domain Name System (DNS) node assists in determining the physical network address associated with the symbolic address (URL) of the requested information. The Domain Name Server node provides the technology platform to provide host-to-IP address mapping. It allows for the translation of names (referred to as URLs) into IP addresses and vice versa.

### Protocol firewall node

The protocol firewall node is a hardware and software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private

networks connected to the Internet, especially intranets, and can block some virus attacks, as long as those viruses are coming from the Internet. A firewall can separate two or more parts of a local network to control data exchange between departments. Components of firewalls include filters or screens, each of which controls the transmission of certain classes of traffic.

Firewalls provide the first line of defense for protecting private information. Comprehensive security systems combine firewalls with encryption and other complementary services, such as content filtering and intrusion detection.

Firewalls control access from a less trusted network to a more trusted network. Traditional implementations of firewall services include:

- Screening routers (the protocol firewall)
- Application gateways (the domain firewall)

A pair of firewall nodes provides increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is implemented as an IP router.

### Domain firewall node

The domain firewall node is implemented as a dedicated server node. See “Protocol firewall node” on page 75 for a description of firewalls.

### Directory and security services node

The directory and security services node supplies information about the location, capabilities, and attributes (including user ID and password pairs and certificates) of resources and users known to this Web application system. This node can supply information for various security services (authentication and authorization). It can also perform the actual security processing, for example, to verify certificates. The authentication in most current designs validates the access to the Web application server part of the Web server, but this node also authenticates for access to the database server.

### Existing applications and data node

Existing applications are run and maintained on nodes, which are installed in the internal network. These applications provide for business logic that uses data maintained in the internal network. The number and topology of these existing application and data nodes is dependent on the particular configuration used by the existing systems.

### Web server redirector node (Redirector)

The Web server redirector node separates the Web server from the application server, used in conjunction with a Web server. The Web server serves HTTP

pages, and the Redirector forwards servlet and JSP requests to the application servers. The advantage of using a Redirector is that you can move the application server behind the domain firewall into the secure network, where it is more protected than within the demilitarized zone (DMZ).

### Application server node

The application node provides the infrastructure for application logic and can be part of a Web application server. It is capable of running both presentation and business logic but generally does not serve HTTP requests. When used with a Web server redirector, the application server node can run both presentation and business logic. In other situations, it can be used for business logic only. See also “Web server redirector node (Redirector)” on page 76.

### Integration server node

The Integration server node is an interface between any front-end access channel, such as the Web, a call center, or a client/server (“fat client”) PC and any back-end application system that is needed (including applications from other companies). It performs the following kinds of services:

- Converts protocols from the front end to match what the back-end systems understand
- Decomposes a single message from the front end (such as a Web server) into several back-end messages (or transactions), and then recomposes the replies
- Navigates from the front end to the back-end system that needs to be accessed
- In more complex cases, controls the process or unit of work for a number of back-end interactions based on a request from the front end

This node relieves each front end from having to handle the complexity of interfacing with potentially multiple back-end systems, which may be in different companies. The front end, such as the Web server, should only need to send a message to the integration server and have it look after the interface.

A second purpose for locating these interface services on the Integration server concerns security. There is a firewall between the Web server and the integration server, and the Web server does not need to know all the back-end addresses. Many location do not want a server located in the DMZ to have access directly to sensitive data and systems. In this case, the Web server can only send messages to the integration server, and nowhere else.

### Process manager node

The process manager node contains the process flow execution engine. It provides the capability for model-driven business process automation. It also enables tracking by leveraging the process execution rules stored in the associated database.

These processes can span multiple applications and organizational boundaries within an enterprise. The node maintains state and tracks sequencing through the process flow. In doing so, it often leverages the associated repository to store intermediate results. Finally, it is responsible for invoking target applications as necessary through their associated connectors.

### Data server/services node

The data server/services node is a generic data storage node that provides managed, persistent storage of any type of data and a means to directly access and manipulate that data. Data may be stored in files and accessed through file input/output (I/O) routines or may be stored in a database with more structured and managed access methods.

### Workload balancer node

The workload balancer node is used to balance workload between clients and servers. It consists of several components found only on the zSeries, such as the zSeries Workload Manager and the zSeries Sysplex Distributor. The workload manager uses metrics from a workload advisor and a Quality of Service Policy Agent to decide to which server to send the work. The workload balancer has the ability to provide high availability of the IP applications running on server clusters even if one physical network interface fails.

### Workload advisor node

The workload advisor node provides information to the workload balancer. This information includes metrics such as whether a server is available, the CPU utilization of individual servers, the amount of I/O being done by a server, the amount of memory being used by a server, and so on. This information is generally gathered from real-time information provided by lower levels of the operating system. This node is currently only available on the zSeries platform.

### Quality of Service policy agent

The Quality of Service (QoS) policy agent provides many capabilities including traffic shaping to reduce low priority connections to “best effort” when high-priority connections are not meeting QoS objectives set by service-level agreements (SLAs). It also provides SSL encryption to protect from unauthorized entry through the Lightweight Directory Access Protocol (LDAP) interface, and

provides metrics for use by the workload balancer. This node is currently only available on the zSeries platform.

### Cryptographic node

The cryptographic node provides capabilities that significantly enhance the performance of runtime components that leverage security features such as SSL, virtual private network (VPN), and encrypted data use of CP Assist for cryptographic function. The cryptographic node provides capabilities to allow components to comply with security levels right up to FIPS 140-2 Level 4 compliance rating. This node is currently only available on the zSeries platform.

### Load balancer node

The load balancer node provides horizontal scalability by dispatching HTTP requests among several, identically configured Web server nodes.

## 5.4 Product mapping

The next step after choosing a Runtime pattern is to determine the products and platforms to be used. The platform selected should fit into the client's environment and ensure qualities of service such as availability and performance so that the solution can grow along with the On Demand Business.

We use the Self-Service::Decomposition::Process Manager runtime pattern mapped to the products as shown in Figure 5-5. The front-end application running on WebSphere Application Server invokes the business process instance implemented by the Process Manager node. The process can be invoked directly using the binding (SOAP, Java Message Service (JMS), and Enterprise JavaBeans™ (EJB™)) generated at deploy time. Or it can be invoked as a Web service using a proxy generated from the binding or through the API exposed over Internet Inter-ORB Protocol (IIOP) or JMS.

Depending on the binding, there is a session bean or a message-driven bean that clients can call for sending messages into the business process. The session bean provides a synchronous connection, and the message-driven bean provides an asynchronous connection.

The process manager node is implemented using the Process Choreography functionality in WebSphere Business Integration Server Foundation (WBISF) and Business Process Execution Language for Web services (BPEL4WS) technology. The Process Manager invokes the back-end applications which have been exposed as Web services. These services can be invoked synchronously or asynchronously. In our samples, the back-end applications are accessed through EJB exposed as enterprise services and running on WebSphere

Application Server. However it could be anything that could be exposed as a Web service.

The work-in-progress database is used by the Process Choreography to maintain state for business process instances and to manage work items for staff activity elements. Directory and Security services and the Repository run on products that are external to the Process Manager.

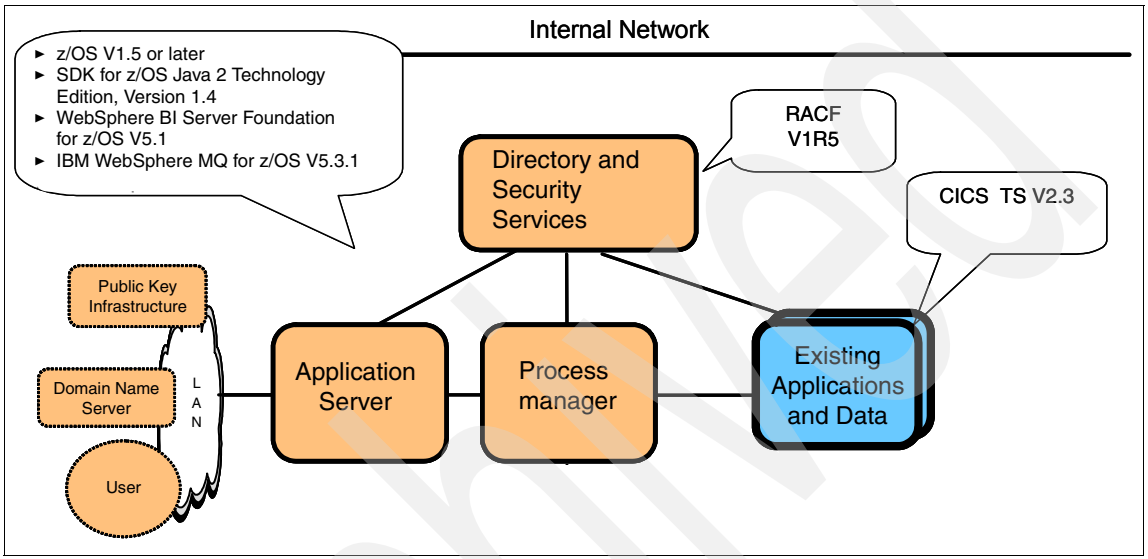



*Figure 5-5 Self-Service::Decomposition::Process Manager product mappings for zSeries*

# Part 2

# Methodology, design, and WBISF on z/OS

This part presents guidelines for applying the patterns approach to a sample business scenario and selecting technologies for self-service applications.

In Chapter 6, “Methodology considerations when designing business processes” on page 83, we explore business process modeling methodologies and describe the workflow and its dimensions and benefits. We also introduce the two IBM process engines, their interoperability, and the programming models of Process Choreographer.

Then in Chapter 7, “Design considerations” on page 95, we present several perspectives that you need to consider when planning and designing any type of process implementation.

And in Chapter 8, “Value of WBISF on z/OS” on page 117, we describe the WebSphere Business Integration Reference Architecture and the Programming Model Extensions of WebSphere Business Integration Server Foundation (WBISF). In this chapter, we also discuss the value of the zSeries platform.

6

# Methodology considerations when designing business processes

In this chapter, we explore classical and new methodologies to approach business process modeling, in consideration of its objectives. We also describe workflow, its dimensions, and its business benefits. In addition, we provide information about the interoperability of the two IBM process engines. As we concentrate on the new approach in this Redbook, we briefly describe the programming models of Process Choreographer.

## 6.1 The business process model

The challenges of enabling the attributes of an On Demand Business require fundamental changes in the way an enterprise defines and creates its business processes and supporting IT elements. Both of these aspects of business function design must work synergistically together to maximize their effectiveness in an on demand IT environment and culture.

The concept of components is the strategic element in this new business architecture. Although the component concept is not a new idea, components in this architecture are defined as a combination of business functions and data.

Allowing business analysts to create business applications from pre-built IT elements, and to modify them in real-time as needed is a step in the direction of the holy grail. The business process has become a key principle of the new organization style and the primary means by which the organization executes its strategy and delivers value to its clients. Almost equally important is the mechanism through which the organization gathers, analyzes, and conveys the information necessary for continuous process improvement. The underlying business drivers are to:

- Improve business agility
- Maximize reuse of business and application functions
- Consolidate overlapping infrastructures and applications
- Reduce implementation and maintenance costs

But what is a business process? Hammer and Champy in their book *Reengineering the Corporation: A Manifesto for Business Revolution* (hereafter cited in text), define a business process as “a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer”.[1]

Some people associate business processes with typical operations inside the companies. Such operations lead us to understand business processes as a sequence of activities performed by various persons to achieve an objective. These well-defined and repeatable activities can be modeled as processes, the *process model*.

A business process is a set of business-related activities that are invoked in a specific sequence to achieve a business goal. The business process defines a sequence of the flow, how external events are to handled, human interaction requirements, and conditional branching.

[1] Hammer, M. and Champy, J. *Reengineering the Corporation: A Manifesto for Business Revolution*. Collins, revised and updated edition January 2004. ISBN: 0-06055-953-5.

A business process-based application consists of the business process and the applications it invokes. A business process executes in a process engine and accesses business applications running on the application servers. This effectively separates the business flow logic from the implementation of each individual function.

A new development paradigm is created by using a visual interface for developing and maintaining the business flow logic and keeping the business rules separate from the implementation design and coding considerations.

The business process model is a combination of:

- Relevant business activities, IT resources, staff, applications, as well as data and data flows
- Connections of discrete activities with the processes including the rules to control the flow of work and data

According to Peter M. Senge, in his book *The Fifth Discipline: The Art and Practice of the Learning Organization*, the role and the responsibility of an IT architect in this context is the ability to see the big picture and to make decisions based on imperfect information and to understand the implications of such decisions.[2]

### 6.1.1 Business process modeling objectives

In the quest for total process integration, business must employ a complete business solution to identify, integrate and manage business-relevant data simply and seamlessly. For this to become a reality, the business departments must work closely with the IT experts to produce an optimal solution, one that meets the business requirements and enhances the process performance.

Historically the challenge has been to get IT and business professionals working together on this task. Developers and their business counterparts speak different languages, use different terminologies, and have different knowledge backgrounds. While business uses business process models (BPM), IT uses Unified Modeling Language (UML) models. Integration and transformation between the UML models and BPMs guarantee good communication, avoid costs associated with miscommunication, and reduce a project's duration. Other drivers are:

- Increased pressure to improve organizational productivity
- The need to leverage existing assets
- Increasing need to coordinate business processes with key business partners
- Higher time to market requirements

[2] Senge, Peter M. *The Fifth Discipline: The Art and Practice of the Learning Organization*. Doubleday Publishing, New York, first edition, September 1994. ISBN: 0-38526-095-4.

The objective of business modeling is to come up with a set of documents that describe the current operation of a company, the “AS-IS model”, and the intended operation of the company, the “TO-BE model”. Numerous methods are available to create such models: from simple paper and pencil, via Microsoft® Visio® or Microsoft PowerPoint®, to more complex computer-assisted modeling.

### 6.1.2 The methodology

Olle et. al. in the book *Information Systems Methodologies: A Framework for Understanding*, define an information systems methodology as “a methodical approach to information systems planning, analysis, design, construction and evolution”, and characterize different methodologies primarily by their “steps” and “components” (or deliverables).[3] The key idea of a methodology according to Simison, in “A methodology for Business Process Reengineering, IFIP Transactions”, is that it has some level of universality. It can be applied or adapted to a variety of business domains.

There are many approaches to identify and describe business processes. Among them are methodology and intuition, proposed by Davenport, in *Process Innovation: Reengineering Work Through Information Technology* (hereafter cited in text), and by Hammer and Champy in *Reengineering the Corporation: A Manifesto for Business Revolution*.[4] [5]

The main distinction is that intuition tells you “where to go”, while methodology tells you “what to do to get there”. For Simison, using a methodology is appealing for a number of reasons:

- Introduces a minimum level of discipline into the task
- Entails less effort and intellectual input than an intuitive approach
- Provides a means of codifying experience and ideas in a form which can be evaluated and tested
- Facilitates planning and monitoring
- Establishes clear definitions for each step and the deliverables
- Allows a standard set of skills to be identified and developed
- Is a prerequisite for the use of CASE tools

[3] Olle, T.W.; Hagelstein, J.; McDonald, I.G.; Rolland, C.; Sol, H.G.; Van Assche, F.J.M.; and Verrijn-Stuard, A.A. *Information Systems Methodologies: A Framework for Understanding*. Addison Wesley, 2nd edition, 1991. ISBN: 0-20154-443-1.
[4] Davenport, T. *Process Innovation: Reengineering Work Through Information Technology.* Harvard Business School Press, October 1992. ISBN 0-87584-366-2.
[5] Hammer and Champy, page 84

In “A methodology for Business Process Reengineering, IFIP Transactions”, Simison states that the advantages of using methodologies have been generally well recognized at the management level. However, implementation has been poor, for the following reasons:

- Most methodologies do not reflect a best practice. Some methodologies are based on the view of how systems should be developed, rather than successful practical experience.
- Common methodologies for information systems conflict with practice in the assumption that developers will work from scratch, treating each business situation as though it were unique in all respects.
- There is a potential risk that only those tasks that are readily incorporated are considered in a methodology and others are ignored.
- The focus can shift from designing a solution to just peripheral tasks, such as documentation of the current system, because they are easier to manage.
- Creativity is stifled when attempting to break creative stages down into smaller tasks, imposing rules or wasting resources that are not essential to meeting the overall objective.

### 6.1.3 Movement from the AS-IS process to the TO-BE process

Before you create new processes, you need to clarify which methodology you want to follow. Both of the following methods provide sound reasoning to help you choose that particular approach.

- Hammer and Champy, in *Reengineering the Corporation: A Manifesto for Business Revolution*, believe that to redesign a business process, the “TO-BE” process, understanding of the existing process is crucial.[6] In particular, you need to know the following information about the existing process, the “AS-IS” process:
  - What it does
  - How well it performs
  - Critical issues that govern its performance

  Hammer and Champy argue that the business analysis team’s goal is not to improve the existing process. It does not need to analyze and document the process to expose all of its details. Their observation suggests that the most frequently committed error in re-engineering is that the re-engineering team tries to analyze a process in detail rather than attempting to just understand it.

[6] Hammer and Champy, page 84

- Davenport's opinion, in *Process Innovation: Reengineering Work Through Information Technology*, contrasts with this approach.[7] He suggests to take the time to understand an existing process to establish the benefit of adopting the new process. The reasons are:
  - Understanding existing processes facilitates communication among participants in the re-engineering initiative.
  - Documentation provides the means to migrate to a new process. It is useful for understanding the magnitude of the anticipated change and the tasks required to move from the "AS-IS" process into the "TO-BE" process.
  - Recognizing problems in an existing process can avoid repeating them.
  - Understanding the current process provides a measure of the value of the proposed redesign.

### 6.1.4 The new development paradigm

The new approach to define processes is based on components. The traditional approach was based on procedures or objects and data. It differs in the explicit consideration of who performs the activities that make up the process. The classic process definition technique focuses only on the procedural aspects of the process and is described solely by its input and output.

First you define the process requirements, and then you define the activities that are necessary to meet the requirements. Since the requirements are set when the process is defined, allowances are seldom made for additional requirements negotiated during the execution of the process itself. Therefore, conditions of statisfaction are usually fixed. They are established during the definition of the process rather than during the performance of the process itself.

For example, measuring customer satisfaction is much easier. In the classic approach, customer satisfaction is measured using post-sales surveys, making it difficult to identify root causes of dissatisfaction when they occur. Using the new methodology, measuring customer satisfaction is easy. It is effectively built-in and determined on the transaction-by-transaction basis as part of the assessment phase.

When processes are defined around procedures and activities, the resulting process structure is often arbitrary, more strongly reflecting organizational history and interpersonal relationships than the actual work of the process. It is rare for two different views of the same set of activities to result in the same grouping and relationships.

[7] Davenport, page 86

### 6.1.5 The IBM process modeling methodology

Several BPM tools and development methodologies are offered by various vendors. IBM offers the WebSphere Business Integration Workbench (WBI Workbench) that follows this approach:

1. Determine the modeling objective.
2. Define the process boundaries.
3. Identify the key process issues.
4. Set the process performance targets.
5. Outline the data collection plan.
6. Plan for the modeling phase.
7. Develop the modeling policies.
8. Determine the modeling approach.

WebSphere Business Integration Modeler is a state-of-the-art product used to define, model, analyze, simulate, and report business processes. Processes modeled in WebSphere Business Integration Modeler can be exported to WebSphere Studio Application Developer Integration Edition (WSAD-IE) for implementation and deployment.

Business processes can also be modeled in the WSAD-IE environment. Processes modeled in this way can be deployed directly into IBM WebSphere Business Integration Server Foundation (WBISF), a business process engine.

## 6.2 Workflow model

Workflow technology helps to implement business processes that can be easily adapted to the changing needs of a dynamic environment. These business processes can be within an enterprise (inter) and between enterprises (intra-enterprise). Workflow supports heterogeneous and distributed environments.

### 6.2.1 Workflow and its three dimensions

A business process may be defined as a set of related tasks to achieve a defined business outcome. A business process may involve applications, components, and people that perform multiple activities, using diverse information sources. The workflow manager provides the responsible person or application with the relevant activity and the necessary data at the right point in the business process and across and beyond the enterprise.

Defining enterprise-wide processes involves identifying actions and activities of the process, the people, or means that perform the activities and the resources

used. A workflow resource manager must support the definition of processes through three independent elements, also called *dimensions*:

- *Process logic* is the “what”, which describes the activities to be performed and their sequence
- *Organizational structure* is the “who”, which describes the organizational structure of the company in terms of departments, managers, roles and users.
- *Applications, components, and IT infrastructure* determine the “how”, which are used to perform the activity.

Application integration workflow is typically of short duration (seconds to minutes) with routing based on state data and events. This is called a *short-running process*. People-oriented workflow is typically of longer duration (minutes to weeks) involving the integration of applications as well as people and in general provides a user interface. This is called a *long-running process*.

Traditionally workflows are divided into four categories according to their business value and repetitive qualities, as you can see in Figure 6-1.

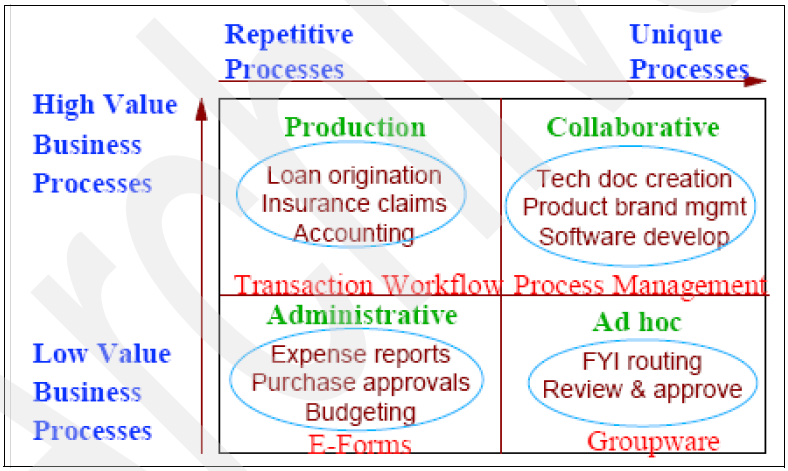



*Figure 6-1 The four Workflow categories*

**Note:** Highly repetitive processes are the best candidates for workflow automation.

A workflow model is a complete representation of one or more business processes. It is a comprised of all the relevant business activities, IT resources, staff, applications, as well as the data and data flows.

### 6.2.2 Business benefits of workflows

The business benefits of using workflows are:

- Separation of what needs to be done (process logic) from how it is implemented (applications and components)
- Enabling rapid delivery of new products and services that leverage existing and new investments in infrastructure
- Enabling system designers to make changes without negatively impacting the business
- Business process-driven needs
- Providing reliable and correct processing of business functions (robustness) by providing:
  - A restart and recovery event for catastrophic failures
  - Exception management for automating frequently executed processes, including the ability to manage exceptions manually
  - Based on robust communication and security capabilities
- Integration across the enterprise and into B2B transactions with partners by:
  - Separation and synchronization of external and internal processes
  - Integrating multiple trading partners within a single process
- Reuse of existing application logic through connectors

## 6.3 The IBM business process engines

IBM offers two major process engines on z/OS:

- IBM WebSphere MQ Workflow
- Process Choreographer within IBM WebSphere Business Integration Server Foundation

IBM WebSphere MQ Workflow is based on the traditional methodology, on components. Process Choreographer is based on the new approach, which includes procedures, objects, and data. The following section provides information about interoperability of the two engines. As we concentrate on the new approach in this redbook, we introduce the programming model of the Process Choreographer, also called *Business Process Engine* (BPE).

### 6.3.1 Workflow engines and their interoperability

Workflows that are developed for both business process managers, IBM WebSphere MQ Workflow and Process Choreographer, offer the advantage of both products. They also allow a smooth and staged migration from WebSphere MQ Workflow to Process Choreographer.

Why is interoperability between WebSphere MQ Workflow and Process Choreographer important? The following items answer this question:

- Usage of WebSphere Business Integration Monitor

  We can use WebSphere MQ Workflow monitoring tools with Process Choreographer if we model Process Choreographer processes as subprocesses in a WebSphere MQ Workflow master process.

- As a migration path for coexistence of both products

  To reuse the existing WebSphere MQ Workflow process in Process Choreographer, we can model them as subprocesses inside new Process Choreographer processes. This way we can avoid migrating WebSphere MQ Workflow processes into the Process Choreographer base.

- Reduction of the number of transactions needed by a business process

  If we have a business process with several activities as part of a non-interruptible process and if interacts frequently with Java applications, Enterprise JavaBeans (EJB) or Web services, we can take advantage of a better built-time support in WebSphere Studio Application Developer Integration Edition for modeling these interactions, transactional support for EJB invocations, and transactional support for Web services invocations using EJB or Java Message Service (JMS) bindings.

- Support for CICS EXCI and WebSphere MQ bridge
- Support for Atomic and Compensation Spheres in a WebSphere MQ Workflow if we model Process Choreographer processes as subprocesses in a WebSphere MQ Workflow master process

### 6.3.2 Process Choreographer programming model

The programming model offered by the Process Choreographer in WebSphere Business Integration Server Foundation can be described in four interacting models, which are illustrated in Figure 6-2.

- The information model

  In business enterprises, this model comprises business objects that model such entities as client accounts or purchase orders. These objects may reside in Enterprise Information Systems (EIS) or database systems. In the Java 2 Platform, Enterprise Edition (J2EE™) programming model, they are modeled into message types or entities such as EJB.

- A services model

  This model involves the programming necessary for business functions as services. These services can be implemented as business methods of a session EJB, Java bean methods, or JMS operations. The Process Choreographer capabilities expose these business service methods of J2EE components into operations and messages in a service-oriented architecture described in open standards such as Web Services Description Language (WSDL).

- An organization model

  This model deals with people in enterprises who are organized into groups such as departments. People can be organized based on lines of business, lines of products, locations, and business roles. For business applications, you can organize people based on roles in running and accessing services and processes. End users and business partners can be assigned roles for accessing business services and processes offered by an enterprise. Organization models are typically implemented through some form of user registries or directories.

- The composition model or business process model

  This model involves choreographing business services into a business process. It provides standard interfaces to a business process for end users and business partners. If the process models involve human activities (staff activities), the activities are assigned to people based on organizational roles or groups in user registries.

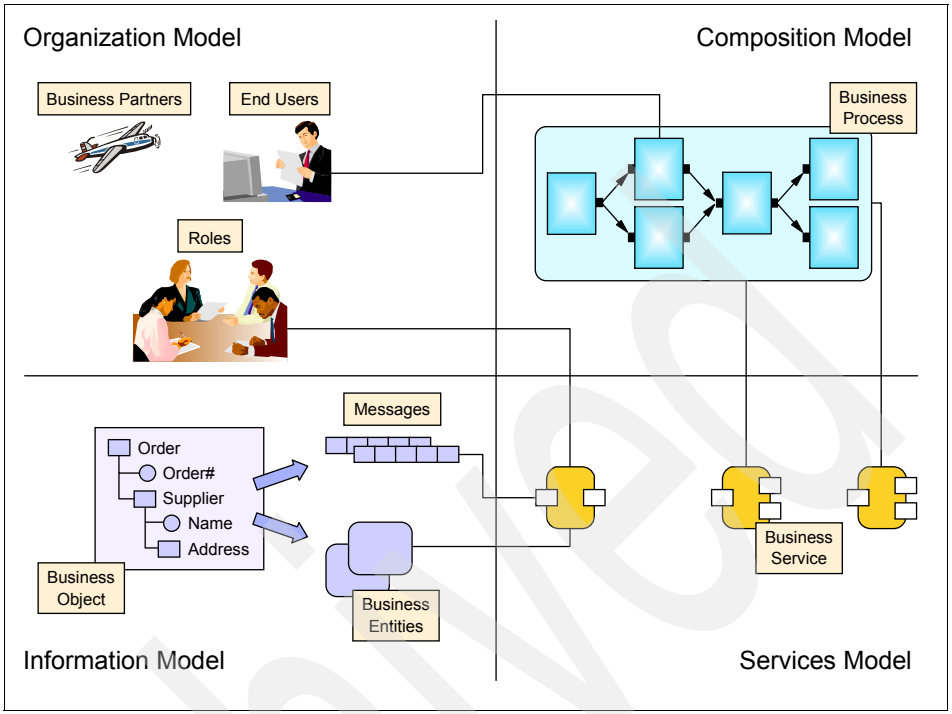



*Figure 6-2 Conceptual view of business process-based programming model*

7

# Design considerations

This chapter discusses several perspectives that you need to consider when planning and designing any type of process implementation, in particular:

- Non-interruptible versus interruptible processes
- Synchronous versus asynchronous processes (process invocation method)
- Sequences or flows
- Workload balancing
- Human interaction and staff resolution
- Handling exception situations
- Process security
- Business process monitoring
- Performance and process analysis
- Audit trail

## 7.1 Non-interruptible versus interruptible processes

Processes may be *interruptible* or *non-interruptible*. As the names suggest, interruptible processes can be suspended and resumed, while non-interruptible processes stay active from the time they start to the time they complete.

**Note:** Interruptible processes are also called *long running processes* or *macroflows*. Non-interruptible processes are called *short running processes* or *microflows*.

The key differences between the two modes of operation are:

- State and status persistence
- Transactionality
- Asynchronous processing

### 7.1.1 Interruptible processes

A process must be interruptible if it contains asynchronous activities or runs for an extended period of time. This enables system resources to be released for other processes that are active in the process container. It also allows the process container to be shutdown and restarted without losing the state and status of the process.

A business process becomes interruptible if each step of the process is processed within its own physical transaction. Classic workflow systems have provided support for interruptible processes for quite a while.

Interruptible processes contain multiple transactions with each activity running in its own transaction. They persist their state and status to disk between activities. With this persistence, the process is forward-recoverable. When the process is suspended, execution can continue from the suspended state when the proper event or a time-out occurs.

### 7.1.2 Non-interruptible processes

Non-interruptible processes are short running and manage state and status in memory. All activities in a non-interruptible process run within a single transaction. All those activities run in one physical thread from start to end without interruptions.

A non-interruptible process cannot contain anything that needs a commit in the middle of an activity, such as:

- Send/receive a Java Message Service (JMS) message
- Wait for an event
- Interact with a person

Non-interruptible processes are small in footprint and fast in execution. They are suitable for high throughput applications that cannot tolerate the overhead associated with storing the state to disk and for operations that do not need to be suspended.

A global transaction that runs under the two-phase commit protocol ensures that all transactional activities on a non-interruptible process either succeed or are rolled back. A non-interruptible process can have different transactional capabilities:

- It can run within a distributed transaction.
- It can run as part of an activity session.
- It may not use transactions at all.

**Important:** As long as any activity within the non-interruptible process might become executable, the transaction manager does not end the corresponding global transaction. As a consequence, an interruptible process should run fast. Otherwise, the corresponding global transaction lasts for a long time, reducing the overall throughput of the environment due to resource locking mechanisms.

If you request to collectively undo a set of activities in a non-interruptible process, you have to use compensation. Compensation is possible in two ways:

- To invoke the compensable activities, that are associated with undo actions, in reverse order of the forward navigation
- To invoke the compensable activities that are associated with the business process itself

### 7.1.3 Compensation

Compensation support in WebSphere is integrated with transaction management. The WebSphere transaction log is used to store information required to undo activities, such as the name of the undo operation and the associated data. If a transaction is rolled back, then compensation is run as part of the rollback processing of the transaction manager.

When using the Process Choreographer as process engine, compensation properties for business processes and related activities can be defined in

WebSphere Studio. The sphere of compensation spans all sub-processes of the overall process.

The objective of compensation processing is to revert to a consistent state and to any operations that were committed up to when an error occurred. The compensation processing starts as a result of that error, during the run of a process instance for which compensation is defined in the process model.

In interruptible processes, the compensation is triggered if either of the following actions occur:

- The overall process instance ends with a business error.
- The process instance was explicitly terminated with a request for compensation.

In non-interruptible processes, the compensation is triggered on rollback of the work unit (the transaction or the activity session) that contains the process. Therefore, compensable activities are typically specified for activities that cannot be reversed by rolling back the unit of work. Depending on the outcome of the unit of work (rollback or commit), compensation starts.

**Important:** Only Web service activities can be compensated. Java and Enterprise JavaBeans (EJBs) need to be wrapped in a Web service.

The compensation stack is global, which means that there is one compensation stack for both the parent process and related subprocesses.

**Attention:** For compensation to work, undo operations have to be specified for each activity, and the process itself needs to be enabled for compensation.

### 7.1.4 Transactionality

In the WebSphere Business Integration Server Foundation perspective, a transaction can be viewed as an activity between two or more parties that must be completed in its entirety with mutually agreed outcome. Transactionality enables multiple application operations to be coordinated to provide an atomic deterministic end result.

Resource managers are used to control access to the resources involved in a transaction. A transaction manager is responsible for coordination and transaction control. WebSphere Business Integration Server Foundation relies on WebSphere Application Server transaction support.

WebSphere Application Server on z/OS permits the usage of a single one-phase commit capable resource with any Resource Recovery Services (RRS)-capable resources in the same transaction.

RRS connectors provide a z/OS Quality of Service (QoS) enhancement, so that high performance native z/OS componentry is used to provide two phase commit functionality. When deploying the application on z/OS, connection factories configured for a given application generally perform better when you specify the RRS capable connector. This is known as a “Local OS” or “binding mode” connector.

The z/OS platform architecture provides the capability to transactionally coordinate some types of resource managers in z/OS resource sub-system. That is, CICS and IMS™ can connect to a local WebSphere Application Server for z/OS by connectors.

## 7.2 Synchronous versus asynchronous processes

In Process Choreographer, a process can be called in two ways:

- Synchronously

  A synchronous process is invoked by a request/response operation, and the result of the process is returned to the caller immediately via this operation. A response is not returned from the process engine until the process completes. However, the completion of the process does not mean that it completed successfully. The reply needs to be inspected to determine the success or failure of the process.

- Asynchronously

  An asynchronous process is invoked by a one-way operation, and the result and any faults are returned by invoking other one-way operations. The process result is returned to the caller via a callback operation. An asynchronous invocation of a process receives an immediate response, but this response only indicates whether the process instance was started successfully. Depending on the process engine, a response may or may not be sent back to the user at the completion of the process.

If a process is invoked synchronously, a response is expected and the process is stopped. An asynchronous process is one way. For example, think of a synchronous process as a telephone, and an asynchronous process as the postal system. When you have a conversation on the phone, you send and receive messages instantaneously using the same connection. If you were to send the same message in a letter via the postal service, it would be delivered in one manner, and its response returned in another.

The characteristics of the two process invocation methods determine which method to choose for a given process template.

- Choose a *synchronous invocation* if:
  - The process is non-interruptible.
  - The calling application will receive a response in a reasonable amount of time.
  - The user who invoked the process expects to wait for a response.
- Choose an *asynchronous invocation* if:
  - The process is interruptible.
  - Human interaction is required within the process.

**Tip:** In general, if a process is likely to take more than a few seconds to complete, it should be called asynchronously.

Exceptions to these guidelines are:

- Subprocesses

  A process that is spawned from another process is called a *subprocess*. Even though the subprocess may be interruptible (long running, possibly even taking months to complete), it is quite common to call this subprocess synchronously to ensure that the higher level process does not continue until the subprocess completes.
- Calls to another process container

  A process may contain an activity that invokes a process running in a second process container. This call might also be synchronous, regardless of whether the process running in the second process container is interruptible. Using a synchronous call ensures that the initiating process does not continue until the invoked process has completed.

## 7.3 Sequences or flows

Process Choreographer provides support for flow and sequence structures which allow nesting. Processes can include combinations of sequences and flows contained within one another.

- **Flow**: Activities within a flow execute concurrently. The addition of links and their associate conditions provides additional control over the order of execution. The end of flow activity is a synchronization point; all activities must complete before the process continues.
- **Sequence**: Activities in a sequence start sequentially. The order of execution is defined by the position within the block.

Either a sequence or a flow can be used as a starting point for development. The process has exactly the same characteristics and capabilities. The decision is largely a matter of personal preference. Nevertheless there are design differences to consider. For example, parallel processing is only available in flows.

## 7.4 Workload balancing

The goal of workload balancing is to ensure that incoming work requests are distributed to the resources and environment (application servers, enterprise beans, servlets, and so on) that can most effectively process the requests. On the z/OS platform, workload balancing is a core capability of the operating system using a base component of Workload Manager (WLM) for z/OS.

Typical workload balancing situations include:

- When a user sends an HTTP request from a Web-based client
- When a request is sent to the enterprise bean
- Between instances within one application server when an application programming interface (API) is sent to the message-driven beans API
- When an internal message is received by the application process engine

**Tip:** Dispatching Process Choreographer activities to an arbitrary server is a good workload-balancing practice.

Chapter 10, “Workload management on z/OS” on page 165, provides more information about workload balancing topologies.

## 7.5 Human interaction and staff resolution

In its current specification, Business Process Execution Language (BPEL) does not allow the definition of human-based activities nor the assignment of humans responsible for such a business process. IBM defined BPEL+ extensions to support both needs.

Staff support allows the dynamic assignment of responsibilities based on existing organizational definitions. Work lists can be created to let the designated recipients know that their action is required. From the business process point of view, a *staff-activity* (human-based activity) is a step in the process with the associated Web service not implemented by a program but by an action performed by a human.

**Restriction:** Process Choreographer supports business process with human interaction *only* when security is enabled in WebSphere Application Server. This allows authentication to take place and work items for a specific user to be determined.

When a staff activity is processed, work items are created and distributed to all persons who are eligible to perform this activity. Eventually, one of these users decides to work on the activity. When the user starts that activity, the process engine provides the user with unique access to it. The user can:

- Read the input data
- Perform any necessary actions
- Create output messages with the result of the activity or a fault

Creating the output message or a fault completes the activity and causes navigation to a forward activity in the business process.

Syntactically, staff-activities based on BPEL+ are treated as special invoke activities. In particular, staff-activities have a signature defined by a port type and an operation. As a normal function, they read input and produce output or a fault. However, staff-activities do not refer to a *partner link*. Instead they define a set of potential owners (staff-assignment). Users who participate in business processes receive work items for activities that can be from different business processes.

**Note:** BPEL+ uses elements from the staff support of WebSphere to define staff assignment verbs (a set of potential owners).

### 7.5.1 Web Client and Staff Support Service

The WebSphere Business Integration Server Foundation out-of-the-box interface for interacting with work items and activities is the Web Client. The Web Client is based on JavaServer Pages (JSP) technology and is structured like a portal. It is commonly called the *process portal*.

The Web Client provides users a set of work lists that contain work items, on which they can run queries, view details, or perform certain actions, for example:

- Query all the work items that are currently assigned
- Apply filters on the type of activity or its current state
- Sort work items by priority or age
- Start and stop work items

The work lists are the process portal central place.

**Tip:** The Web Client can be customized and adapted for any particular business need. It allows the inclusion of user-defined JSPs that support staff activities with complex data or interaction scenarios.

The Web Client interacts with the process engine via the API. It does not interact directly with other components that support human interaction, as shown in Figure 7-1.

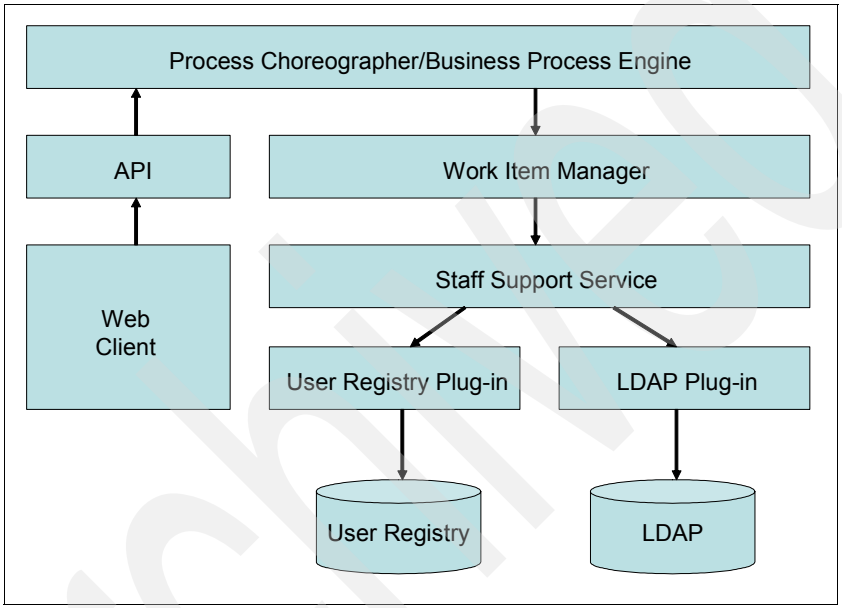



*Figure 7-1 Main components involved in a staff resolution*

The Work Item Manager (WIM) is the component that is responsible for invoking staff resolution, which is performed by the Staff Support Service staff resolution plug-ins.

The Staff Support Service is a master plug-in that:

- Manages the life cycle of the specialized staff resolution plug-ins
- Transforms the staff query verbs into plug-in specific queries
- Delegates deployment and staff resolution requests to these specialized plug-ins

If a business process includes a staff activity, execution of the process stops until the activity has been completed by an individual or the activity times out.

## 7.5.2 Delegation and permission setting

When human interaction is introduced into a process, the process manager needs to determine who the eligible users are for the activity instance. If possible, this determination of eligible users should be dynamic at run time. This late-binding allows the process manager to apply the most current staff definitions of an organization.

There are several methods to delegate work items to a manager:

- Delegate to a manager role within a certain organization
- Delegate to the manager of the user who started the process
- Delegate to a specific user that is identified at the start of the process
- Delegate to the coordinator of a role

Permissions that allow individuals to complete an activity are defined through a combination of verbs, parameters, and roles.

### Verbs

A verb specifies the type of query to be made against the security system. A comprehensive set of predefined staff verbs is available that define abstract definitions used in assigning specific work items. Here are some examples of supplied staff verbs:

- Users (full distinguished name)
- Users by user ID (uid attribute)
- Group members (full distinguished name)
- Manager of employee

### Parameters

Parameters provide input for the query. For example, if the verb for the query is “Users by user ID”, the parameter could be the user ID.

One approach is to completely specify the information at modeling time with, for example, a distinguished name such as `cn=George Washington, ou=Retail, o=ITSOElec, c=US`. This is called *early binding* in a Process Choreographer context.

As an alternative, you can use expressions within the parameters of your chosen staff verb, for example `%wf:process.starter%` within the `users` verb. This is called *late binding*.

Alternatively, you can define expressions that get evaluated at run time. For example, you can specify “users by user ID” as the staff verb, but instead of entering a uid, you can enter an expression such as:

- `%wf:process.starter%`
- `%wf:process.administrators%`
- `%wf:activity(-enter activity name-).potentialOwners%`
- `%wf:activity(-enter activity name-).readers%`

It is also possible to use variable parts for staff query parameters, allowing you to set, for example, a value for a user by a user ID verb from the output of a process variable.

You parameterize *staff verb sets* by adding arguments for a staff activity. The corresponding *verb set mapping file* translates the parameterized staff verb set into an executable query. This executable query is input for the staff support service that produces the concrete executable LDAP query.

## Roles

Table 7-1 shows the work item actions that are authorized for each role in the Process Choreographer Web Client. This refers to roles that are defined for staff activities within a process.

*Table 7-1 Work item actions and roles*

| Action | Business process administrator | Potential owner | Owner | Reader | Editor |
|---|---|---|---|---|---|
| Claim work item | Yes | See note 1 | See note 2 | | |
| Transfer work item (5.1 and later) | Yes | | Yes | | |
| Save work item | Yes | Yes | Yes | | Yes |
| Display work item | Yes | Yes | Yes | Yes | Yes |
| Complete work item | Yes | | Yes | | |
| Force restart work item | Yes | | | | |
| Force complete stopped work item | Yes | | | | |
| Create work item (5.1 and later) | Yes | | | | |
| Delete work item (5.1 and later) | Yes | | | | |

**Note:**

1. If a potential owner claims a work item, the potential owner becomes the owner of the work item.
2. An owner of a work item can transfer the work item only to a user that is a potential owner of the associated activity.

Processes can also have roles associated with them to limit who can perform certain actions with a process instance. See Table 7-2.

*Table 7-2 Process actions and roles*

| Action | Business process administrator | Process administrator | Starter | Potential owner | Reader |
|---|---|---|---|---|---|
| Start process | Yes | | See note | Yes | |
| Terminate process | Yes | Yes | | | |
| Monitor process (5.0.2 and later) | Yes | Yes | | | Yes |
| Delete process | Yes | Yes | | | |
| Display process | Yes | Yes | Yes | | Yes |
| | | | | | |
| Save user-defined lists (5.0.2 and later) | Yes | | | | |
| Force complete stopped activity | Yes | Yes | | | |
| Force restart activity | Yes | Yes | | | |
| | | | | | |
| Claim work item | Yes | Yes | | | |
| Save work item | Yes | Yes | | | |
| Display work item | Yes | Yes | | | |
| Complete work item | Yes | Yes | | | |
| Transfer work item (5.1 and later) | Yes | Yes | | | |
| Create work item (5.1 and later) | Yes | Yes | | | |
| Delete work item (5.1 and later) | Yes | Yes | | | |

**Note:** If a potential owner of a starting activity starts a process, this person becomes the *starter* of the process. If a potential owner claims a work item, the potential owner becomes the *owner* of the work item.

The starter of the process instance automatically becomes the process administrator of the process instance if no one else is specified. That is why you can terminate or delete a process instance even if you are not explicitly defined as the process administrator.

### 7.5.3 Expiration of staff requests

To avoid having processes permanently suspended waiting on human activity, make sure to set an expiration for the activity. This allows the process to continue if no one completes the activity.

If the duration set by the expiration parameters is exceeded, the staff activity state will be set to STATE_EXPIRED. This value can be checked in the control links from the output terminal of the staff activity, allowing you to handle the expiration condition. The process can handle the expiration, or if the control link is wired to a fault activity, compensation can take place.

## 7.6 Handling exception situations

There are several methods for handling exception situations, such as a failure in the process, an invoked service, or a timeout condition. We discuss these four techniques:

- Ending the process
- Using faults
- Using compensation
- Using human involvement

### 7.6.1 Ending the process

This approach suggests that, if an error is returned by any of the activities in the process, the process will end immediately and the error be returned to the user. If any data has been updated before the error or a timeout occurred, some level of compensation should be applied before the process ends.

With this approach, the process must be defined to support having the process end at each activity in the process if a failing condition is encountered. Every activity must be allowed to go to the end of the process if the activity does not complete successfully.

### 7.6.2 Using faults

Faults can occur when a process instance is created or when operations that are invoked as part of the navigation of a process instance fail. Some mechanisms exist to handle these faults, such as:

- Passing control to the corresponding fault handlers
- Passing the fault to the client application as an API exception

  For example, an exception is thrown when the process model from which an instance is to be created does not exist.

These approaches handle exception situations using fault processing, most likely combined with compensation logic to rollback committed changes. Faults in a BPEL process are equivalent to exceptions in Java programs. Similar to Java programs, faults can be thrown during process execution. These faults can be thrown from within activities, from within a process, or by the process itself. Faults can be caught by fault handlers inside processes where they can be handled, or re-thrown.

Failures that occur during a process may be either business related or programming related, for example:

- Business: A credit risk check fails.
- Programming: A service fails.

Faults allow you to handle both types of failure in the business process, decoupling the implementation from the business process logic. For example, an exception occurs in an activity implementation that updates a client record. To handle this in the process:

1. Catch the failure in the activity's implementation logic.
2. Throw an exception.
3. Catch the exception as a fault ("Update Failed") and handle it in the process's fault handler.

**Important:** All specific exceptions inherit from a generic ProcessException. It is a *best practice* to catch generic exceptions with a final `catch(ProcessException)` statement. This statement helps to ensure the upward compatibility of your application program because it takes account of all other exceptions that can occur.

### 7.6.3 Using compensation

When a process fails before completion, a technique is needed to reverse the completed activities. Compensations are operations that logically “undo” operations performed by activities. Compensation is available for both Interruptible and non-interruptible processes.

Each activity in a business process has a primary operation that is invoked during run time. For each of these activities, you have the option to define a compensation operation to be run that will undo the actions of the primary service. You can find more details in “Compensation” on page 97.

### 7.6.4 Using human involvement

When you have an interruptible process, timeouts and conditions can be used to ensure that only successfully completed automatic system activities allow the process to move forward in the business process. If any exceptions occur (that is, if an automated activity returns an error), the process can be routed to the appropriate support personnel to handle. Typically, this pattern of exception handling is applied to all automatic system activities within a workflow.

Depending on the process manager used, timeouts or non-responses from automatic system activities can be handled in one of two ways:

- A *notification* can be placed on each automatic system activity. If an automated system activity does not respond in a specified period of time, an activity notification is generated. The modeler builds the process template in such a way that notifications for automated system activities are sent to the work list of a support person. Notification is not available with Process Choreographer.
- An *expiration* can be placed on each automatic system activity. If a response is not received in the specified period of time, the activity expires and the process continues down a control connector path that puts a work item on a system administrator’s work list to investigate. After the problem is corrected, the administrator retries the automatic system activity by completing the work item and having the process manager take control.

The difference between notification and activity expiration is that the original request to the automatic system activity is still valid with a notification and that activity waits until it receives a response. In the expiration, the original (timed out) request to the automatic activity is deemed disposable when the activity expires. The request is eventually re-sent after the system administrator investigates the problem.

## 7.7 Process security

For workflow-based applications, security refers to authentication of users, of user requests that are accessing methods of a business process, and of staff resolution queries to authorize users to work with an activity in a business process instance. Staff queries are executed at run time to determine which persons are candidates for working on a particular staff activity and need an instance-based authorization to run.

You must consider the following levels of security when working with processes:

- Process related security

  This level of security defines who can start, edit, and view process instances. Authenticated users can be assigned roles such as process starter and process editor. Also authorization checks can be made against these roles whenever a user attempts to work with a process instance.

  Delegation is a process security identity propagation from a caller to a called object. According to the J2EE specification, a servlet and enterprise beans can propagate either the client (remote user) identity when invoking enterprise beans, or they can use another specified identity as indicated in the corresponding deployment descriptor.

- Activity-related security

  Typically whenever automatic system integration is used, the components or services that are invoked require some level of authentication to ensure that the requests are from a valid source. Additionally, external event activities and work items may require certain authorization privileges.

When designing a process, take into account both of these security considerations, and define the roles accordingly.

We use LOCAL-OS (Resource Access Control Facility (RACF®)) in WebSphere to authenticate client requests. When the Web-Client (business process client) connects to WebSphere Application Server, the application server enforces authentication of the client.

The credentials entered by the client are checked against the entries in RACF (native authentication). If authentication succeeds, the client requested is authenticated. The Business Process Engine (BPE) stores both the caller principal name and the security context used to initiate the process as part of the persistent state of the business process instance. The security contexts can also be passed to subsequent activities within the process.

However, some existing services may not allow the typical user to perform the type of operation requested in the process. In this case, the user ID is not

acceptable to authenticate the request. Situations like this require that some sort of system user ID be used and passed as part of the process flow to each service.

> **Important:** To avoid manual synchronization of staff managers of the processes and activities, use the same repository for staff queries and WebSphere Application Server authentication. To support such queries as “is manager of”, you need to use an enhanced security repository that enables you to create and maintain those kinds of schemes. The z/OS Security Server LDAP Server component is an example of such a security repository.

## 7.8 Business process monitoring

The business process instance monitor presents the current or accumulated states and other related information of a particular process instance, for example:

- The average amount of time it took to process an activity
- The minimum and maximum amount of work items per activity
- Staff resolution

Monitoring business processes provides the following advantages:

- The ability to map the IT resources used by various business processes, and monitor and control all tiers of an end-to-end solution
- To protect systems from intrusions and threats by using monitor and alert systems to allow automated self-protection of the IT environment
- To allow establishment of business service-level agreements (SLAs) and automated detection and remediation of violations via monitor and alert systems
- To monitor and analyze performance data
- To monitor for exceptions in a business process, for example, if the process does not complete within a time limit

In WebSphere Business Integration Server Foundation, the process instances can be monitored using the Business Process Web Client. On any of the process instance list pages, such as *Administered By Me*, select the process instance that you want to monitor and then click **Monitor**. Optionally, you can click an activity to see more information about it.

## 7.9 Performance and process analysis

A business process that runs within one WebSphere Application Server is considered to have server affinity. Affinity to a WebSphere Application Server means affinity to a JVM™ process. If a business process engine can assume affinity, then the execution of business processes can be optimized by using advanced caching strategies to reduce the number of database operations.

The BPE is a J2EE application. It directly benefits from the performance characteristics of enterprise beans, servlets, Java Database Connectivity (JDBC™), and JMS connections.

The throughput and response time of workflow-based applications depend on several factors, such as:

- The type of business process used

  The navigation costs of a noninterruptible business process are much cheaper compared to interruptible processes.

- The structure of the business process such as:
  - The number of activities
  - The number of loops
  - The amount of parallelism
  - The performance of services invoked by activities

- The performance of components used by the BPE

  Database and messaging systems performance has an immediate impact on the performance of workflow-based applications.

> **Important:** The performance of the underlying database and message queuing systems has a direct impact on the performance of interruptible business processes. Provide hints to the BPE on how it can optimize transaction behavior of interruptible business processes.

By default, each activity in an interruptible business process runs within its own physical transaction. If we provide a transaction hint for an activity, the engine can select more flexible transaction boundaries, like combine multiple activity invocations into one single transaction. This simple hint reduces the navigation effort, resulting in a better response time and system throughput.

The disadvantages of specifying a transaction hint are:

- Increased recovery costs as more work needs to be undone in an event of an error
- Reduced concurrency on a recovery process; larger transactions require resource locks to be held longer

**Note:** If you specify transaction hints, be aware of the consequences of these design decisions for processes in Process Choreographer.

You also can assign priorities within business process templates to improve response time of some business processes at the expense of others. BPE uses the priority as a hint to decide which business processes it should prefer during navigation. Priorities can be overridden at run time for the individual business process instance. You can establish priorities based on data associated with the business process like (income > 100000) or by an administrative action. For more details about workload management, see 7.4, “Workload balancing” on page 101.

Another way to increase performance is to reduce the number of instructions required for navigation by compiling non-interruptible business processes. Reducing the number of instructions reduces the involvement of the process engine, in some cases even to the point where only one JVM is needed.

**Note:** Compiling an interruptible business process is less important than compiling an non-interruptible business process. This is because the number of instructions required for navigation is small compared with the overall amount of instructions consumed.

## 7.10 Audit trail

All important events in the life of a process or an activity can and should be recorded as an entry in audit trail. Use audit trail records to:

- Audit or track the history of a process or activity
- Gather specific historical information of a business process as a valuable source of information to reengineer the business process itself

For example, using historical information, business process analysts could review business constraints and bottlenecks in the current model (AS-IS model) and propose business solutions by creating a new business process scenario (TO-BE model).

In WebSphere Business Integration Server Foundation, the audit trail is enabled when you select the Business Relevant check box on the server tab for the process or for any activity that you deem worth auditing, as shown in Figure 7-2.

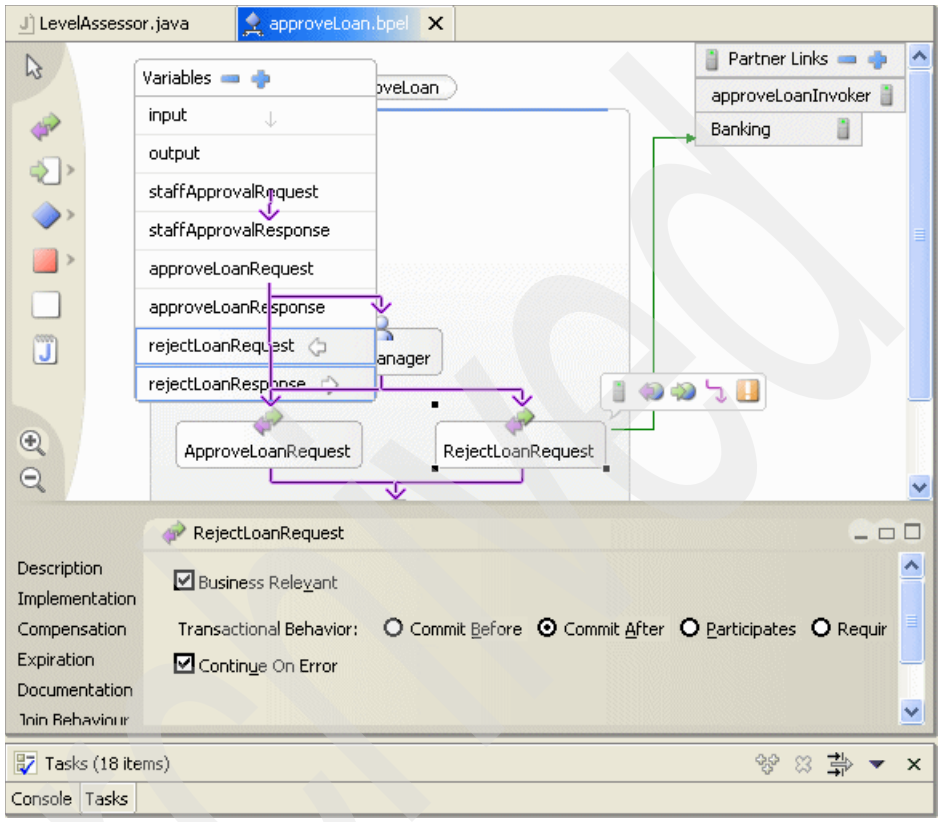



*Figure 7-2 Activating the audit trail log records for an activity*

The audit entries are stored in the BPE container database and can be accessed via JDBC or standard SQL calls.

Physically, process audit information is persistently written in a BPE container database to a defined table *Audit_log_T*. Process Choreographer provides two database views to show audit log information.

- AUDIT_LOG contains audit logs for both V5.0 style processes and processes based on the BPEL.
- AUDIT_LOG_B contains audit logs for BPEL based processes only.

Entries in the audit trail contain various information, for example:

- A time stamp when the event took place
- An event code that uniquely characterizes the event, such as a start of an activity or the finish of a process
- An identifier of the process for which the event is written
- A requestor of the action
- An identifier of the activity

Audit events (event codes) are related to process entities. The audit event types depend on the entity to which the event refers. Audit event types include:

- **PIE**: Process instance events
- **AIE**: Activity instance events
- **VAR**: Events related to variables
- **CLE**: Control link events
- **PTE**: Process template events
- **SIE**: Scope-related events

These events are applicable to BPEL processes only.

8

# Value of WBISF on z/OS

WebSphere Business Integration Server Foundation (WBISF) V5.1 on z/OS provides the same functionality as WBISF V5.1 on distributed platforms. It is implemented using WebSphere Application Server for z/OS V5.1 and takes advantage of the z/OS operating system. It provides a powerful standards-based integration platform for deploying composite applications within a service-oriented architecture (SOA).

In this chapter, we describe the WebSphere Business Integration Reference Architecture (WBI Reference Architecture), the Programming Model Extensions (PME) of WBISF, and the value of the zSeries platform. This chapter is based on the IBM Redbook *WebSphere Business Integration Server Foundation V5.1 for z/OS*, SG34-6382.

## 8.1 WebSphere Business Integration Reference Architecture

In every day business, we experience enterprises with a set of different platforms, operating systems, and proprietary applications (developed in-house or acquired from various software vendors). The challenge in such an environment is to combine all of these existing assets into one, integrated solution.

To participate in such a heterogeneous environment, software vendors are driven provide solutions that adhere to such open standards as J2EE, XML, and Web services. These standards provide the technology to create a business process consisting of different applications available in an enterprise.

This concept is called *service-oriented architecture*. It enables flexible connectivity of applications or resources by:

- Representing every application or resource as a service with a standardized interface
- Enabling them to exchange structured information (messages, documents, business objects)
- Mediating the message exchange through an Enterprise Service Bus (ESB)

This flexibility enables new and existing applications to be easily and quickly combined to address changing business needs. The ability to easily choreograph applications allows IT services to more readily reflect business processes.

The WBI Reference Architecture (See Figure 8-1) is an extended version of the SOA. The WBI Reference Architecture covers all aspects of today's enterprises IT centers. This includes application development, integration of existing applications, access to existing data and ERP systems, a run-timeinfrastructure and different methods to expose information to end-user.

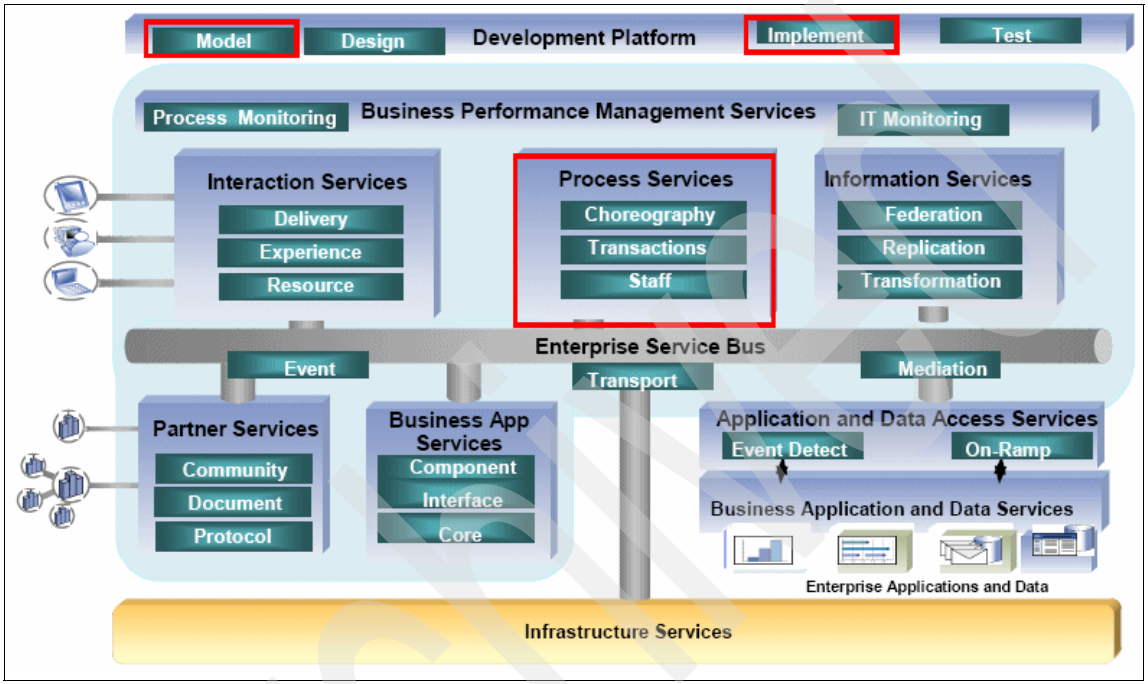



*Figure 8-1 Concept of WebSphere Business Integration Reference Architecture*

IBM enables all of the WBI Reference Architecture's products and components to comply with these standards. This allows IBM to offer solutions, shown in Figure 8-2, that are based on the WBI Reference Architecture.

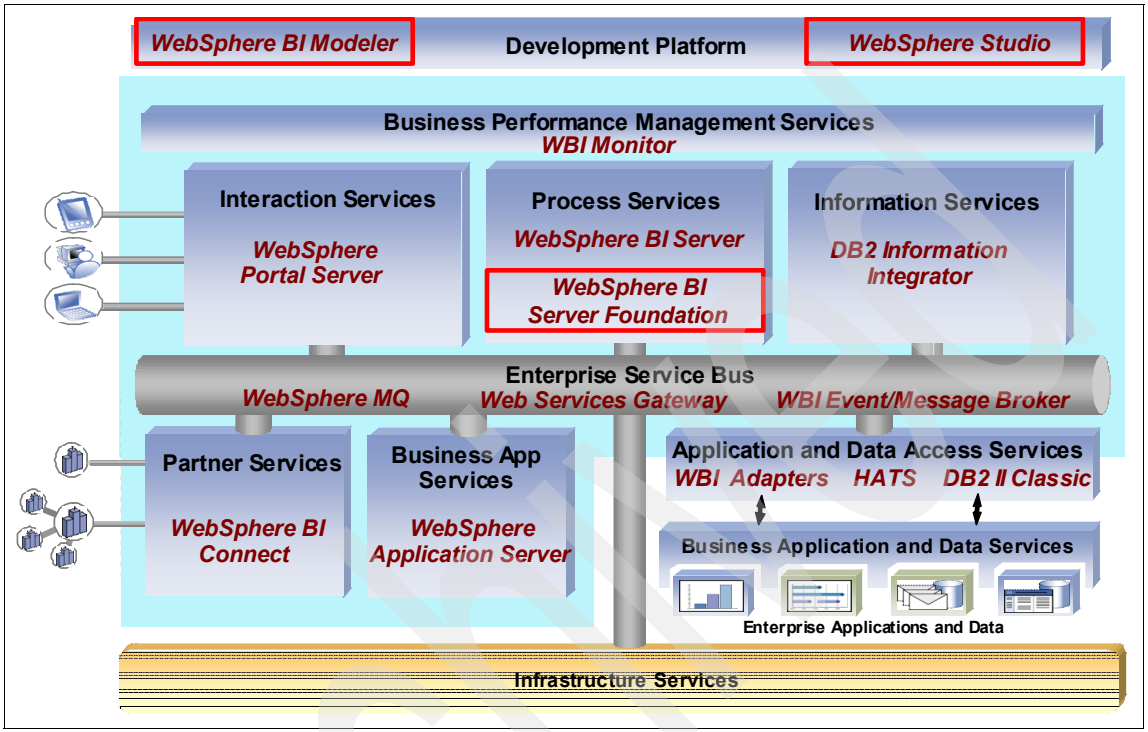



*Figure 8-2 IBM Software offerings for WebSphere Business Integration Reference Architecture*

The white paper *WebSphere Business Integration Server Foundation V5.1, Building and Deploying Service-Oriented Applications That Extend and Integrate Existing IT Assets*, published by D.H. Brown Associates, Inc., highlights the benefits of SOA and how WBISF uses it. You can download this paper from the Web at:

http://www-306.ibm.com/software/integration/wbisf/library/rbwp.html

## 8.2 Programming Model Extensions overview

WBISF has implemented native deployment support for Business Process Execution Language (BPEL). Business processes within WBISF are defined with BPEL for Web services (BPEL4WS) and can be created in two ways:

- The business analyst defines a new business process using WBI Modeler.
- The application developer uses WebSphere Studio Application Developer Integrated Edition V5.1 (WebSphere Studio) to develop a new process.

The IBM Redbook *WebSphere Business Integration Server Foundation V5.1 Handbook*, SG24-6318, covers the application development with WebSphere Studio and WBI Modeler in detail. We focus on the run time of WBISF V5.1 for z/OS in this redbook.

The Programming Model Extensions include:

- **Asynchronous beans**: Enable J2EE applications to decompose operations into parallel tasks in order to speed performance
- **Startup beans**: Enable J2EE applications to execute business logic automatically, whenever an application starts or stops normally
- **Last participant support**: Can provide automated coordination for transactions that include two-phase commit resources and a single one-phase commit resource
- **Internationalization service**: Can allow customers to build applications that can automatically adjust to handle global audiences
- **Work areas**: Provide the ability to efficiently share information across a distributed application
- **Scheduler service**: Enables tasks to be executed at a requested time.

  When used in conjunction with asynchronous beans, it enables batch processing applications within J2EE.
- **Activity session services**: Provide the ability to extend the scope of and group multiple local transactions
- **Dynamic query service**: Provides the ability to pass in and execute SQL query statements at run time
- **WSGW filters**: Can allow a customer to write filters for the Web Services Gateway such as filters that select a target service and port, capture Web service invocation information, or handle exceptions
- **Object pools**: Enable an application to avoid creating new Java objects repeatedly

- **Container Managed Messaging**: Can offer automated support for outbound (as well as inbound) messaging
- **Distributed map**: Can offer an interface to enable J2EE applications and system components to cache and share Java objects by storing a reference to the object in the cache in order to help improve performance
- **Container Managed Persistence over anything**: Can extend the existing J2EE Container Managed Persistence (CMP) framework to support any back-end system or service that supports create, retrieve, update, and delete methods

WBISF for z/OS V5.1 and WebSphere Studio for Linux® and Windows V5.1 can help to:

- Accelerate large-scale application development
- Integrate J2EE based workflow
- Advance transactional connectivity
- Optimize application performance

WBISF for z/OS V5.1 and WebSphere Studio for Linux and Windows V5.1 allow you to leverage some of the latest innovations that build on today's J2EE standards, such as PMEs. The following sections explain in more detail how PMEs do this.

### 8.2.1 Integrated J2EE-based workflow

When a developer creates Web services from an organization's software assets, the Web services can be used as parts of a business process. Integrated J2EE workflow capabilities offer developers intuitive, flow-based development tools, to quickly define how existing software assets are used within a J2EE-based application.

For example, the visual workflow development tools, called *Business Process Choreographer*, can be used to combine inventory information from a packaged ERP application and J2EE components from a previously built client-facing application with new business logic to create a new Web-based order entry application. The reach of the application can then be extended by exposing it as a Web service for use by business partners or to allow manual intervention for exception handling. The result is faster development of new applications, improved consistency, and lower costs through the reuse of existing components protecting IT investment.

The major components of workflow are:

- Business Process Choreographer

  This component provides intuitive drag-and-drop tools to easily compose and choreograph application interactions and dynamic workflow among J2EE components, Web services, existing applications, and human activities. Developers can quickly and easily build, debug, and deploy complex applications using powerful workflow tools and advanced messaging capabilities to streamline and automate business processes. New services can be added, or existing services modified, without affecting other components in the process.

- Human interaction

  This component offers support for activities that require a person to perform a task as a step in an automated business process. Specialized staff support allows the dynamic assignment of responsibilities based on existing organizational definitions. Lists can be created to let the designated recipient know that their action is required. Human interaction is an IBM Extension and is not part of the BPEL 1.1 specifications today.

- Event triggering

  Event triggering offers support for including asynchronous events such as Web services to be included as part of a business process. You can use event triggering to start a business process. Or you can use it to configure a business process to stop and wait for an external event to occur before resuming the process.

- Compensation pairs

  This component provides transaction *rollback*, such as support for long running, loosely-coupled business processes that cannot be undone automatically by the application server. For example, the compensating transaction for an order that has already started manufacturing might be to put the complete item into inventory rather than disassembling the item. Compensation pairs allow you to visually define, for each step in your business process, the associated *undo* service.

- Flexible workflow design

  Flexible workflow design provides developers with the ability to design workflow using a top-down, bottom-up, or meet-in-the-middle approach. Using a top-down approach, developers can create skeleton processes that choreograph the sequence of events in a workflow without worrying about the underlying implementation. Building from the bottom-up, developers first create the individual components and then use them as building blocks to define a workflow. Meet-in-the-middle offers the flexibility of using both approaches at the same time.

### 8.2.2 Advanced transactional connectivity

Since its inception, the J2EE platform has made tremendous strides in providing enterprise level support for integration including support for messaging, security, and database access. The Java Connector Architecture (JCA) 1.0 standard begins to offer support for integrating with packaged and legacy applications. However, due to lack of adherence to data standards and limited transactional support, integrating with most back-end resources and legacy data can still be extremely difficult and risky.

WBISF V5.1 and WebSphere Studio V5.1 offer advanced transactional capabilities to help developers avoid custom coding. They provide support for the many challenges related to integrating existing software assets with a J2EE environment, such as:

- Dynamic application adapter support

  This support offers the ability to build and deploy rich, open standards-based application adapters for popular enterprise information systems such as SAP and IBM Customer Information Control System (CICS).

- Last participant support

  This support provides automated coordination for transactions that include two-phase commit resources and a single one-phase commit resource. This support eliminates hand coding in this scenario. It also allows you to include one-phase commit resources, common for many existing and package applications, in real transactions.

- Activity session services

  These services provide the ability to extend the scope and group of multiple local transactions. These local transactions can then be committed based on deployment criteria or through explicit program logic. This ability reduces the complexity of dealing with commitment rules and limitations associated with one-phase commit resources.

### 8.2.3 Accelerated large-scale application development

Companies today strive to respond with flexibility and speed to client demands, market opportunities, and external threats. However, for most companies, the time, cost, and complexity of large-scale application development make this goal extremely difficult to achieve.

WBISF V5.1 and WebSphere Studio V5.1 leverage the latest innovations that build on today's J2EE standards to help you deploy a high performance e-business infrastructure designed to cut costs, build client loyalties, promote business agility, and gain a competitive advantage.

## Enablement of ‘next generation’ development

Ironically, J2EE’s main advantage, its specification, can also be its biggest disadvantage for developers building applications that require them to have more control over their applications than the J2EE specification provides. For those developers, WBISF V5.1 and WebSphere Studio V5.1 enable “next generation” development. They leverage the latest innovations that build on today’s J2EE standards to provide greater control over application development, execution, and performance than ever possible before.

- Asynchronous beans

  These beans offer exceptional performance enhancements for resource intensive tasks by enabling a single request to be executed as multiple tasks or threads processed in parallel within the J2EE environment. Asynchronous scheduling facilities can also be used to process parallel processing requests in batch mode at a designated time.

- Startup beans

  These beans allow business logic to be automatically executed when an application starts or stops. For example, they might be used to pre-fill application specific caches, initialize application level connection pools, or perform other application specific initialization and termination procedures.

- Scheduler service

  This service provides the ability to process workloads using parallel processing, set specific transactions as high priority, and schedule less time sensitive tasks to process during low traffic off hours. This helps to minimize IT costs and increase application speed and responsiveness by maximizing utilization of existing computing resources.

- Object pools

  These pools allow an application to obtain an instance of a Java object and return the instance to the pool when it has finished using it. This increases application performance by allowing instances of objects to be reused, reducing the overhead associated with instantiating the objects and garbage collecting them.

## Increased development productivity

The time required to roll out new applications is a key concern across all industries. One way to vastly improve developer productivity is to reduce the need for handcrafted solutions that can be time-consuming, costly, and difficult to maintain. WBISF V5.1 and WebSphere Studio V5.1 were designed to improve developer productivity by leveraging the latest innovations that build on today’s J2EE standards to provide supported, pre-built solutions to many of these challenges.

- Extended messaging

  Extended messaging allows you to quickly create applications that integrate with other systems through a messaging infrastructure. This extended messaging capability offers automated support for outbound (as well as inbound) messaging, allowing you to focus on business logic instead of complex messaging APIs. Handcrafted Java Message Service (JMS) code is no longer required. WebSphere Business Integration Server Foundation V5.1 includes WebSphere MQ and WebSphere MQ Event Broker to further extend your messaging infrastructure to take advantage of both products' qualities of service and to enable seamless integration with existing MQ infrastructures.

- Internationalization service

  This service allows you automatically recognize the calling client's time zone and location information so your application can act appropriately. This technology allows you to deliver to each user, around the world, the right date and time information, the appropriate currencies and languages, and the correct date and decimal formats.

- Work areas

  Work areas provide a "global variable" like ability to efficiently share information across a distributed application. For example, you might want to add profile information as each client enters your application. By placing this information in a work area via numerous application interfaces, it will be available throughout your distributed application and eliminate the need to hand-code a solution or to read and write information to a database.

- Cheat sheets

  Cheat sheets make new or complex tasks easy by providing a checklist for common development patterns. The cheat sheet invokes each step in the checklist and provides detailed online help for each step, as you need it.

- Best-in-class integrated development environment (IDE)

  Best-in-class IDE, which is included with WebSphere Studio V5.1, is a fully integrated application development environment for creating and maintaining J2EE applications and Web services. Built on Eclipse V2 innovations and written to J2EE specifications, WebSphere Studio V5.1 helps optimize and simplify J2EE application development with best practices, visual tools, templates, code generation, and the most comprehensive development environment in its class.

### Enablement of real-time application flexibility

Maintaining competitive advantage in today's changing business environment requires companies to respond quickly to client demands, market opportunities, and external threats. Often this means making frequent updates to e-business applications to reflect changes in market conditions or to provide access to

strategic information. Unfortunately, these updates usually take a great deal of time: time to bring down the application, time to make programming changes, time to test the new application, and time to redeploy it.

To enable real-time application flexibility, WBISF V5.1 and WebSphere Studio V5.1 offer dynamic application support, such as:

- Business rule beans

  These beans offer a powerful real-time framework for defining, executing, and managing business rules that encapsulate business policies that vary based on changes in the business environment. For example, a simple business rule might be, “If a client's shopping cart is greater than $X, then offer a Y% discount.” Once the business rule is defined, a developer or a business analyst can update the business rule at run time using a straightforward user interface without the need to bring the application or server down.

- Dynamic query service

  This service delivers application flexibility by allowing you to dynamically build and submit queries that select, sort, join, and perform calculations on application data at run time. Dynamic query service provides the ability to pass in and process Enterprise JavaBeans Query Language (EJB QL) queries at run time. This eliminates the need, as with today’s EJB 2.0 standards, to hard-code required queries into the deployment descriptors during development.

### 8.2.4 Optimized application performance

Increasingly, organizations are using Web applications both internally and externally to incorporate clients, partners, and suppliers into their business processes. For these important processes, application performance can make the difference between competitive advantage and failure to compete.

WBISF V5.1 and WebSphere Studio V5.1 help you optimize performance and minimize downtime for applications that require highly available, high volume, multi-server environments. They do this through powerful application profiling techniques, sophisticated deployment management, and advanced support for Web services.

- Application profiling

  Application profiling delivers powerful new capabilities that allow you to carefully optimize the performance of applications without any impact on source code. This capability offers a mechanism for specifying the access intent of persistent entity Jabs, allowing them to interact with the run-timeinfrastructure, such as a database, differently depending on the access intent (for example read versus update) of the application that calls it.

The result is unprecedented control in defining strategies that dynamically control concurrency, pre-fetch and read-ahead.

- Deployment manager

  This manager addresses the needs of highly available, high volume, multi-server environments through enhanced workload management and dynamic caching, centralized security capabilities, and performance management tools that distribute the workload across multiple servers through sophisticated load balancing and clustering capabilities. The deployment manager also enables isolation of application servers to avoid single points of failure and provides first failure data capture to report and analyze problems as they occur.

- Advanced Web services support

  This support offers advanced support for Web services, including a Universal Description, Discovery, and Integration (UDDI) Registry that acts as a repository that allows storage of business units. The business units describe basic Web services and a Web Services Gateway that enables Web services invocation by users from outside the firewall with the benefit of robust security protection. Advanced Web services support also extends the Web Services Gateway by providing a programming model that allows you to use the gateway in large-scale Web services implementations. It serves as a bi-directional control point for critical tasks such as validation, logging, transformation, auditing, and metering.

## 8.3 The zSeries value proposition

The zSeries servers are powered by the z/OS operating system. z/OS is the foundation for the future of the zSeries, an integral part of the z/Architecture™ designed and developed to quickly respond to the demanding quality of service requirements for enterprise e-business.

z/OS provides high levels of availability, scalability, and manageability for important workloads, with such features as Workload Manager and IBM Parallel Sysplex® support. It also provides the qualities of service needed to meet the demands of global On Demand Business.

The next sections go into more detail about the value of zSeries.

### 8.3.1 Base system hardware and system availability

The zSeries processor and architecture have been enhanced with availability features since their introduction over 34 years ago. Consequently, they have become the standard to which other systems and architectures compare themselves. A number of hardware strategies, ranging from parity checking, recovery from intermittent failures, to error correcting codes to redundancy of components have been used for years to achieve high levels of availability.

With the recent transition of zSeries processors to CMOS microprocessor technology, the complexity and, therefore, the number of parts in the machine have been dramatically reduced, leading to even more reliable processors. More recently the zSeries processors have added such features as transparent CP sparing, storage recovery and soon nondisruptive I/O replacement. These and many other availability features make for a robust processor building block.

Use of the z/OS clustering technology is usually associated with multiple physical processors or systems. However, its use in conjunction with the ability to logically partition a zSeries processor and run multiple copies of the z/OS operating system, subsystems, and applications on a single physical processor is an attractive way to protect against planned or unplanned software outages. The ability to dynamically change partitions and rebalance workload represents a level of sophistication that is one of the highest in the industry. Thus, a z/OS client has the option to gain most of the availability benefits of clustering while retaining the cost, scale, and footprint attributes of a single processor.

### 8.3.2 High availability with system clustering

A key objective of clustering single systems is to address the increasing customer demands for improved application availability, not only in terms of failure recovery, but also for the more important reduction of planned outage times. To that end, z/OS images may be clustered to construct an environment with no single points of failure. Parallel Sysplex components can be redundantly configured as well as configuring multiple subsystem and application instances.

Without clustering, workloads that cannot fit on a single processor require splitting the workload and repartitioning the database between processors, which is a complex, resource-intensive process. Clustering single systems allows the nondisruptive addition of processing capacity, in increments matching the growth of workload requirements and without requiring the re-engineering of applications or re-partitioning of databases.

### 8.3.3 z/OS: the robust operating system for e-business

z/OS is the robust operating system that is based on the 64-bit z/Architecture. It delivers the highest qualities of service for enterprise transactions and data, and extends these qualities to new applications using the latest software technologies. It provides a highly secure, scalable, high-performance base on which to deploy Internet and Java-enabled applications, providing a comprehensive and diverse application execution environment.

z/OS takes advantage of the latest software technologies: new object-oriented programming models that permit the rapid design, development, and deployment of applications essential to enterprise e-business. It protects your investment in your present mainframe applications by providing options for integrating existing applications within your e-business infrastructure. It provides a solid base for delivering on the benefits of industry-specific UNIX® and e-business applications, supporting new technologies such as EJB, XML, HTML, LDAP, Digital Certificates, and Unicode. z/OS supports such technological advances as Parallel Sysplex processing, Intelligent Resource Director, and TCP/IP networking capabilities.

z/OS helps make critical data and processing functions accessible to end users, regardless of their location in the heterogeneous e-business world. The z/OS base includes z/OS Communications Server, which enables: world-class TCP/IP and SNA networking support, including enterprise class dependability, performance, and scalability; highly secure connectivity; support for multiple protocols; and efficient use of networking assets.

The z/OS operating system combines many features that change the playing field of I/T infrastructure design.

- Intelligent Resource Director expands the capabilities of z/OS Workload Manager to react to changing conditions and prioritize critical business workloads
- Support for 64-bit real memory and initial support of 64-bit virtual storage
- A new installation and configuration infrastructure that simplifies the installation and configuration of z/OS and related products
- Software pricing models designed to support the reality of On Demand Business

## 8.3.4 TCP/IP networking enhancements

z/OS can provide near continuous availability for TCP/IP applications and their users with two key features in z/OS: Sysplex Distributor and Virtual IP Address (VIPA) nondisruptive takeover and takeback. This is a prime example of IBM innovation and integration in software and hardware to bring value-added qualities, namely high resiliency and availability, to the zSeries networking environment. Figure 8-3 illustrates a zSeries network.

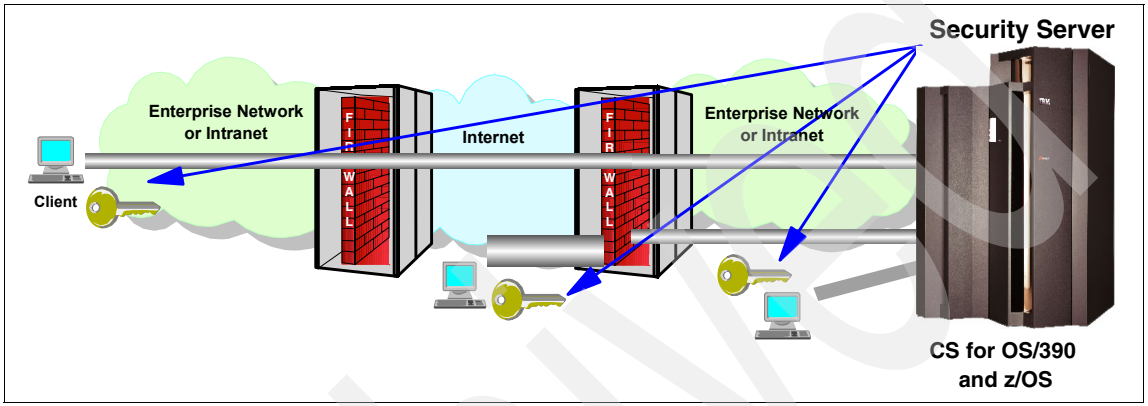



*Figure 8-3 zSeries network*

### Virtual IP address non disruptive takeover

VIPA represents an IP address that is not tied to a specific hardware adapter address. The benefit is that if an adapter fails, the IP protocol can find an alternate path to the same software, whether it is the TCP/IP services on zSeries or an application.

VIPA takeover supports movement to a backup IP stack, on a different server in a Parallel Sysplex cluster, in case of a failure of the primary IP stack. VIPA nondisruptive takeover provides VIPA takeback support. That allows the movement of workload back from the alternate to the primary IP stack.

### Sysplex Distributor

Introduced in OS/390® V2.10, Sysplex Distributor is a software-only means of distributing IP workload across a Parallel Sysplex cluster. Client connections appear to be connected to a single IP address, yet the connections are routed to servers on different zSeries or S/390 servers. In addition to load balancing, Sysplex Distributor simplifies the task of moving applications within a Parallel Sysplex environment.

### Traffic regulation management daemon

eNetwork Communication Server for z/OS, part of the operating system, provides a traffic regulation manager for limiting TCP connections. This support can be used to take preemptive action to prevent overconsumption of S/390 resources if an unexpected increase in TCP connection requests occurs. Traffic regulation can regulate the number of connections on a port basis, based on the requester's IP address and current connection consumption, as well as the state of the system. This function can be implemented to prevent flood attacks of hackers.

## 8.3.5 WebSphere Application Server for z/OS

WebSphere for z/OS V5 is an agile e-business platform designed to support today's business imperatives. WebSphere for z/OS V5 can help facilitate reducing overall costs, adapting quickly to exploit new business opportunities. Serving as an operating system for the Internet, WebSphere for z/OS V5 can help to provide:

- A comprehensive open and industry standards-based integration platform
- A highly integrated application development environment
- Agile application deployment and administration
- Intelligent end-to-end application optimization

WebSphere for z/OS V5 is a J2EE 1.3-compatible Web services application server designed to use the unique qualities of service provided by zSeries hardware and the z/OS operating system. WebSphere Application Server for z/OS includes the robust, enterprise-level capabilities designed to provide the following features:

- Support for next generation technologies is designed with J2EE 1.3-compatibility with support for key Web services.
- Integration with enterprise data and transactions enables comprehensive utilization of resources, from hardware to IT personnel, through the use of open and industry standards and integration with zSeries processes and procedures.
- Simplified deployment and management features contain improvements to both deployment and management functions to help provide a easy-to-use and more flexible environment, in a manner designed to be consistent with WebSphere Application Server Network Deployment V5.
- A configuration optimized for the z/OS and zSeries programs and products continues the legacy of close integration and leverage of the z/OS operating system and zSeries hardware.

WebSphere for z/OS works with the WebSphere Studio application development platform, which is based on the Eclipse open source project. This combined development and deployment environment can help support rapid time to market of new applications and dramatic improvements in asset reuse across the organization and over time.

WebSphere Application Server and WebSphere Studio provide a common foundation for IBM software including the rest of the WebSphere family, Tivoli® management, Lotus® collaboration, and DB2 data management products. As a result, WebSphere Application Server and WebSphere Studio can help to provide an immediate impact with today's projects and a highly extensible platform designed to deliver efficient returns as you employ new projects for On Demand Business in the future.

WebSphere Application Server offers high availability and scalability across a broad range of platforms. zSeries is a standard platform for business applications which offer the same quality of services. Therefore, as much as possible, consider deploying new technologies on z/OS. The WebSphere Application Server for z/OS V5 is designed to support the key strengths of the z/OS platform, which is designed to provide:

- **Availability**: The ability to provide near continuous service to clients without interrupts
- **Scalability**: The ability to adapt readily to changes in the workload dynamically
- **Reliability**: The ability to maintain valid data that does not become "out of sync"

WebSphere for z/OS V5 is designed to meet the needs of changing a workload and markets, while helping to maximize your return on investment.

### 8.3.6 Application availability

All processors in a z/OS Parallel Sysplex cluster can have concurrent access to all critical applications and data. Therefore, the loss of a processor or software application instance due to either a hardware or software failure does not necessitate the loss of application availability.

Peer instances of a failing subsystem executing on remaining healthy system nodes (processors) can take over recovery responsibility for resources held by the failing instance. Alternatively, the failing subsystem can be automatically restarted on healthy systems using automatic restart capabilities to perform recovery for work in progress at the time of the failure. While the failing subsystem instance is unavailable, new work requests can be redirected to other

data sharing instances of the subsystem on other cluster nodes to provide continuous application availability across the failure and subsequent recovery.

The same availability characteristics associated with handling unscheduled outages are applicable to planned outages as well. A system can be removed from the Parallel Sysplex cluster for planned hardware or software reconfiguration, maintenance or upgrade. New work can be dynamically redistributed across the remaining set of active systems. When the removed system is ready to be brought back online, it can be reintroduced into the Parallel Sysplex system in a nondisruptive manner.

### 8.3.7 Operational availability through system management

z/OS has developed robust systems management and operational capabilities over the past 30 years. With the introduction of Parallel Sysplex technology, new single system image and dynamic workload balancing capabilities were introduced. These capabilities are essential to managing and operating a cluster of z/OS-based zSeries processors.

A key design objective for Parallel Sysplex systems is to present a single system image to users, applications, and the network, and to provide a single point of control to the systems operations staff. Both hardware and software cluster components have been designed to meet this objective. New data sharing technology hardware enables multiple system nodes to serve common workloads with the appearance of a single large computing resource. Base operating system cluster services provide robust intersystem communication, system monitoring, and automatic failure takeover mechanisms. Shared consoles are provided for managing multiple operating systems and multiple underlying hardware system nodes with a single point of control. Key system profiles, catalogs, and other resources can be shared across the clustered systems to enable efficient cloning of system definitions.

Another key aspect of managing a z/OS Parallel Sysplex cluster and being responsive to changing business needs in a commercial processing environment is *dynamic workload balancing*. Dynamic workload balancing involves the ability to automatically adjust system resources as the system is running to best satisfy workload performance objectives in terms of throughput and response time. In a z/OS Parallel Sysplex cluster, the high-performance data sharing technology provides the means for z/OS and its subsystems to support dynamic workload balancing across the collection of systems in the configuration.

Functionally, workload balancing can occur at two levels. During initial connection to the cluster, clients can be dynamically distributed to server instances across the set of cluster nodes to effectively spread the workload.

Subsequently, individual work requests, such as transactions submitted by a given client, can be executed on any system in the cluster based on available processing capacity. The work requests do not have to be directed to a specific system node because of data-to-processor affinity, which is the case for other cluster designs. In a Parallel Sysplex cluster, work normally runs on the system on which the request is received, but in cases of over-utilization on a given node, work can be directed for execution on other less utilized system nodes in the cluster, taking advantage of available capacity regardless of its location in the configuration. For both online transaction processing and decision support workloads, dynamic workload balancing across systems can be made transparent to the client applications or users.

Through these and other means, system management costs do not increase linearly as a function of the number of systems in the Parallel Sysplex cluster. Rather, total cost of computing efficiencies of scale accrue through the coordinated management facilities of the Parallel Sysplex cluster.

# Part 3

# Preparing WBISF run time and Workload Manager on z/OS

This part begins with Chapter 9, “Preparing the Process Choreographer run time” on page 139. In this chapter, we explain the steps to set up the WebSphere Business Integration Server Foundation (WBISF) run-time environment on z/OS to deploy and test the sample business process application scenarios. To enable Process Choreographer functionality, you need to install the J2EE applications that represent the Business Process Container and its Business Process Engine WebClient, and the Business Rule Beans into the application server.

Then in Chapter 10, “Workload management on z/OS” on page 165, we explain how you can take full advantage of the z/OS environment by creating Workload Manager definitions and classifications. This is an optional task.

9

# Preparing the Process Choreographer run time

This chapter explains the steps to set up the WebSphere Business Integration Server Foundation (WBISF) run-time environment on z/OS in order to deploy and test the sample business process application scenarios discussed in this book. This chapter is written under the assumption that you have a fully functional WebSphere Application Server for z/OS V5.1 environment that is set up and active.

To enable Process Choreographer functionality, you need to install the Java 2 platform, Enterprise Edition (J2EE), applications that represent the Business Process Container, its Business Process Engine WebClient, and the Business Rule Beans into the application server. These applications are not installed in the network deployment manager server. The Business Process Container (BP Container) provides Process Choreographer support to these business applications. The Business Process Engine WebClient application (BPE WebClient) is used to administer and manage the business application processes.

To take full advantage of the z/OS environment, you can create workload manager definitions and classifications. This is an optional task described in Chapter 10, “Workload management on z/OS” on page 165.

## 9.1 Installing and setting up Process Choreographer

There are two ways to install the BP Container and the BPE WebClient applications into the application server:

- Using the supplied bpeconfig.jacl script

  This method automates most of the required tasks. See 3.6, “Installing Business Process Container for Process Choreographer” in the redbook *WebSphere Business Integration Server Foundation V5.1 for z/OS*, SG34-6382, for a description of this method. Also refer to the *WebSphere Application Server Version 5.1 information center* for more information about the bpeconfig.jacl script at:

  http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp

- Using the BP Container installation wizard built into the WebSphere Administrative Console (Admin Console) to configure the BP Container

  You then activate the BP Container and verify that it works. When this process is completed, you have installed these components:

  – The Business Process Engine
  – The WebClient
  – The Business Process Scheduler
  – The work item manager
  – The staff resolution plug-ins

We used the wizard to build our BP Engine. To set up the Process Choreographer, you must complete the following steps.

From the WebSphere AdminConsole:

1. Install the BP Container application bpecontainer.ear file.
2. Install the BP WebClient application processportal.ear file.
3. Define a JDBC Provider using the Universal JDBC driver for both the application server and the deployment manager server. The deployment manager requires access to the database during the installation or uninstallation of a process flow application.
4. Define a data source for the database and a Container Managed Authentication Alias containing the user ID and password used to access the database.

5. Define the required Java Message Service (JMS) queues, queue connection factories, and JMS listener ports.

   We decided to use a separate WebSphere MQ installation rather than the JMS API provided by the messaging service embedded in WebSphere Application Server. You have to install WebSphere MQ before you can start configuring the BP Container. You can use local or clustered queue definitions. Shared WebSphere MQ queues are not supported in WBISF for z/OS V5.1, but will be supported in a future release.

   If you intend to use embedded messaging, you must have selected this option when you installed WebSphere Application Server. Embedded messaging is not supported in a cluster setup.

6. Define the Scheduler configuration used by the BP Container.

From the z/OS environment:

1. Create a DB2 database for use by the Process Choreographer. The only databases supported on WebSphere Business Integration Server Foundation V5.1 for z/OS are:
   - DB2 Universal Database™ (UDB) Server for OS/390 and z/OS: This database is recommended for a production environment.
   - Cloudscape™: This database is only recommended for a single application server in test mode.
2. Create the four local WebSphere MQ queues required by the BP Container.
3. Define the Resource Access Control Facility (RACF) rules for the required EJBROLEs:
   - BPESystemAdministrator
   - WebClientUser
   - JMSAPIUser

We recommend that you follow this process to enable Process Choreographer:

1. Configure the BP Container.
2. Create and configure WebSphere MQ resources.
3. Create the DB2 resources.
4. Define the RACF rules.
5. Install the Business Rules Bean application BRBeansDB2.jar file.
6. Prepare the Business Rules Management application.

## 9.2 Configuring the BP Container

Make sure that you are logged on to the WebSphere Admin Console with sufficient administration rights for your user ID.

1. To start the dialog, click **Application Servers** → **etsr01a**.
2. In the Additional Properties section, click **Business Process Container**.
3. Scroll down past the BP Container settings. Near the bottom of the page, click the **Business Process Container Install Wizard** link.

> **Note:** Where possible, the Install Wizard offers appropriate default values in the parameter fields. However, with some browser and platform combinations, no defaults are provided. In this case, you can view the recommended values on the Install Wizard settings page.

4. In the Business Process Container Install Wizard window, for Step 1, enter the database configuration values. We chose the values as follows and as shown in Figure 9-1.
   a. In the JDBC Providers drop-down list, select **DB2 z/OS 7 (DB2 Universal JDBC Driver Provider (XA))** for the database system with the version and JDBC driver in brackets.
   b. For Implementation Classname, type the default class name `2jccDB2ConnectionPoolDataSource` provided for the JDBC driver implementation.
   c. For Classpath, enter the location of the Java archive (JAR) or zip file of the JDBC driver. To use the path variable that is displayed in the text field, you must set it explicitly under **Environment** → **Manage WebSphere Variables**.
   d. For Datasource User Name, type a name, which must be a user ID that has the authority to connect to the database and to manipulate the data. If you want to have the database schema updated automatically, the user ID must also have the authority to create tables and indexes in the database.
   e. For Datasource Password, type a password for the datasource user name.
   f. In the Custom properties field, make sure that the options match your database requirements. For more information, see the Install Wizard settings page and the product documentation for your database system.
   g. Click **Next**.

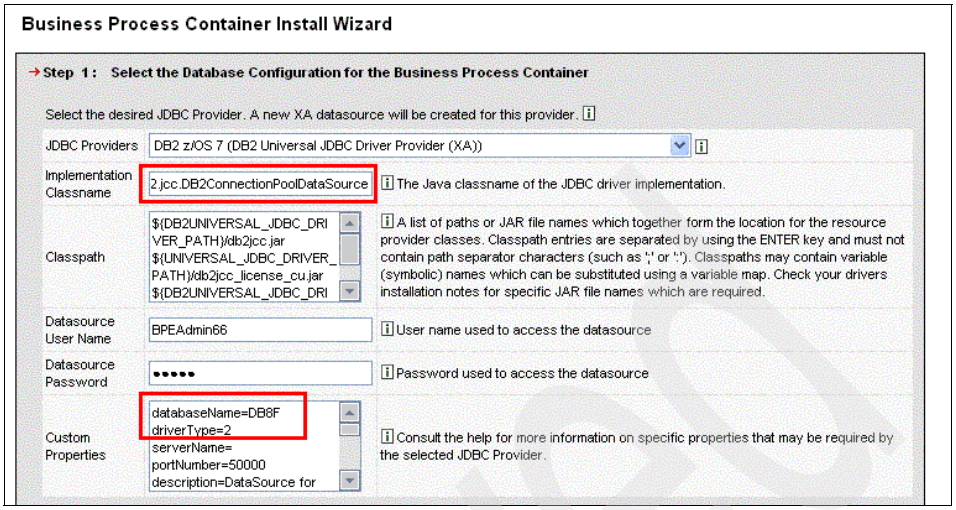



*Figure 9-1 Database configuration*

5. Configure the JMS Provider and security. We used the values that follow and as shown in Figure 9-2.

   a. In the JMS Providers drop-down list, select the messaging service that the business process container will use. We select **WebSphere MQ JMS Provider**.

   b. For Queue Manager, use the default provided (WAS_*nodeName*_*serverName*). We type `MQF1` in this example.

   c. Define the WebSphere environment variable ${MQ_INSTALL_ROOT}. Make sure that Classpath points to the MQ Java lib directory.

   d. For the JMS User ID, enter a user ID that has administration rights for the messaging service.

   e. For JMS Password, type the password for the JMS user ID.

   f. In the Scheduler Calendar field, choose the default value **com/ibm/Websphere/scheduler/calendar/DefaultUserCalendar**. Otherwise if you have your own scheduler calendar, enter its Java Naming and Directory Interface™ (JNDI) name.

   g. For Security Role Mapping, enter the user or group from your user registry that is to be mapped onto the role of Business Process Administrator. Choose to allow all users.

h. For JMS API User ID, enter the user ID that is to be used when processing asynchronous API calls.

i. For JMS API Password, enter the password for the JMS API user ID.

j. Click **Next**.

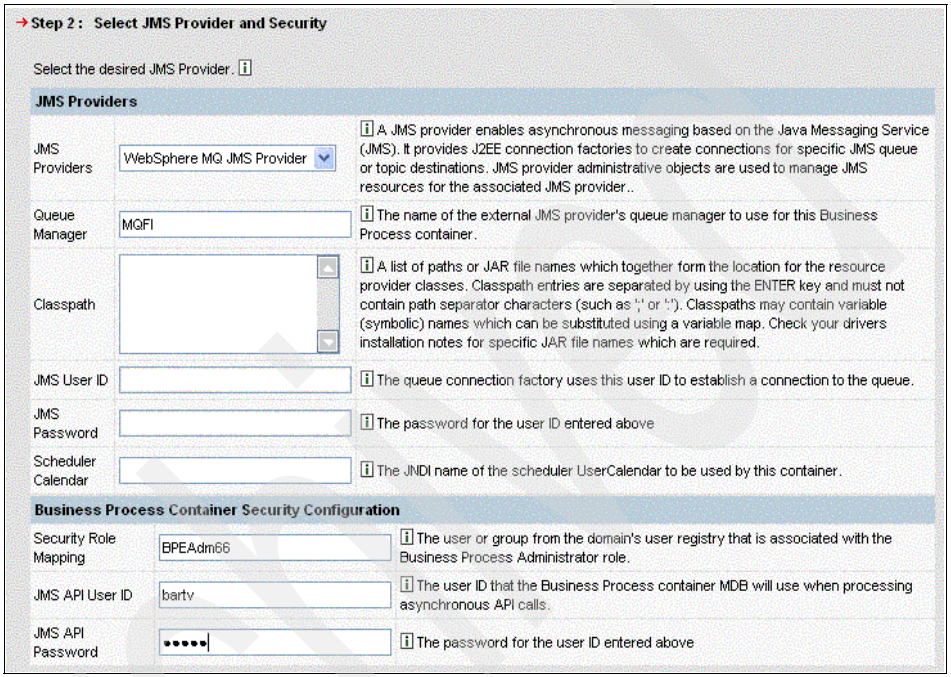



*Figure 9-2 JMS Provider and security configuration panel*

6. Choose the **Select existing JMS resources** option, and use the values we created in the previous section.

   a. From the Connection Factory drop-down list, select **BPECF**.

   b. From the Internal queue drop-down list, select **BPEIntQueue**.

   c. From the External request processing queue drop-down list, select **BPEApiQueue**.

   d. From the Hold queue drop-down list, select **BPEHldQueue**.

   e. From the Retention Queue drop-down list, select **BPERetQueue**.

7. For Step 3, select the check box to include the **WebClient** for the Rule Management application, which we explain later. Click **Next**.

8. The summary page opens (see Figure 9-3). Verify that the information on the summary page is correct.

| Datasource | | |
|---|---|---|
| Implementation Classname | com.ibm.db2.jcc.DB2ConnectionPoolDataSource | |
| Classpath | ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar | |
| Datasource User Name | bartv | |
| Datasource Password | *** | |
| Custom Properties | databaseName | DB8F |
| | driverType | 2 |
| | serverName | |
| | portNumber | 50000 |
| | description | DataSource for Process Choreographer |
| | traceLevel | |
| | traceFile | |
| | fullyMaterializeLobData | true |
| | resultSetHoldability | 2 |
| | currentPackageSet | |
| | readOnly | false |
| | deferPrepares | false |
| | currentSchema | |
| | cliSchema | |
| | enableMultithreadedAccessDetection | false |
| | retrieveMessagesFromServerOnGetMessage | true |
| | preTestSQLString | SELECT 1 FROM TABLE1 |
| | currentSQLID | |
| | loginTimeout | 0 |

*Figure 9-3 DB2 Datasource summary page*

**Note:** The summary includes reminders of which external resources are necessary. If you have not already created them, you can continue configuring the business process container, but you must create the resources before you activate the business process container. Print the summary page to help you to create the correct resources.

## 9.3 Creating and configuring WebSphere MQ resources

Create the queues and then the resource adapters required for the BP Container before you start the installation of the process container.

### 9.3.1 Creating the required WebSphere MQ queues on z/OS

You should have an existing queue manager on the z/OS system that you chose for implementation. For this queue manager, you only need to create four queues, which are named BPEApiQueue, BPEHldQueue, BPEIntQueue, and BPERetQueue.

Run the CSQUTIL job to create the four local queues using the definitions as shown in Example 9-1.

*Example 9-1 CSQUTIL to create the BPE queues*

```
DEFINE QLOCAL( 'BPEIntQueue' ) +
         QSGDISP( QMGR ) +

* Common queue attributes
         DESCR( 'BPE - Internal Queue' ) +
         PUT( ENABLED ) +
         DEFPRTY( 0 ) +
         DEFPSIST( NO ) +
         CLUSTER( ' ' ) CLUSNL( ' ' ) DEFBIND( OPEN ) +

* Local queue attributes
         GET( ENABLED ) +
         NOSHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( PRIORITY ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 999999999 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'DEFAULT' ) +
         USAGE( NORMAL ) +
         INDXTYPE( NONE ) +
         CFSTRUCT( ' ' ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         NOTRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGDPTH( 1 ) +
         TRIGMPRI( 0 ) +
         TRIGDATA( ' ' ) +
         PROCESS( ' ' ) +
         INITQ( ' ' )
```

## 9.3.2 Creating the WebSphere Application Server MQ resources

Use the WebSphere Application Server Console to configure resource adapters for use in the following steps:

1. Click **WebSphere MQ Queue Connection Factory**.
2. Click **New**.
3. Under General Properties, enter the values as shown in Figure 9-4.

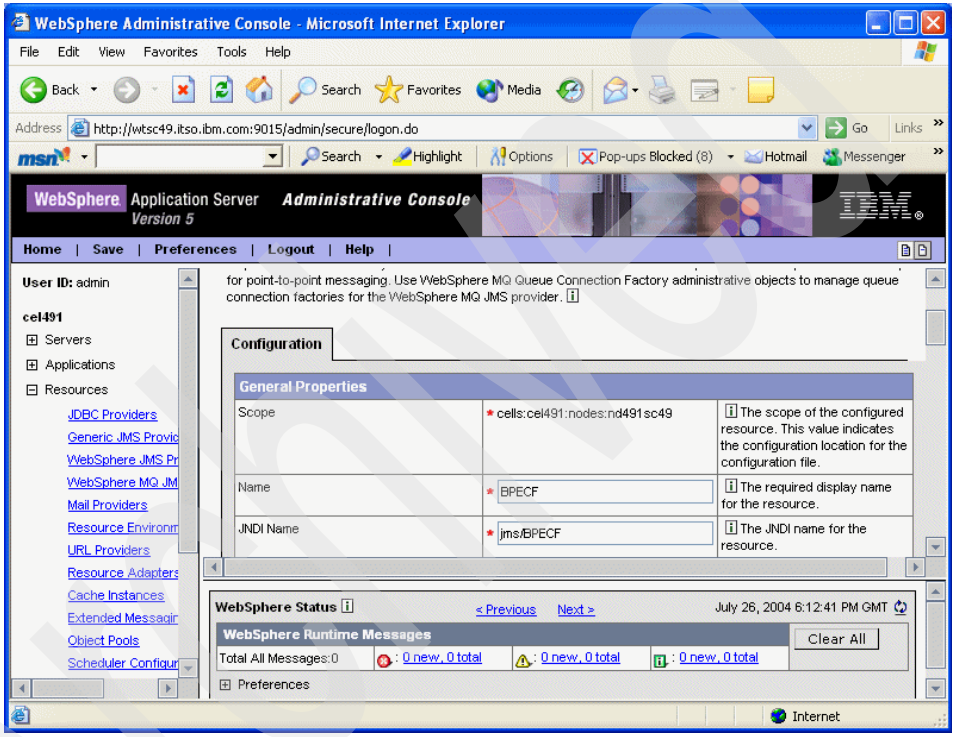



*Figure 9-4 Creating a new WebSphere queue Connection Factory*

a. In the Queue Manager field (see Figure 9-5), type the WebSphere MQ queue manager name `MQFN`.


*Figure 9-5 WebSphere MQ Connection Factory queue manager field*

b. Leave the other fields blank or accept the default values.

c. Click **Apply**, and click **Save**.

The sample WebSphere MQ Queue Connection Factory BPECF is now created and listed on the WebSphere MQ queue connection factory panel.

### 9.3.3 Defining the WebSphere MQ JMS Queue destinations

Use the WBISF default WebSphere MQ JMS queues. Table 9-1 summarizes the default queues.

*Table 9-1 WebSphere MQ JMS default queues*

| Queue name | Description |
|---|---|
| BPEApiQueue | Queue for external (JMS) API requests to BP Container |
| BPEIntQueue | Queue for internal BP Container messages |
| BPEHldQueue | Queue that holds any messages that have failed processing more times then the retry limit |
| BPERetQueue | Queue that contains messages that temporarily cannot be processed, and that will be retried |

To define the WebSphere MQ queue destination:

1. Select **Resources** → **WebSphere MQ JMS Providers**.
2. Click **WebSphere MQ Queue Destination**.
3. Click **New**.
4. For the first queue, enter the values that are listed in Table 9-2.

*Table 9-2 Example values for WebSphere MQ queue destinations*

| Field | Example value |
|---|---|
| Name | BPEApiQueue |
| JNDI Name | jms/BPEApiQueue |
| Specified Priority | 3 |
| Specified Expiry | 3 |
| Base Queue Name | BPEApiQueue |

5. Click **Apply**, and then click **Save**.
6. Repeat steps 1 through 5 to create the other three WebSphere MQ JMS queue definitions.

After you have created the four WebSphere MQ JMS queue definitions, you see them on the WebSphere MQ Destinations list panel, as shown in Figure 9-6.

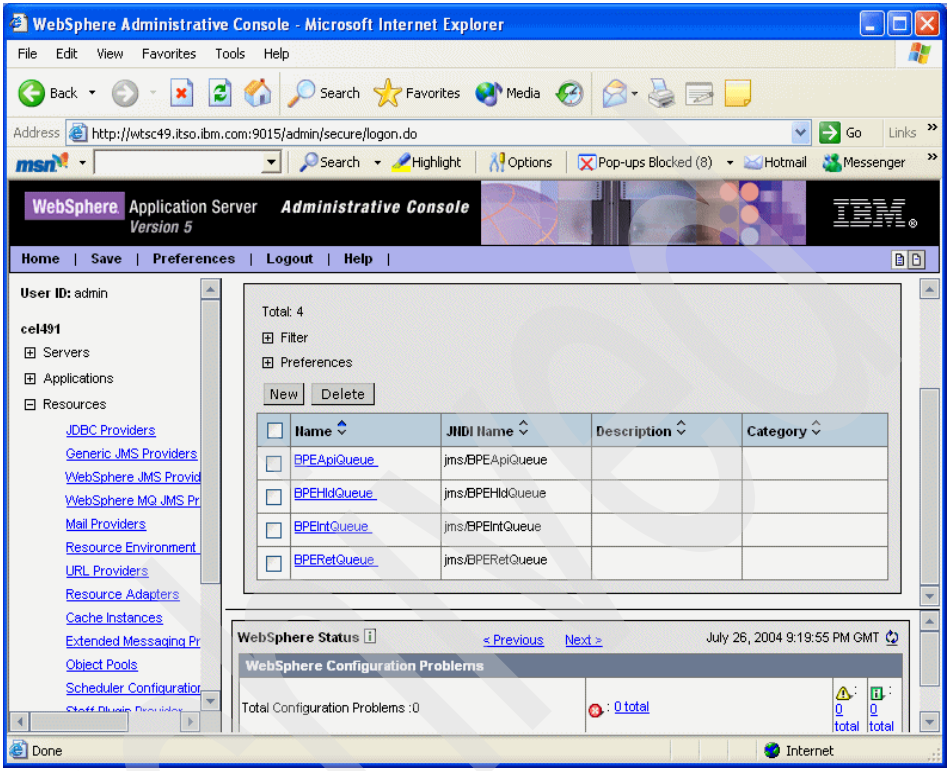



*Figure 9-6 WebSphere MQ queue destination list panel*

## 9.3.4 Creating the WebSphere Application Server listener ports

Use the WBISF default WebSphere Application Server listener ports. Table 9-3 summarizes the default listener ports.

*Table 9-3 WebSphere Application Server listener ports*

| Listener port | Description |
|---|---|
| BPEApiListenerPort | Process choreographer API |
| BPEHoldListenerPort | Process choreographer hold |
| BPEInternalListenerPort | Process choreographer internal |

To define our WebSphere Application Server listener ports:

1. Click **Servers** → **Applications**.
2. Select your application server. We selected our test server **ws491sc49**.
3. Click **Message Listener Service** → **Listener Ports**.
4. Click **New**.
5. Enter the values that are listed in Table 9-4.

*Table 9-4 Example values for listener ports*

| Field | Example value |
|---|---|
| Name | BPEApiListenerPort |
| Description | Process choreographer API |
| Connection Factory JNDI Name | jms/BPECF |
| Destination JNDI Name | jms/BPEApiQueue |
| Max Sessions | 5 |
| Max Retries | 10 |
| Max Messages | 1 |

6. Click **Apply**, and then click **Save**.
7. Repeat steps 1 through 6 to define the Process choreographer hold and internal ports.

The three listener ports that we created are now listed on the Listener Port panel as shown on Figure 9-7.

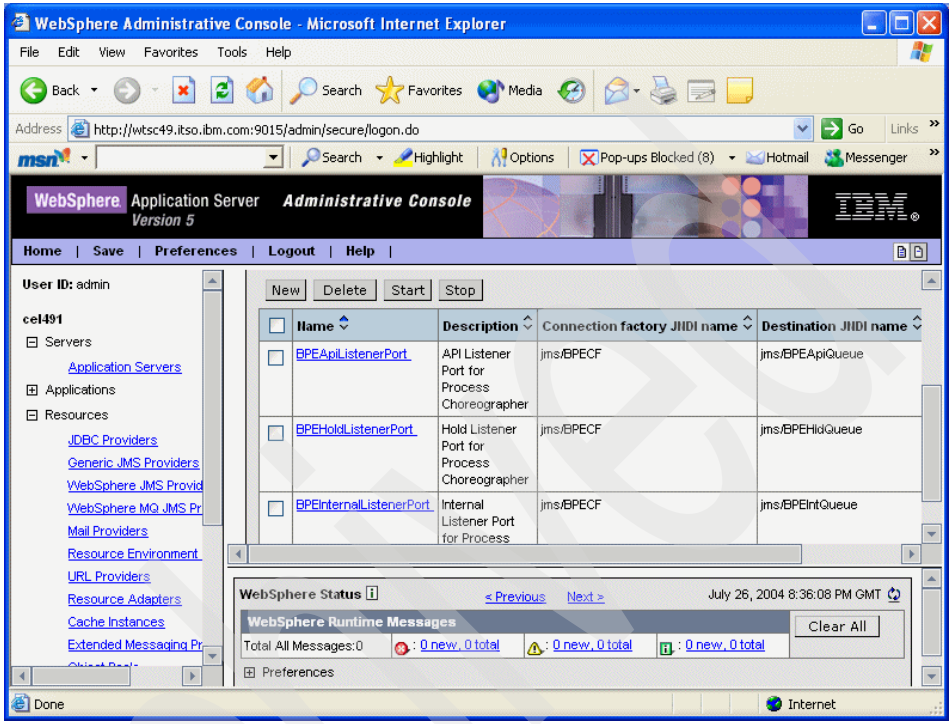



*Figure 9-7 Listener port panel*

## 9.4 Creating the DB2 resources

Create the DB2 databases, tables, and JDBC Provider. Consider the hints and tips provided in this section.

### 9.4.1 Creating the DB2 databases and tables

Create a DB2 database to store the process tables if you do not reuse an existing database. Then create DB2 tablespaces and tables for the BPE datastore in storage group SYSDEFLT. Use the Data Definition Language (DDL) statements that are shipped with WBISF. The DDLs for these definitions are available in the $WAS_HOME/AppServer/ProcessChoreographer directory:

- createSchemaDb2V7zOs.ddl
- createTablespaceDb2V7zOs.ddl

**Note:** Unfortunately, these statement do not respect the SPUFI input record limitations. It took some effort to rework them.

Set a proper current SQLID when generating the DDL to create the SDBPEDB3 objects. Grant all privileges to the WebSphere Application Server on all the tables and views in the SDBPEDB3 database.

Create the tables with the correct schema. A bypass is possible by implementing the custom property shown in Figure 9-8 in the datasource definition.

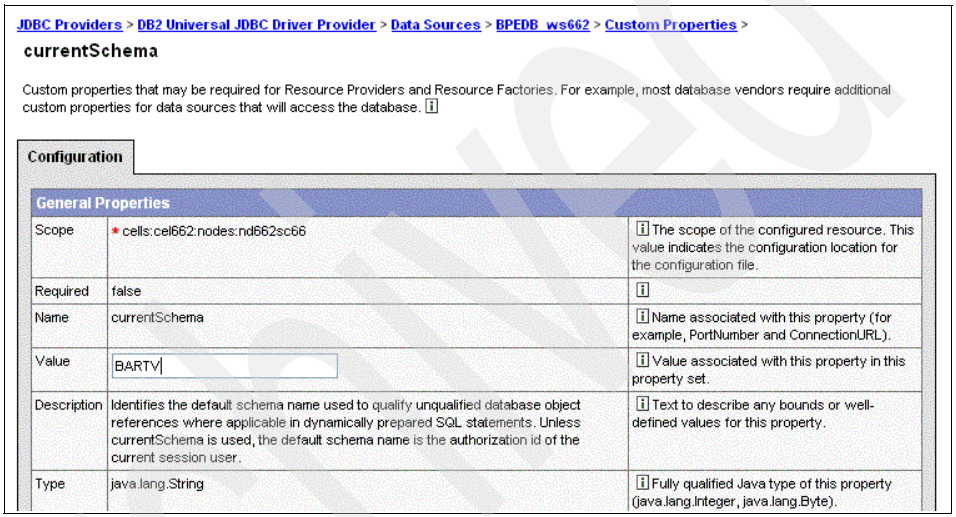



*Figure 9-8 Specifying a current schema name on the data source*

Finally, you must restart the application server to activate this datasource change.

## 9.4.2 Adding the DB2 Universal Driver JDBC provider

The DB2 Universal Driver needs to know the subsystem name of a local DB2 subsystem to connect to the data source, which in theory could be on another system.

If the db2.jcc.propertiesFile has been specified in the application server, and if this file contains the property db2.jcc.ssid, then this value should be the subsystem name of a local DB2 system (not the location name). If the property is not set, or if its value is unknown as a local DB2 ssid, then the driver looks for the subsystem name specified in the load module DSNHDECP, which should be in

the application server STEPLIB. If DSNHDECP is not found, or the subsystem name defined in it is unknown as well, then the above error message results.

We recommend that you use the following steps:

1. Determine the location of the DB2 Universal Driver class files.
2. Update the server environment variables DB2UNIVERSAL_JDBC_DRIVER_PATH and DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH as shown in Figure 9-9 and Figure 9-10.

> **Note:** Because SC66 did not have a local DB2 HFS mounted, we referred to the file system on SC49 as indicated in Figure 9-9 and Figure 9-10.

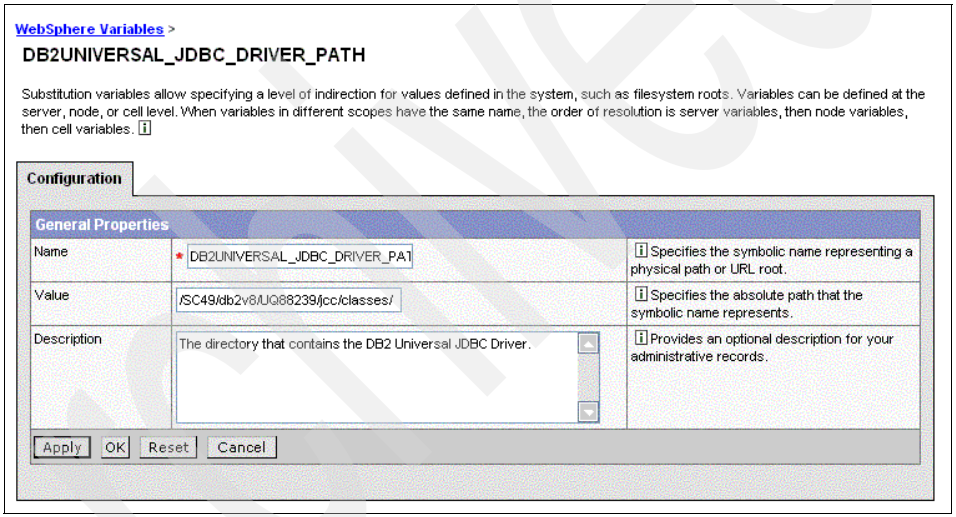



*Figure 9-9 Setting DB2UNIVERSAL_JDBC_DRIVER_PATH*

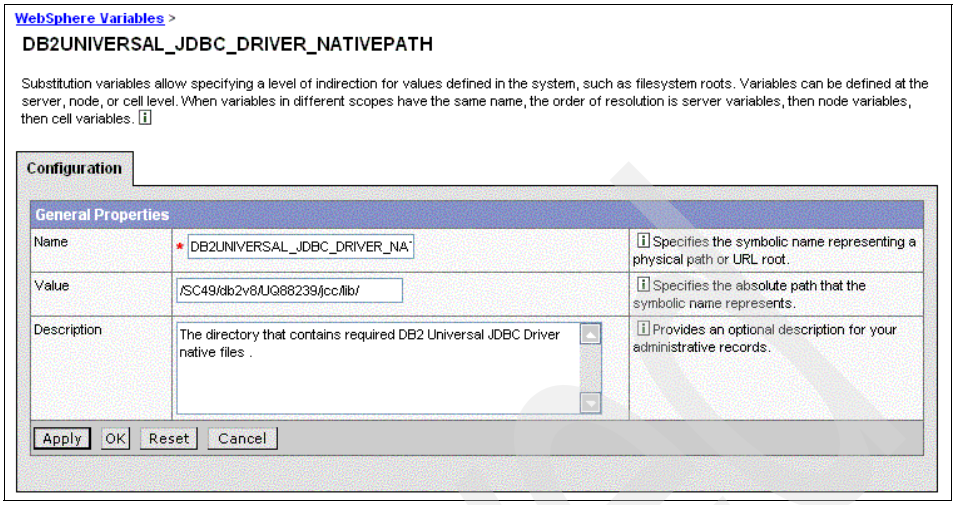



*Figure 9-10 Setting DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH*

3. Create the JDBC Provider using the DB2 Universal Driver. Set its type to a Type2 driver, so that it can exploit RRS. See Figure 9-11.

> **Note:** When using the Type 2 version of the DB2 Universal Driver, you use native operating system calls. This is the reason why you have to define the DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH variable.

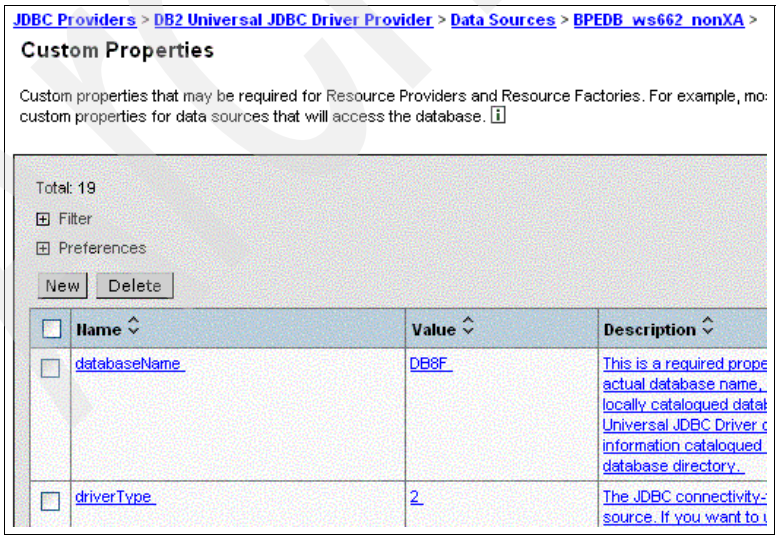



*Figure 9-11 Setting up the Type 2 data source in DB2 Universal Driver JDBC provider*

4. Define a user ID alias for container managed access to the DB2 data source for the BP Container as shown in Figure 9-12.

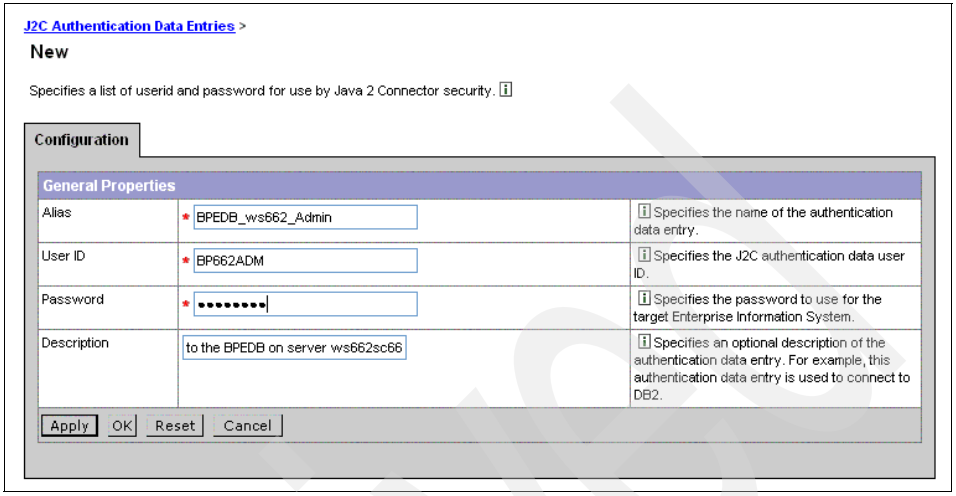



*Figure 9-12 Setting up container-managed alias for Business Process Engine datastore*

5. Set up a data source for the BP Engine.

   You can re-use an existing data source. However if you want to customize security settings and other properties independently from other applications, create a new data source. See Figure 9-13.

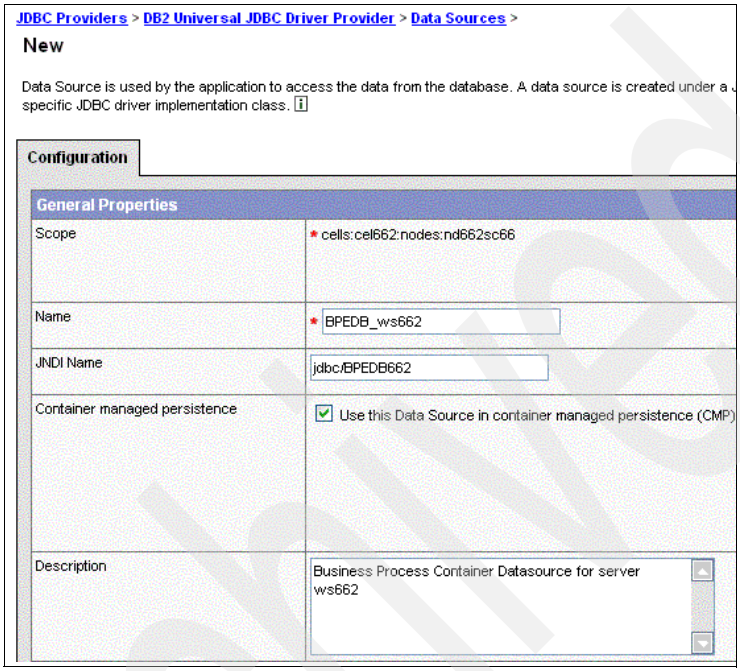



*Figure 9-13 Setting up the BPEDB data source*

6. Set the databaseName property correctly for the data source as shown in Figure 9-14.

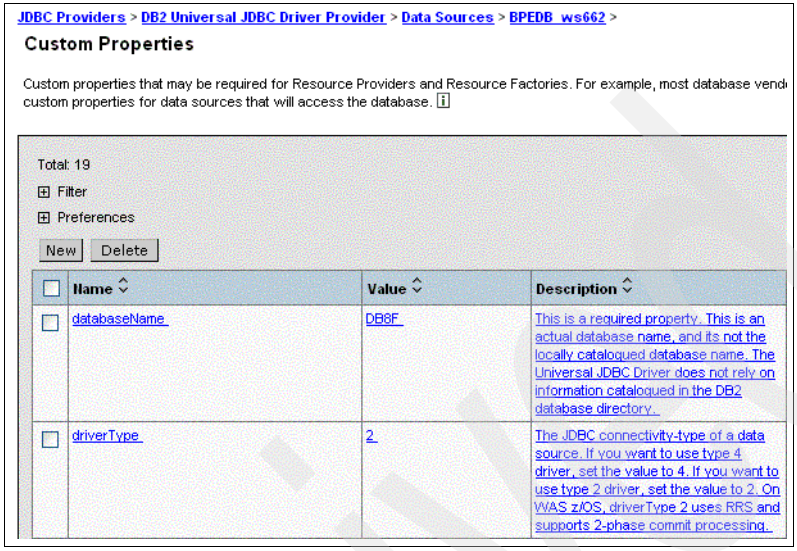



*Figure 9-14 Setting up the databaseName property in the BPEDB data source*

## 9.4.3 BPEScheduler requirement for JNDI to be jdbc/BPEDB

The BPEScheduler component of the BP Container fails to start, because it is unable to find the resource referenced by JNDI name jdbc/BPEDB. We have only installed a data source with the JNDI name jdbc/BPEDB662nonXA.

We have created an additional definition with the JNDI name jdbc/BPEDB as required by the application.

## 9.5 Defining the RACF rules

The BP Container and the WebClient applications require RACF EJBROLE definitions. Table 9-5 displays the roles and their users.

*Table 9-5 RACF roles and users*

| Role | Users are |
|---|---|
| WebClientUser | General business process users |
| BPESystemAdministrator | Business process administrators |
| JMSAPIUser | Able to invoke JMS API calls; BP Choreographer message-driven beans require an ID under this role. |

Example 9-2 shows the definition of the required RACF rules.

*Example 9-2 EJBROLE definitions*

```
RDEF EJBROLE SD.WebClientUser UACC(NONE)
RDEF EJBROLE SD.BPESystemAdministrator UACC(NONE)
RDEF EJBROLE SD.JMSAPIUser UACC(NONE)
SETR RACLIST(EJBROLE) REFRESH
PE SD.WebClientUser CL(EJBROLE) ID(xxx) ACC(READ)
PE SD.BPESystemAdministrator CL(EJBROLE) ID(xxx) ACC(READ)
PE SD.JMSAPIUser CL(EJBROLE) ID(xxx) ACC(READ)
SETR RACLIST(EJBROLE) REFRESH
```

## 9.6 Installing the Business Rules Bean application

Business Rule Beans (BRBeans) are used to separate business rules from an application's core behavior, allowing the application code to remain intact and untouched even as business practices change. Each business rule is represented by an entity bean that persistently stores information related to that rule. Each business rule is assigned an appropriate rule name and stored in a rule folder. The application developer identifies points of variability within an application and codes trigger points at these locations. These trigger points invoke one or more business rules.

The BRBeans are used by the BP Container and can be accessed from your workstation by the BRManagement application, *rulemgmt*.

From the WebSphere for z/OS Admin Console, perform the following steps:

1. Configure the DB2 resource adapter.
2. Install the BRBeansDB2.jar file.
3. Create the DB2 database on z/OS.

## 9.6.1 Configuring the DB2 resource adapter

Consider the differences between Cloudscape and DB2 that were not documented in the installation procedures.

A DB2 data source must be set up as a Version 4 Datasource for the BRBeans support. If not properly set, when you initially execute the beans, you receive the following message:

```
Property "serverName" not specified for BRBeans datasource
```

You can correct that by setting up the proper data source, as shown in Figure 9-15.

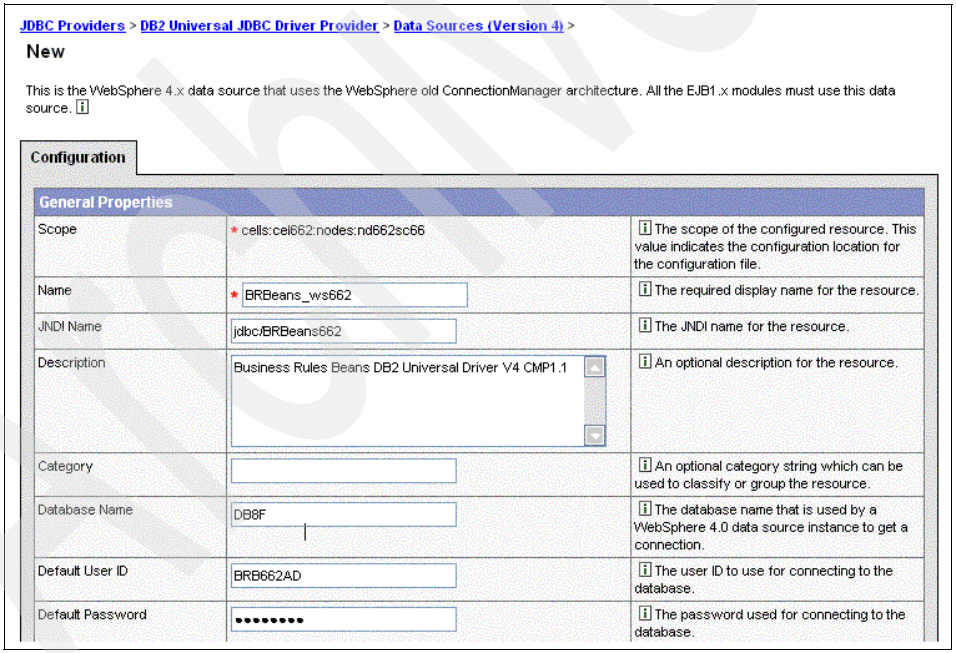



*Figure 9-15 Setting up a Version 4 data source for BRBeans*

This setting implies is that the serverName property is mandatory, even for a Type 2 driver data source. Therefore, be careful to set the properties as shown in Figure 9-16.

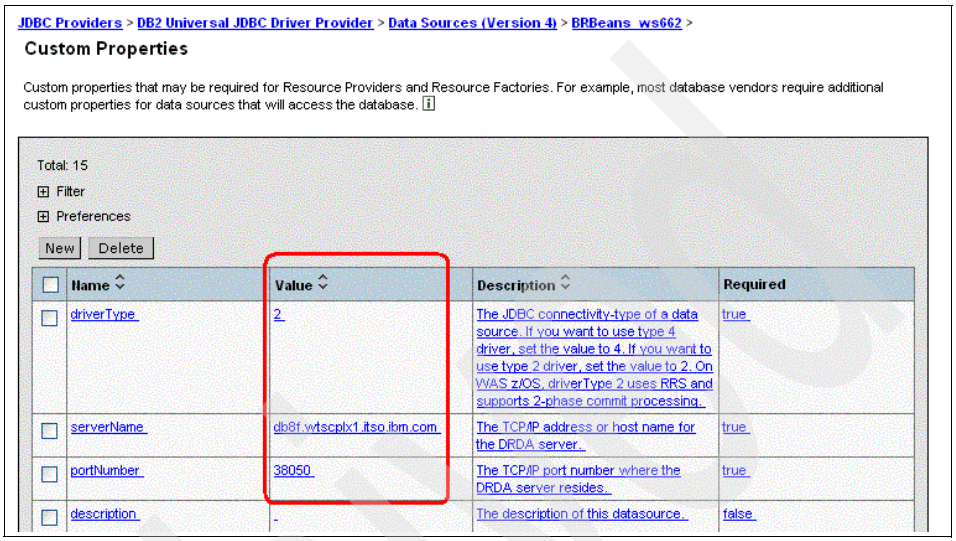



*Figure 9-16 Setting up serverName for BRBeans data source*

## 9.6.2 Installing the BRBeansDB2.jar

From your WebSphere for z/OS Admin Console, install the BRBeansDB2.jar file, which is in the $WAS_HOME/BRBeans directory. You must select the check box to create the DDL statements, which are required to create the tables in the next step.

## 9.6.3 Creating database for the Business Rule Beans

Use z/OS to create the database and the two tables that are needed to run your business rules application.

The DDL that is generated when using the Export DDL button from the Admin Console will not execute on a DB2 for z/OS system because it is geared toward a distributed DB2 environment. For example, a CREATE SCHEMA statement is generated, which is not supported on DB2 for z/OS V8.1.

By default, the tables must be prefixed by EJB. This schema name is set in the BRBeansDB2.jar application file that was installed in the application server. This schema name can be changed only by modifying the definition using WebSphere Studio V5.1.

Create the DB2 table for the Business Rule Beans framework using the DDL provided in Example 9-3. This DDL has been modified for DB2 for z/OS V8.1 system. Modify it to suit your installation.

*Example 9-3 DDL to create Business Rule Beans DB2 tables*

```
-- ------------------------------------------------------------------
-- CREATE DATABASE AND TABLESPACE FOR WBISF Z/OS BRBEANS
-- ------------------------------------------------------------------

  CREATE DATABASE SDBRBDB1 IN STOGROUP SG6382;

  CREATE TABLESPACE SDBRBT1 IN SDBRBDB1
      USING STOGROUP SG6382
            PRIQTY      250
            SECQTY      100
            ERASE       NO
            LOCKSIZE    ANY;

-- ------------------------------------------------------------------
-- CREATE TABLES FOR WBISF Z/OS BRBEANS
-- ------------------------------------------------------------------

  CREATE TABLE EJB.BRBEANS_RULE
     (PRIMARYKEY VARCHAR(41) NOT NULL,
      BUSINESSINTENT LONG VARCHAR,
      CLASSIFICATION VARCHAR(251),
      CLASSIFIER SMALLINT,
      DEPENDENTRULES LONG VARCHAR,
      DESCRIPTION LONG VARCHAR,
      ENDDATE TIMESTAMP,
      FIRINGLOCATION INTEGER,
      FIRINGPARAMETERS LONG VARCHAR,
      INITPARAMETERS LONG VARCHAR,
      JAVARULEIMPLNAME VARCHAR(251),
      ORIGINALREQ LONG VARCHAR,
      PRECEDENCE INTEGER,
      READY SMALLINT,
      RULEFOLDERID VARCHAR(41),
      RULENAME VARCHAR(200),
      STARTDATE TIMESTAMP,
      USERDEFINEDDATA LONG VARCHAR,
      CONSTRAINT PK_BRBEANSR PRIMARY KEY (PRIMARYKEY)
    )
     IN SDBRBDB1.SDBRBT1;

  CREATE UNIQUE INDEX EJB.BRBEANS_RULE_IDX1 ON EJB.BRBEANS_RULE
    (
     PRIMARYKEY
```

```
  )
  USING STOGROUP SG6382;

  CREATE TABLE EJB.BRBEANS_RULEFOLDER
    (PRIMARYKEY VARCHAR(41) NOT NULL,
     NAME VARCHAR(200),
     PARENTKEY VARCHAR(41))
   IN SDBRBDB1.SDBRBT1;

CREATE UNIQUE INDEX EJB.BRBEANSRF_IDX1 ON EJB.BRBEANS_RULEFOLDER
  (
   PRIMARYKEY
  )
  USING STOGROUP SG6382;

COMMIT WORK;
```

## 9.7 Installing the Business Rules Management Client

In BRBs, rule administration involves making changes to the set of business rules being used by applications. This includes the following activities:

- Creating new rules
- Deleting existing rules
- Creating a new rule with the same name to replace an existing one
- Changing the expiration date of an existing rule
- Moving rule changes between systems

To set up the Business Rule Management client, you need to update the rules database via WebSphere Studio V5.1 on your workstation.

1. If you do not have WebSphere Studio V5.1 already installed on your workstation, install the IBM WebSphere Integration Server clients V5.1, by specifying the options as shown in Figure 9-17.

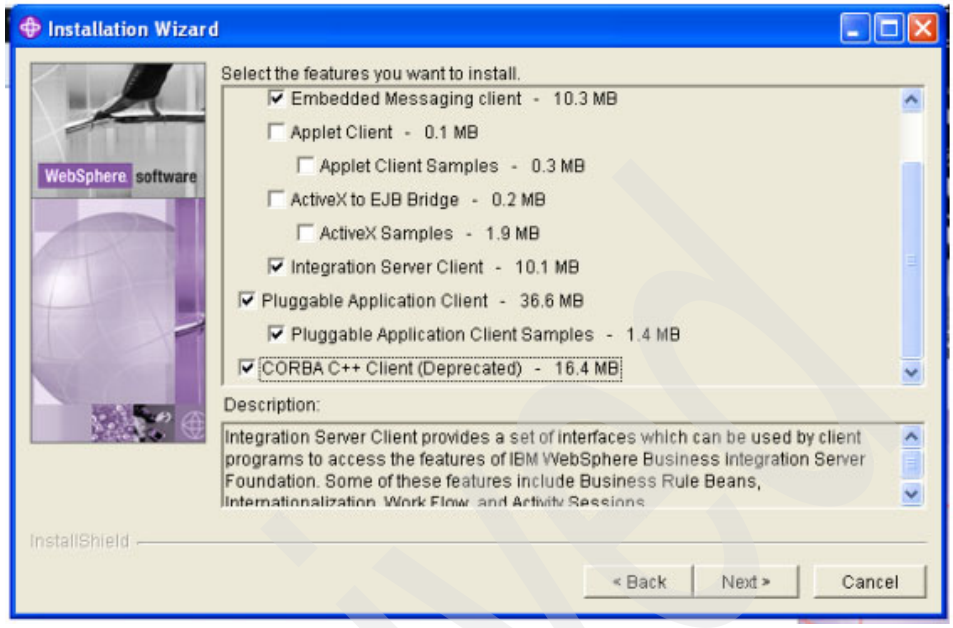



*Figure 9-17 Integration Server Client*

2. Copy the BRB.properties file.

   Copy this file from your installed application directory on z/OS using FTP. This gives you the proper values for your z/OS BRBeans run time as shown in Example 9-4.

*Example 9-4 Sample BRB.properties*

```
host=wtsc49.itso.ibm.com
port=9013
RuleJndi=brbeans/application/Rule
RuleFolderJndi=brbeans/application/RuleFolder
RuleHelperJndi=brbeans/application/RuleHelper
```

3. Edit the rulemgmt.bat file. As of this writing, no fix was available to correct the rulemgmt.bat file. You must fix the typographical errors manually. The location of the file is C:\Program Files\WebSphere\AppClient\bin.

To start the business rule management application:

1. Open a command window.
2. Change the directory to `C:\Program Files\WebSphere\AppClient\bin`.

3. To update the business rule table, type the following command:

```
rulemgmt z:\SG246356\AddMat\BRBeans\BRB.properties wtsc49.itso.ibm.com 9013
```

This command uses the following format:

– *rulemgmt*: The command to update the table

– *z:\SG246356\AddMat\BRBeans\BRB.properties*: The location of the BRB properties file

– *wtsc49.itso.ibm.com*: The host name of your WebSphere Application Server

– *9013*: The bootstrap port for WebSphere Application Server where your business rules beans are located

10

# Workload management on z/OS

In this chapter, we provide an overview of workload management in z/OS and explains how WebSphere on z/OS exploits its functionality. We explain queuing manager services, enclave services, application environment services, and workload profiles.

We also cover the Workload Manager (WLM) classification rules for the STC and CB subsystems. The creation of WLM classifications is an optional task to set up the WebSphere Business Integration Server Foundation (WBISF) run-time environment for our scenarios. We created the workload definitions to take full advantage of the z/OS environment.

## 10.1 Benefits of using WLM on z/OS

Workload management optimizes the distribution of incoming work requests to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests. WLM also provides failover when servers are not available, improving application availability.

WebSphere for z/OS requires that WLM runs in *goal mode*. For details about WLM, see the following product manuals:

- *z/OS MVS™ Planning: Workload Management,* SA22-7602
- *z/OS MVS Programming: Workload Management Services,* SA22-7619

In addition to setting up WLM in goal mode, you need to define workload management policies for your application servers.

> **Note:** You do not need to define special classification rules nor work qualifiers to get started. However, we recommend it for your production system.

WLM on z/OS provides the following benefits to WebSphere for z/OS applications:

- It balances server workloads, allowing processing tasks to be distributed according to the capacities of the different system images in the sysplex.
- It provides failover capability by redirecting client requests if one or more servers is unable to process them. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by a single system image configuration. With clustering, additional instances of servers can easily be added to the configuration.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes the administration of servers and other objects.

WLM in z/OS is based on the concept of grouping work into service classes. The incoming work request is classified to a service class and the WLM schedules the resources to complete the work request according to this service class. Figure 10-1 shows how WebSphere work requests are classified into service classes.

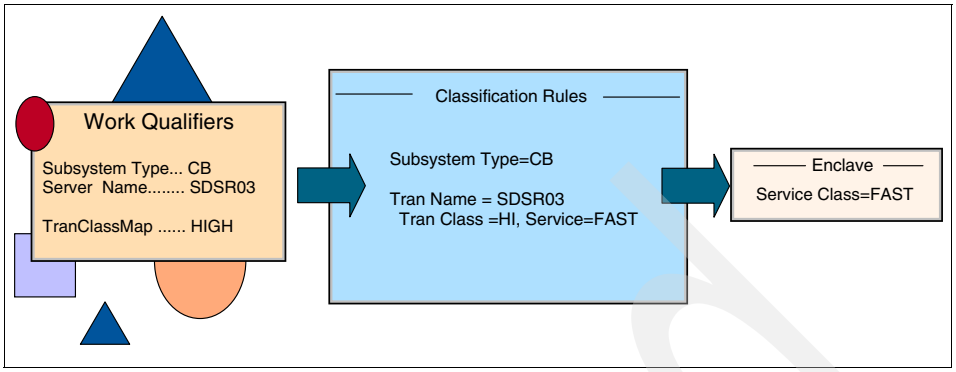



*Figure 10-1 Classification of work requests*

The following components are illustrated in Figure 10-1:

- Work qualifier

  WebSphere for z/OS associates each work request with a work qualifier that identifies a work request to the system.

  The WebSphere application workload runs in subsystem CB.

  > **Subsystem CB:** Subsystem CB is an IBM-supplied workload model for Component Broker-like workloads. It represents a set of CB objects grouped together and run in a logical server. It is used to manage transactions as WLM enclaves. IBM supplies other workload models for subsystems such as CICS, IMS, and OMVS.

  The following work qualifiers are supported for subsystem CB:

  – **CN**: The *collection name* is the server_generic_short_name for the WebSphere for z/OS application server.

  – **SI**: The *server instance name* is the server_specific_short_name.

  – **TC**: The *transaction class* is assigned to the transaction using the transaction class mapping file for the server. This facility allows you to differentiate performance objectives between multiple transactions running on the same WebSphere server.

  – **UI**: The *user name* is the one under which the work request is run.

- Classification rules

  Classification rules associate a work request, as defined by its work identifier, to a WLM service class.

- Service class

  The z/OS WLM organizes work into workloads and service classes. The service class for a group of work defines the performance goal and business importance.

  - Performance goals

    There are three kinds of goals:

    - The response time goal indicates the response time for individual transactions.
    - Execution velocity goals are suitable for started tasks or batch jobs.
    - Discretionary goals are for low priority work.

    A response time objective is usually consistent with the business requirement of a WebSphere application. Also, the response time option automatically generates response time distribution information to be reported through a Resource Measurement Facility (RMF™) report. You may find this option useful for investigating response time issues.

  - Business importance of the work

    The business importance for a service class defines the significance for achieving the performance goal for that service class. At runtime, the WLM component manages workload distribution and allocation of resources to competing workloads. High priority workloads receive guaranteed, consistent results, such as for response time and throughput.

## 10.2 WebSphere on z/OS using WLM services

WebSphere Application Server for z/OS V5 exploits WLM services to process incoming requests:

- Queuing services to manage execution priority of address spaces and work requests
- Enclave services to manage performance of WebSphere transactions across multiple address spaces
- Application environment services to dynamically manage the number of servant address spaces

## 10.2.1 Queuing manager services

Queuing services manage the execution of address spaces and work requests that they process to meet service class performance goals. The classification of each transaction is managed by the *controller region*. The controller region acts as a queuing manager that queues work requests to WLM for execution in servant address spaces. WLM maintains the queues for passing work requests from the controller region to each servant region.

The controller region listens for work requests and puts them on the WLM queue. The WLM component of z/OS dispatches the work to the servant region according to the WLM policy specified by the work identifier. Figure 10-2 illustrates the flow from the WebSphere controller region to the servant regions.

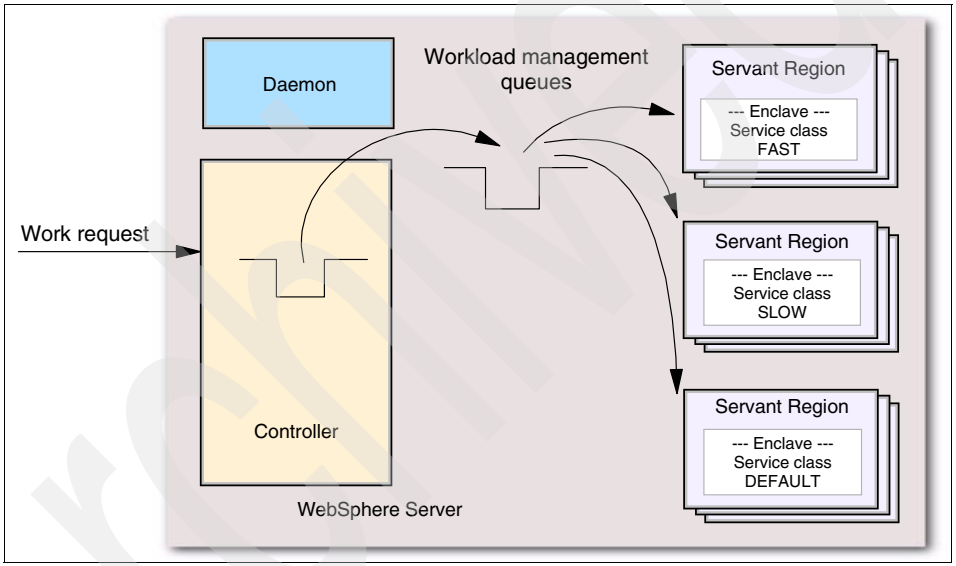



*Figure 10-2 Interaction between WLM and controller and servant regions*

## 10.2.2 Enclave services

Enclave services allow performance management of a transaction across multiple address spaces and systems inside a sysplex. WebSphere for z/OS uses an independent enclave type, which reports on resource consumption based on a performance goal associated with the transaction. This is unrelated to the performance goal or goals of the address space or spaces in which the enclave dispatchable units run.

The controller region creates the enclave and associates the transaction with this classified enclave. Then the transaction is queued, waiting to be served by an available thread in a servant region.

> **Note:** A WLM enclave is a transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces and is reported on and managed as a unit.

## 10.2.3 Application environment services

Application environments allow WLM to start new servant address spaces in order to meet transactions performance goal. WLM starts a new servant region address space or stops a servant region address space as the workload varies.

WLM starts additional servant regions to meet performance goals, based on:

- The prioritization of its work compared to other work in the system
- The availability of system resources needed to satisfy those objectives
- A determination by WLM of whether starting more address spaces will help achieve the objectives

By default, WebSphere for z/OS servers are configured to allow only a single servant region. To enable multiple regions and specify the maximum and minimum number of servant regions, use the Admin Console. From the initial panel, select **Servers** → **Application Servers** → ***server_name***. Then from the Additional Properties section, select **Server instance**. Figure 10-3 shows the Server Instance panel.

Under General Properties, select the **Multiple Instances Enabled** option. Then specify the minimum number and maximum number of instances to set boundaries on the number of servant regions that WLM will start.

- **Minimum Number of Instances** is used to start a basic number of servant regions before the day's work arrives. This can save time in waiting for WLM to determine that more servant regions are needed.

- **Maximum Number of instances** is useful to cap the number of servant address spaces started by WLM if you determine that excessive servant regions could lead to service degradation.

The minimum and maximum number of instances specified also have an influence on the dispatching of transactions managed by WLM.

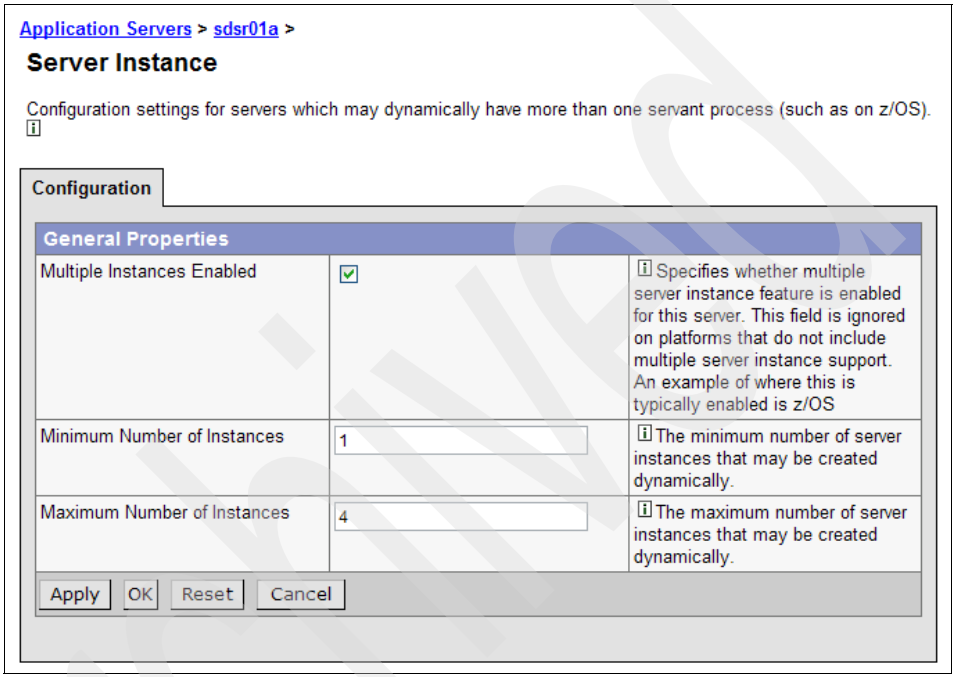



*Figure 10-3 Minimum and maximum number of instances*

Transactions received by the WebSphere for z/OS controller region are passed to servant regions through a set of WLM queues. The number of queues is determined by the number of service classes defined, and one servant region only serves one service class at a given time.

To ensure that you do not limit the parallelism of execution under a full load, the maximum number of instances should be set at least as large as the number of service classes that are defined.

If you specify a maximum number of instances that is too low, there might not be enough servants available for the number of WLM queues, which will result in a bottleneck of the queue under full load conditions. As a consequence, the system might experience queuing delays, resulting in transactions having an elongated response time.

Another performance factor to consider to determine the optimum number of servant regions is the number of threads that are available to each servant region, as defined in the workload profile. The workload profile controls workload-pertinent decisions made by the WebSphere for z/OS runtime, such as the number of threads used in the server region. The default workload profile value is IOBOUND.

You can change the workload profile value through the Admin Console. Select **Application Servers** → ***server_name*** → **ORB Service** → **Advanced Settings**. Figure 10-4 shows the Advanced Settings panel.

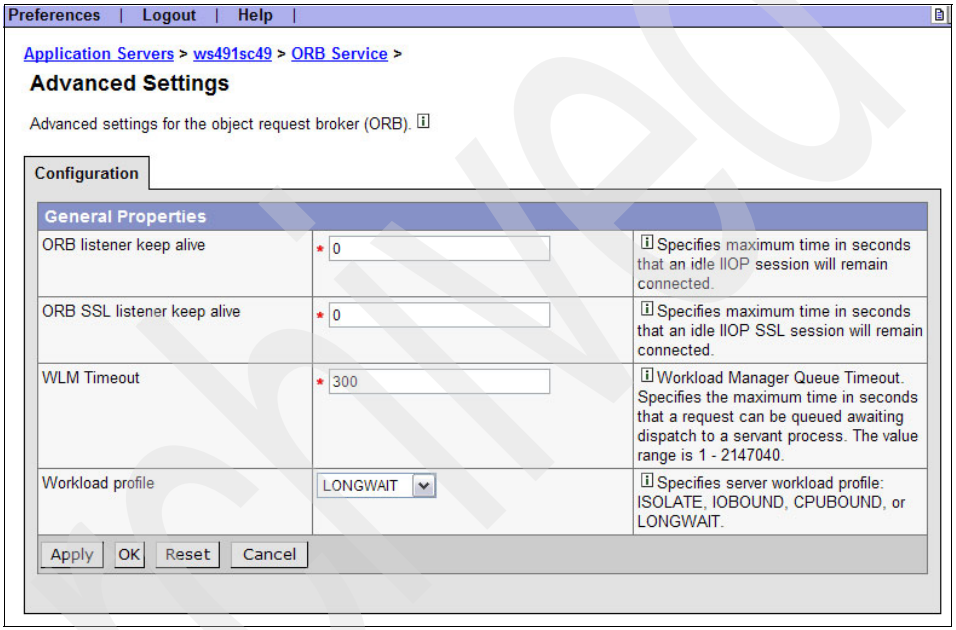



*Figure 10-4 Workload profile*

The server workload profile can be specified as ISOLATE, IOBOUND, CPUBOUND, or LONGWAIT.

- **ISOLATE**: This value specifies that the servant region is restricted to one single application thread.
- **IOBOUND**: This value specifies more threads for applications that perform I/O-intensive processing. IOBOUND is recommended for most applications that have a balance of central processing unit (CPU) activity and remote operation calls. The calculation of the thread number is based on the number of CPUs, using the following formula:

  `MIN(30, MAX(5,(Number of CPUs*3)))`

- **CPUBOUND**: This value specifies that the application performs processor-intensive operations, and therefore, would not benefit from more threads than the number of CPUs. The calculation of the thread number is based on the number of CPUs, using the following formula:

  `MAX((Number of CPUs-1),3)`

  **LONGWAIT**: This value specifies more threads than IOBOUND and is intended for applications that spend most of their time waiting for network or remote operations to complete. LONGWAIT allocates 40 threads.

## 10.3 WLM classifications rules

Classification rules associate a work request, as defined by its work identifier, with a WLM service class. Figure 10-5 illustrates the flow of the classification process.

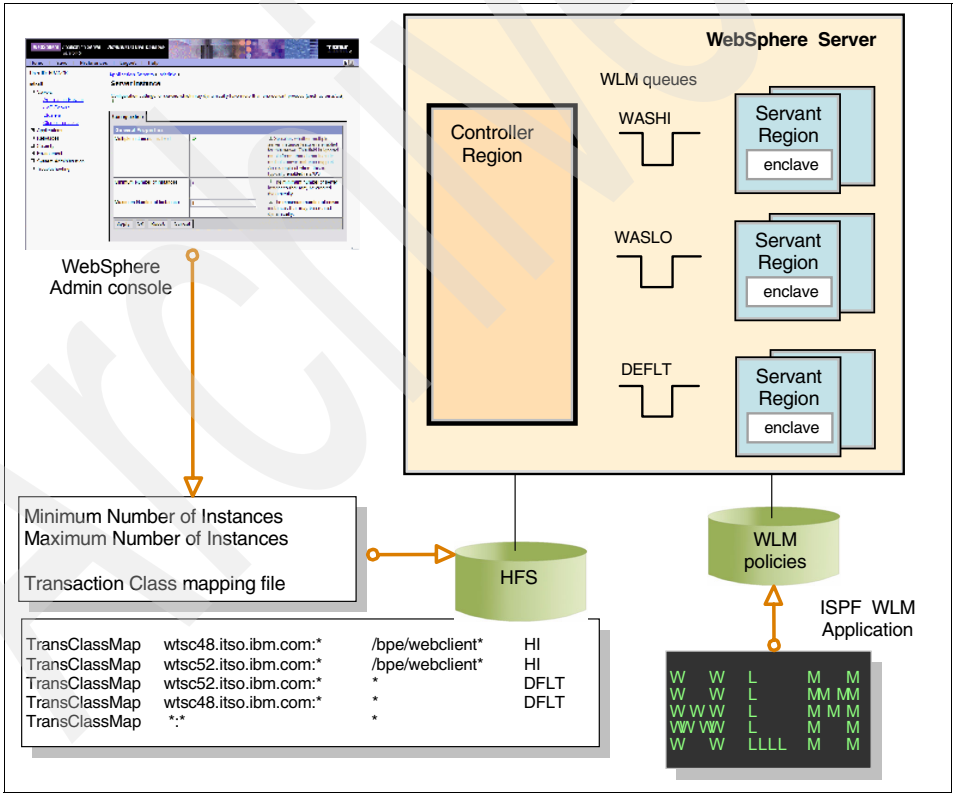



*Figure 10-5 WebSphere for z/OS run time and WLM*

## 10.3.1 Classification rules for the STC subsystem

The rules for the STC subsystem are created using the WLM ISPF application, as shown on Figure 10-6.

```
                 Modify Rules for the Subsystem Type     Row 1 to 16 of 66
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : STC         Fold qualifier names?   Y  (Y or N)
Description  . . . ________________________________

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                            More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                Service     Report
                                             DEFAULTS: SYSSTC      OTHER
 ____  1  TN         SDSRO3%S ___                    VEL75       WASS
 ____  1  TN         SDSRO3%  ___                    SYSSTC      WAS
 ____  1  TN         SCSCE*   ___                    CICSRGN     WASC
 ____  1  TN         RRS      ___                    VELRRS      ________
 ____  1  TN         D8I%IRLM ___                    SYSSTC      WASD
 ____  1  TN         D8I%MSTR ___                    VEL80       WASD
 ____  1  TN         D8F%IRLM ___                    SYSSTC      WASD
 ____  1  TN         D8F%MSTR ___                    VEL80       WASD
```

*Figure 10-6 WLM definition of WebSphere server regions for the STC subsystem*

The controller region should be assigned to an STC service class with a high importance and velocity goal. There is a certain amount of processing in the WebSphere application control regions to receive work into the system, manage the HTTP transport handler, and classify and route the input work. Since controller regions act as work routers, they must have high priority.

The servant regions should be assigned to an STC service class so that they can be initialized quickly when WLM determines they are needed. This service class is used for tasks that run in the servant region under control of the step task and not as part of the enclave. The service class should be lower in the service class hierarchy than more important work such as Controller region, production WebSphere enclaves, or CICS and IMS transaction servers.

The service class chosen also determines the WLM goal when work that is not part of the enclave is run under control of the servant address space (for example, the Java Garbage Collection). Also, any threads that are spun off in the servant regions are not run under the enclave, but under the servant region STC classification.

You might also want to define report classes for the address spaces STC activity. This allows you to monitor the activity within the WebSphere regions for service tasks running outside the enclave.

## 10.3.2 Classification rules for the CB subsystem

The WLM classification rules are used for WebSphere for z/OS applications that run in the servant region as part of the dispatched enclave. Each WebSphere transaction is dispatched as a WLM enclave and is managed within the servant region according to the service class assigned through the CB service classification rules.

```
                 Modify Rules for the Subsystem Type     Row 1 to 15 of 16
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : CB          Fold qualifier names?   Y  (Y or N)
Description  . . . WebSphere App Server

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                            More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                Service     Report
                                          DEFAULTS: WASDF       OTHER
 ____   1  CN         ETDMGR*  ___                  WASDM       WASE
 ____   1  CN         SDSRO3*  ___                  WASLO       WASE
 ____   2    TC         LO       ___                WASLO       WASE
 ____   2    TC         MED      ___                WASDF       WASE
 ____   2    TC         HI       ___                WASHI       WASE
 ____   2    TC         DFLT     ___                WASDF       WASE
 ____   1  CN         ETSRO*   ___                  WASHI       WASE
 ____   2    TC         SGETA001 ___                WASDF       SGETA001
 ____   2    TC         SGETA002 ___                WASDF       SGETA002
 ____   2    TC         SGETA003 ___                WASDF       SGETA003
 ____   1  CN         ET*      ___                  WASDF       WASE
 ****************************** BOTTOM OF DATA ******************************
```

*Figure 10-7 WLM definitions of the servant regions for the CB subsystem*

The CB classification rules can be based on a server name, server instance name, user ID, or transaction class. You can assign a default transaction class to the servant, or you can use the virtual host name, port number, or URI template to map specific requests to a transaction class using a transaction mapping file.

When a transaction mapping file is used, it must be defined to the server through the administration console as shown in Figure 10-8.

From the initial panel, select **Servers** → **Application servers** → ***server name*** → **Web Container** → **Advanced Settings**.

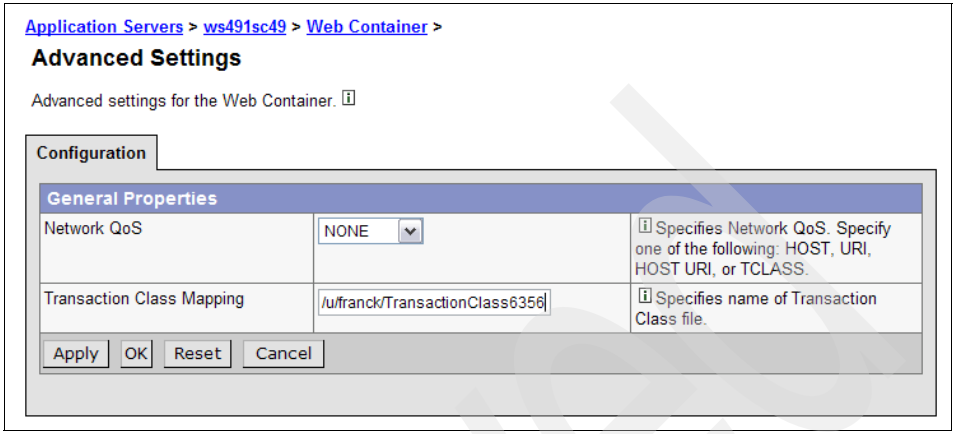



*Figure 10-8 Transaction class mapping*

A transaction class mapping file allows you to associate a set of URIs with a specific transaction class. At execution time, WLM uses the transaction class to associate the work request with a service class. Example 10-1 shows a transaction class mapping file used for our tests.

*Example 10-1 Transaction class mapping file*

```
TransClassMap wtsc49.itso.ibm.com:*   *                LO
TransClassMap wtsc49.itso.ibm.com:*   *                LO
TransClassMap wtsc66.itso.ibm.com:*   *                LO
TransClassMap wtsc66.itso.ibm.com:*   /bpe/webclient*  MED
TransClassMap wtsc48.itso.ibm.com:*   /bpe/webclient*  HI
```

For more information about WLM, refer to *How to Classify HTTP Transactions in WebSphere for z/OS V5* on the Technical Sales Library Web site at:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD101151

# Part 4

# Technical scenarios

This part describes sample business scenarios and their implementations in WebSphere Business Integration Server Foundation (WBISF) on z/OS.

First in Chapter 11, “Applied Decomposition pattern: Spot Loan scenario” on page 179, we describe the business scenario called Spot Loan for the imaginary client ITSO Bank. We explain how the Decomposition pattern is applied and mention a few design guidelines. In addition, we outline how to deploy the sample application from several perspectives.

Then in Chapter 12, “Applied Decomposition pattern with human interaction: Staff Assisted Loan scenario” on page 203, we present a sample business scenario called Staff Assisted Loan for the ITSO Bank that uses the Decomposition pattern with human interaction. We outline the business scenario, present some design guidelines, explain how to prepare the sample application, and describe the environment in which it will be deployed.

In Chapter 13, “Run-time scenario on z/OS Parallel Sysplex” on page 219, we introduce more complexity to the ITSO Bank business scenario. We illustrate the deployment process of the sample application into WBISF on z/OS and provide details about the test environment. We also explain how to generate Web services-based deployment code from CICS business logic in a step-by-step approach. And we explain how it should be implemented into a z/OS Parallel Sysplex environment.

In addition, in Chapter 14, “Best practices for high availability on z/OS” on page 255, we document our architectural design decisions for leveraging the requirements of the business process scenario, for example high availability and scalability. We provide guidelines and best practices for using specific z/OS features, such as TCP/IP, Workload Manager, DB2 Data Sharing, Resource Recovery Services (RRS), and Session affinity.

11

# Applied Decomposition pattern: Spot Loan scenario

This chapter presents a sample business scenario, called *Spot Loan*, for an imaginary client called ITSO Bank. It describes the business profile, business goals, and existing environment for which we chose a Self-Service pattern. We apply the Decomposition pattern and mention a few design guidelines.

Then we outline the steps necessary to deploy the Spot Loan sample application from the perspectives of:

- The developer
- The WebSphere z/OS deployment administrator
- Application support team

We also explain the steps to run the Spot Loan sample application in our particular environment.

## 11.1 Business scenario

Our business scenario involves an imaginary client, ITSO Bank. ITSO Bank wants to make its banking services available to its customers online. Currently, customers have to visit a bank location to do all their banking activities. The bank wants to implement a solution that allows customers to perform some activities online, such as pay bills, transfer money, and get loan approvals. A primary business goal is to reduce the customer transactions that require teller involvement. Figure 11-1 shows a high level overview of the ITSO Bank business processes.

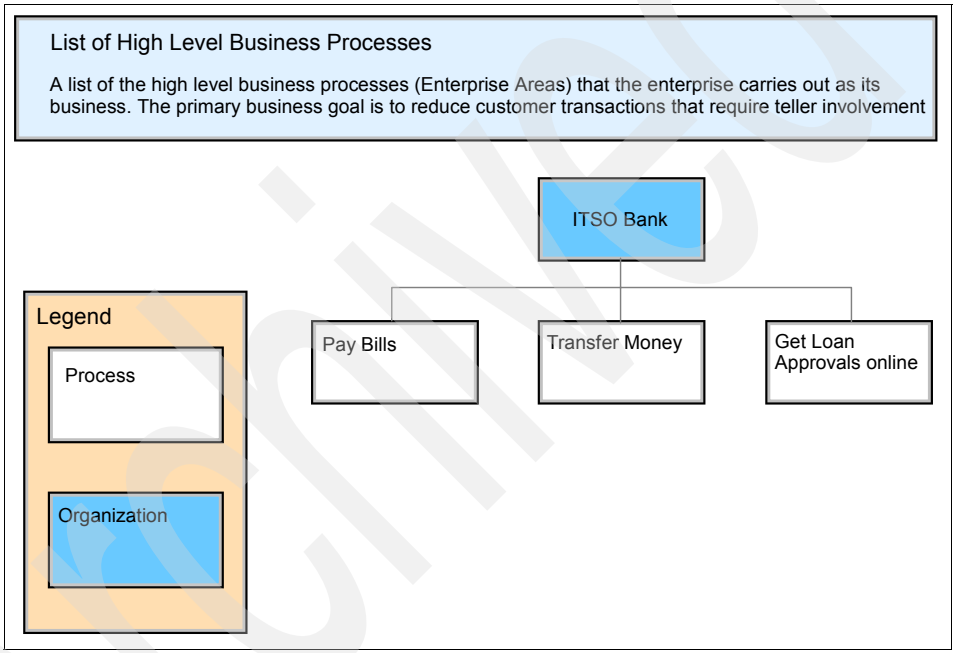



*Figure 11-1 ITSO Bank: High-level business processes*

**Note:** This section is common to all samples used in this part and describes the business profile, business goals and existing environment.

### 11.1.1 Business profile

ITSO Bank is a banking institution that was founded 25 years ago. The bank has grown from a small family-owned community bank to a large regional bank. It provides the typical banking services including checking, savings, credit card services, and loan services.

### 11.1.2 Business goals

Traditionally, clients had to visit a bank location to conduct business with the bank. As the bank's client base grows, their goal is to provide a Web site for client convenience and to automate many of their business processes.

By creating the online Web site and automating business processes, ITSO Bank plans to:

- Reduce costs by decreasing the number of physical branch locations and tellers required to handle the large number of clients. Automation of more common banking tasks will further reduce costs by pre-screening loan applications before involving loan managers.
- Increase customer satisfaction by providing quick and easy access to banking services.
- Increase the flexibility of the business by externalizing business rules from the business process.

Figure 11-2 summarizes the list of ITSO Bank business goals and critical success factors.

There is also the possibility of a future merger with a larger bank and partner relations with other financial institutions. ITSO Bank wants to structure their business processes to easily accommodate any changes in data processing that future mergers might require.

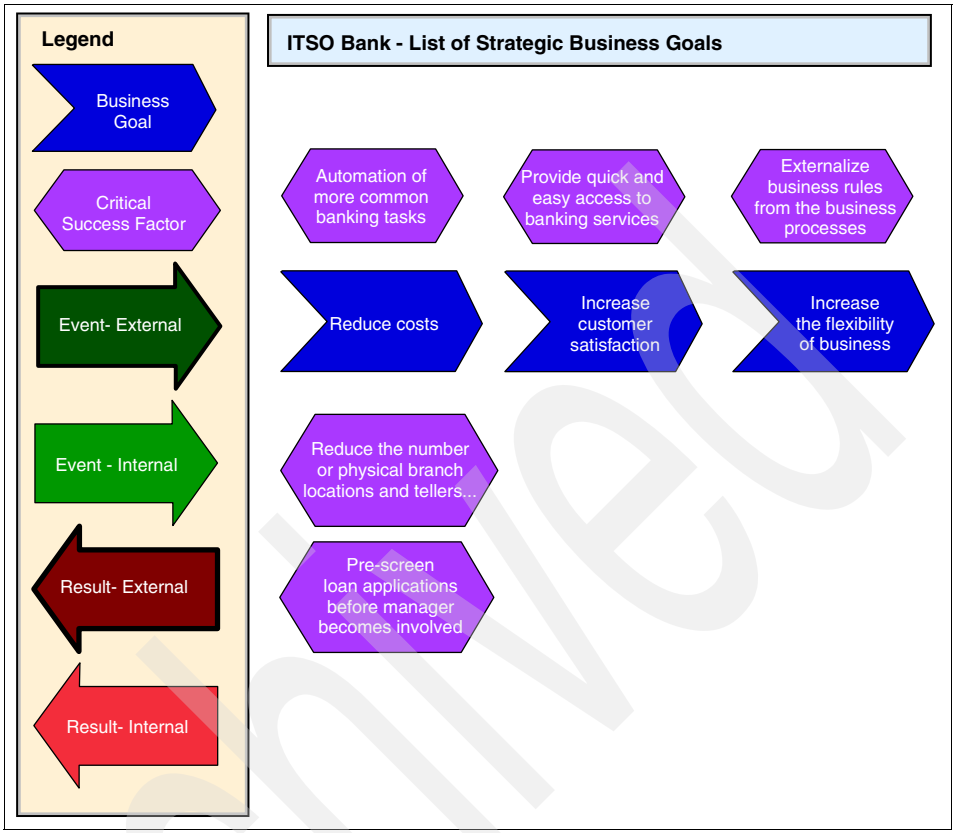



*Figure 11-2 ITSO Bank: List of strategic business goals*

### 11.1.3 Existing environment

From the business perspective, the existing environment includes the following functions:

- Customer service is done on a face-to-face basis with bank tellers. Tellers interact with the banking systems.
- Loan requests are approved by a loan manager. The loan manager manually checks account balances, customer credit history, and the current credit rating rules used by the bank.

From the IT perspective, the existing environment uses the following systems:

- An existing banking mainframe system that provides standard bank services, including checking, savings, loans, and account transfers
- An internal system that supplies an indicator of a customer's credit rating based on indicators calculated from that customer's information
- Machines where the teller enters a customer's request

### 11.1.4 Non-functional requirements

ITSO Bank requires that all solutions provide a standard Quality of Services (QoS) set. The following specific criteria must be met:

- Availability
  - Solutions meet both the defined unplanned- and planned-downtime requirements.
  - Meaningful messages are provided to system users during downtime.
- Operability

  Solutions provide suitable logs and traces.
- Federation

  The responsibilities of the stakeholders are clearly defined and agreed to by all parties.
- Performance
  - Solutions meet the defined throughput and response times.
  - Solutions scale to provide for future growth.
- Security
  - Sensitive systems and data are protected from unauthorized access.
  - Non-repudiation of the end user for all transactions is provided.
- Standards compliance

  Appropriate standards are identified and applied.
- Transactionality

  Transaction support is required for our business scenarios where funds are transferred.

In a real-world implementation, you would define such requirements in measurable terms to ensure that the solution meets the demands of the organization. Since this is beyond the scope of this book, we do not define these requirements for our sample scenarios.

### 11.1.5 Business pattern selected

The Self-Service business pattern captures the essence of direct interactions between interested parties and a business. Interested parties include customers, business partners, stakeholders, employees, and all other individuals with whom the business intends to interact. For this reason, the Self-Service pattern is an appropriate business pattern for the ITSO Bank to use in developing solutions that allow their customers to do online banking.

## 11.2 Design guidelines

The ITSO Bank wants to implement an online bill payment process that allows customers to pay bills by transferring money from an existing account. In most cases, sufficient funds will exist in the customer's account to pay the bill. However, in the event that the funds are not there, the bank wants the ability to advance an instant loan to qualified customers.

### 11.2.1 Spot Loan scenario: online funds transfer with Spot Loan

For this scenario, we can identify two *actors*:

- The customer
- The credit rating service

We can also identify a *use case*: Spot Loan.

Table 11-1 provides details about the customer.

*Table 11-1 Customer actor details*

| | |
|---|---|
| **Actor name** | Customer |
| **Brief description** | The customer is a banking client that holds one or more accounts in the bank. |
| **Status** | Primary |
| **Relationships** | |
| **Associations to use cases** | 001 Spot Loan |

Table 11-2 provides details about the credit rating service.

*Table 11-2 Credit rating service actor details*

| | |
|---|---|
| **Actor name** | Credit rating service |
| **Brief description** | The credit rating service provides a credit rating level for the customer. |
| **Status** | Primary |
| **Relationships** | |
| **Associations to use cases** | 001 Spot Loan |

### 11.2.2 Use case 001: Spot Loan

Table 11-3 provides details about the Spot Loan use case.

*Table 11-3 Use case 001: Spot Loan*

| | |
|---|---|
| **Use case name** | 001 Spot Loan |
| **Subject area** | Online funds transfer |
| **Business event** | A customer transfers money from one account to another to manage account balances or to pay a bill. |
| **Actors** | Customer, credit rating service, and banking service |
| **Use case overview** | The customer requests a transfer of money from one account to another. If the account has sufficient funds, the banking service transfers the money. If the account doesn't hold sufficient funds, the bank reserves funds for a loan.<br><br>It uses the customer's pre-calculated credit rating held on file as input to the credit rating service, which determines if the customer meets the requirements for a spot loan. If the credit rating is high enough, the loan is approved and banking services transfers the money from reserved bank funds to the account. If the loan is not approved, the reserved bank funds are released. |
| **Preconditions** | The customer supplies the amount of funds, the account to transfer from, and the account to transfer the money to. |
| **Termination outcome 1** | The bank service transfers the money and records the transaction. |

Figure 11-3 illustrates the use case model for this scenario.

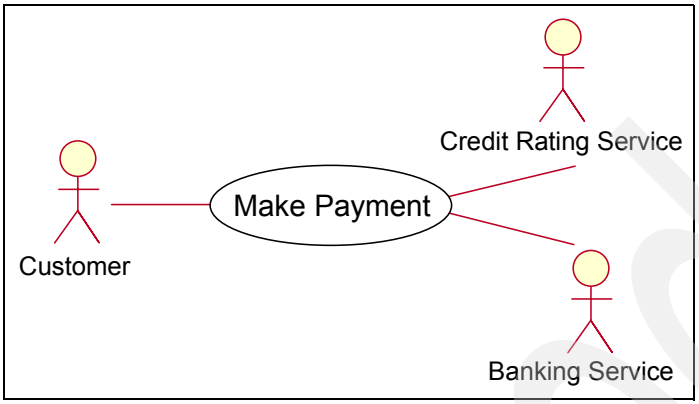



*Figure 11-3 Spot Loan scenario use case model*

We can also use the traditional business process methodology to describe our sample business scenario as shown in Figure 11-4.

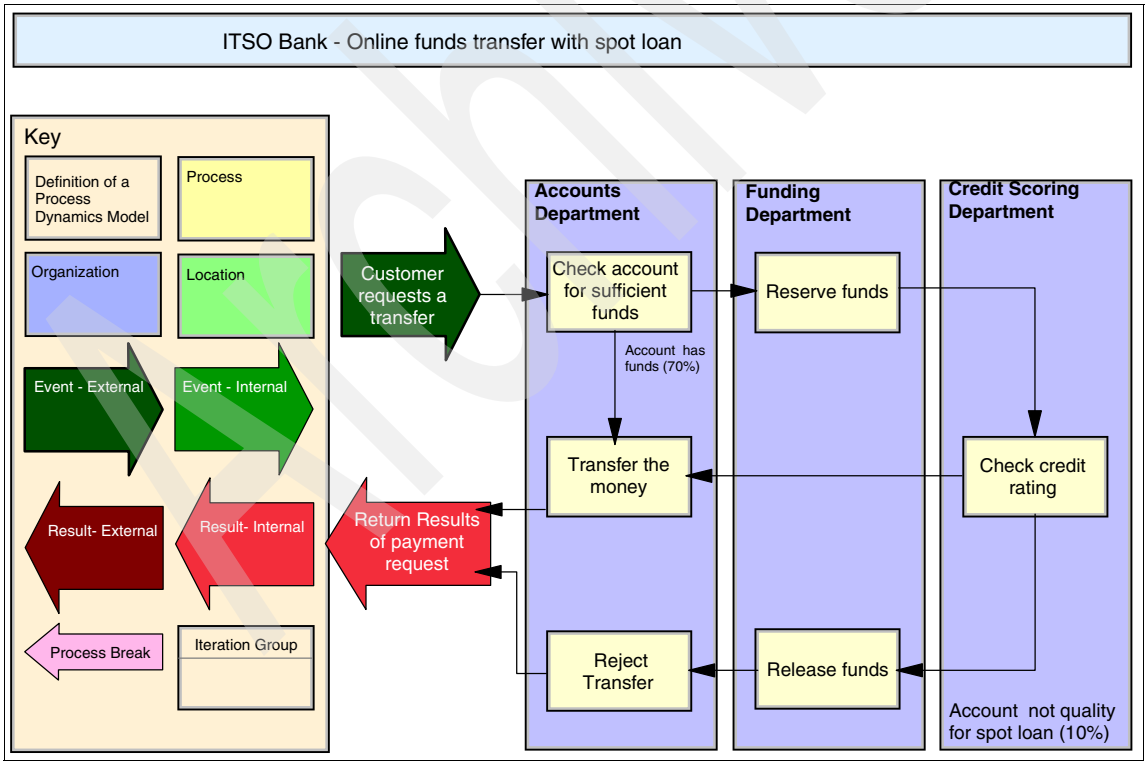



*Figure 11-4 ITSO Bank: Online funds transfer with spot loan*

### 11.2.3 Application pattern selected

The Decomposition pattern is applied to this scenario. The solution requires intelligent decision making capabilities and the ability to invoke multiple back-end services in a specific order for the transaction to complete successfully.

The Process Manager provides the capability of invoking the banking services required and making logical decisions about how the flow through the process should proceed.

## 11.3 Deploying the Spot Loan sample: the developer perspective

To prepare the application for deployment, you need to set up the development environment, import the business process, and generate deploy code for the process. This is the Spot Loan sample deployment process from the developer perspective.

The Spot Loan scenario operates in a common WebSphere Studio Application Development environment.

> **Attention:** You must install WebSphere Studio Application Developer Integration Edition 5.1 fixpack 2 to use this sample.

### 11.3.1 Importing the existing banking application

Follow these steps to import the existing banking application:

1. Open the WebSphere Studio workbench.
2. Select **Windows** → **Preferences**.
3. In J2EE preferences window, select the **Enable server targeting support** check box.
4. Select **File** → **Import** and select the **EAR file**. Click **Next**.

5. In the Enterprise Application Import panel (Figure 11-5), complete the following tasks:

   a. Click **Browse** and select the **BankingEAR.ear** file.
   b. Ensure that the project name is `BankingEAR`.
   c. Select **Integration Server v5.1** for the target server.
   d. Click **Finish**.

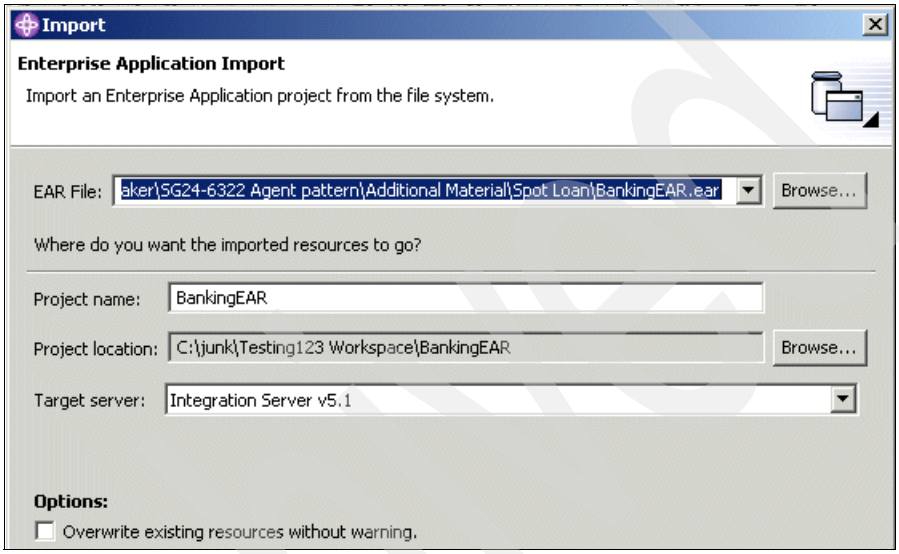



*Figure 11-5 Importing the bankingEAR file*

6. In the Project Navigator view, select **BankingEJB**, right-click, and select **Properties**.
7. In the Properties window, select **Java Build Path**. Complete these steps:

   a. Click the **Libraries** tab.
   b. Click **Add Variable**.
   c. Select **WAS_EE_V51**. Click **Extend**.
   d. From the lib directory, select **wsatlib.jar**, and click **OK**.

8. In the Project Navigator view, select **BankingRuleEJB**, right-click, and select **Properties**.
9. In the Properties window, select **Java Build Path**. Complete these steps:

   a. Click the **Libraries** tab.
   b. Click **Add Variable**.
   c. Select **WAS_EE_V51**. Click **Extend**.
   d. From the lib directory, select **brbRuleMgmtApp.jar**, and click **OK**.

You have now imported the existing banking application.

## 11.3.2 Importing the business process

Next you need to import the business process into the service project that you just created.

1. Unzip the BankingProcessBPEL.zip file directly into the workspace. See Figure 11-6 as an example for a workspace called *Sample Workspace*.

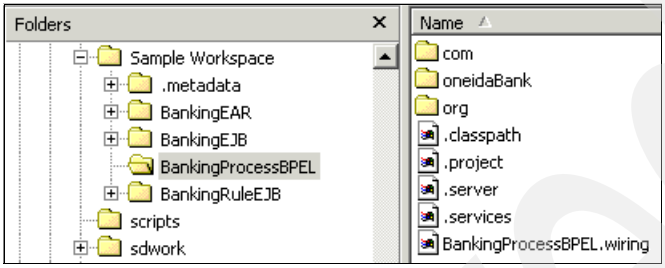



*Figure 11-6 Business process unzipped to workspace*

2. In WebSphere Studio, select **File** → **Import** → **Existing Project into Workspace**.
3. In the window that opens, click **Next**.
4. Click **Browse**, locate your workspace, and select **BankingProcessBPEL**. Click **Finish** to import the service project and the business process.

## 11.3.3 Generating the deployment code for the process

Now you create the deployment code for the business process.

1. In the Services view, expand **BankingProcessBPEL** → **com.ibm.oneida.bs.flow.makePayment** and select the **makePayment.bpel** business process file. Right-click the file, and select **Enterprise Services** → **Generate Deploy Code**, as shown in Figure 11-7.

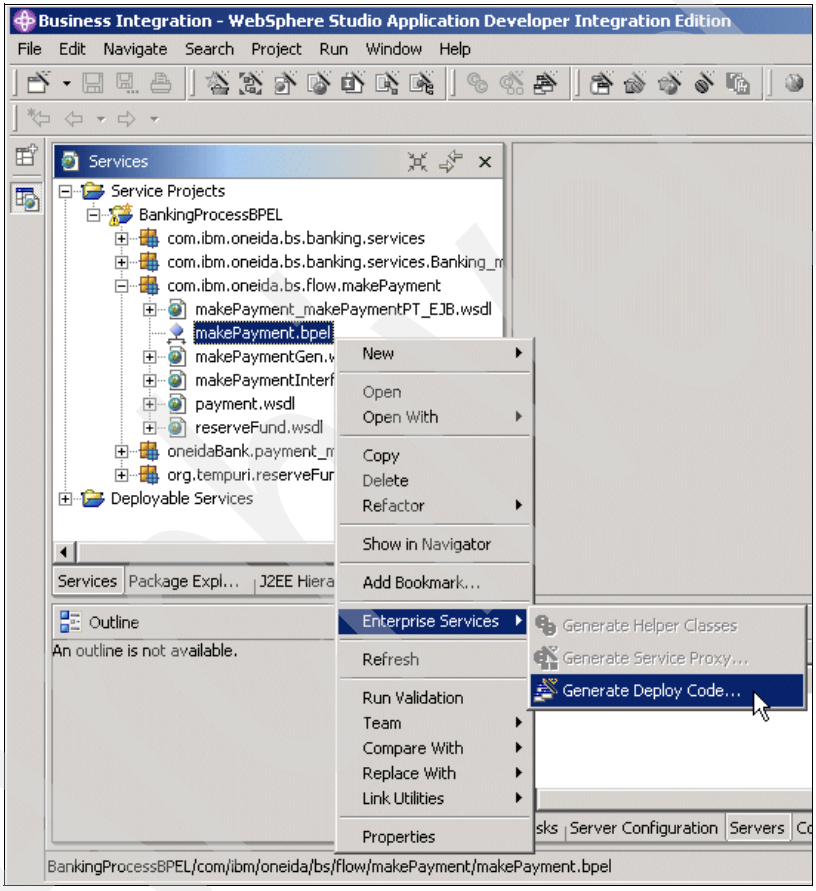



*Figure 11-7 Generating deploy code for the business process*

2. In the window that opens, click **OK**. This generates an EAR file for deployment.

## 11.4 Deploying the Spot Loan sample: the WebSphere for z/OS administrator perspective

This section explains how to create and define resources. It also explains how to deploy the enterprise and rules management applications into the application server. This is the WebSphere z/OS deployment administrator perspective.

In this scenario, we need the Enterprise Application Archive Files (EAR) listed in Table 11-4 to install the Spot Loan sample. This sample installation procedure covers the single server deployment.

*Table 11-4 Spot Loan sample resources*

| Resource | Description |
|---|---|
| BankingEAR.ear | Services provided by EJB |
| BankingProcessBPELEAR.ear | Choreography of those services |

The Spot Loan EAR files contain the application and choreography objects to be deployed.

This sample does not require additional resources, such as DB2, WebSphere MQ, CICS or IMS definitions, nor WebSphere security to be enabled. It is not necessary to change any security definitions on our security server.

**Note:** The propose of this sample is to show how to solve a simple business problem using a Process Choreographer process.

In a typical single server deployment scenario, a z/OS system programming or application technical support team has to perform the following tasks:

1. Deploy the applications objects, most likely the EAR files.
2. Configure a list of adapters, such as CICS or IMS adapters.
3. Define and configure resources such as database tables, JMS queues, and possibly business rules.

## 11.4.1 Deploying the Spot Loan sample enterprise application

To deploy the Spot Loan sample enterprise application, use the following steps:

1. On the WebSphere Application Server console, select **Applications** → **Install New Application**.
2. Click **Browse**, and select the **BankingEAR.file** on a local path as shown in Figure 11-3. Click **Next**.

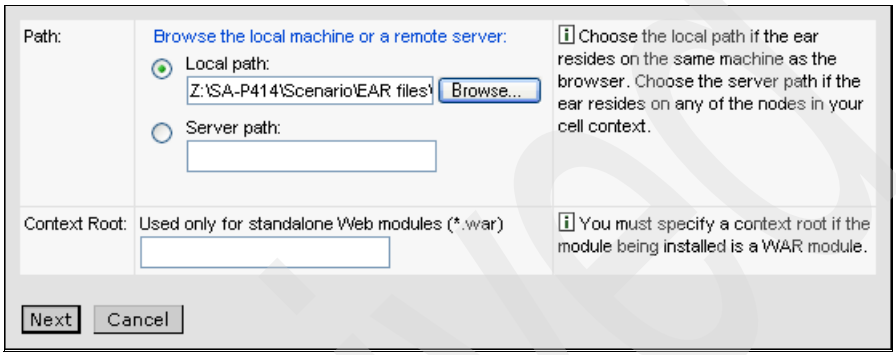



*Figure 11-8 Spot Loan: Browsing for Spot Loan enterprise project*

3. Select the **Deploy EJB** check box. Make sure that the Application Name is `BankingEAR`, as shown in Figure 11-9. Click **Next**.

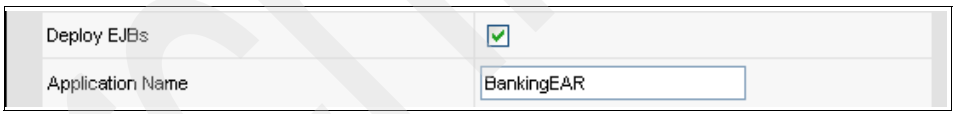



*Figure 11-9 Spot Loan: Choosing the application name*

> **Note:** We decided to deploy EJBs to make sure that all the bindings were compatible with the other WebSphere Application Server definitions.

4. In the Step 2 pane, for Database Type, select **DB2UDBOS390_V7** as shown in Figure 11-10. Click **Next**.


*Figure 11-10 Spot Loan: Selecting the database type*

5. Accept the default values in the Step 3 and Step 4 panes and click **Next** on each pane.

6. In the Step 5 pane, for our datasource JNDI name, specify `jdbc/BRBeans662` as shown in Figure 11-11.

> **Important:** This JNDI name provides the path to the rules engine database.

Click **Next**.

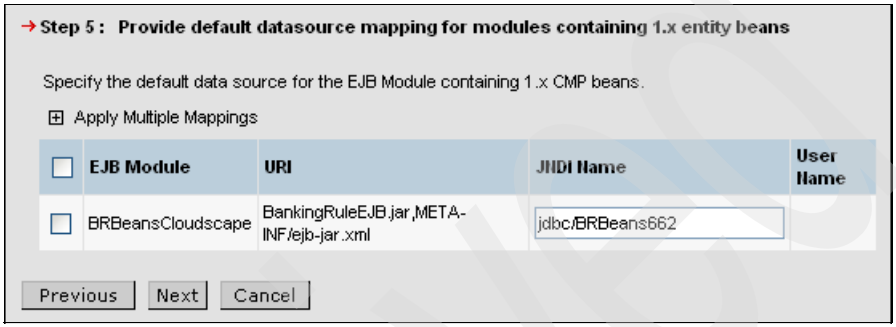



*Figure 11-11 Spot Loan: Specifying the datasource JNDI name*

7. For Steps 6 through 11, accept the default values and click **Next** panes.
8. At the end, you see a display like the example in Figure 11-12. It notifies you about the starting of the EJB deployment process. The installation process takes several minutes to complete.

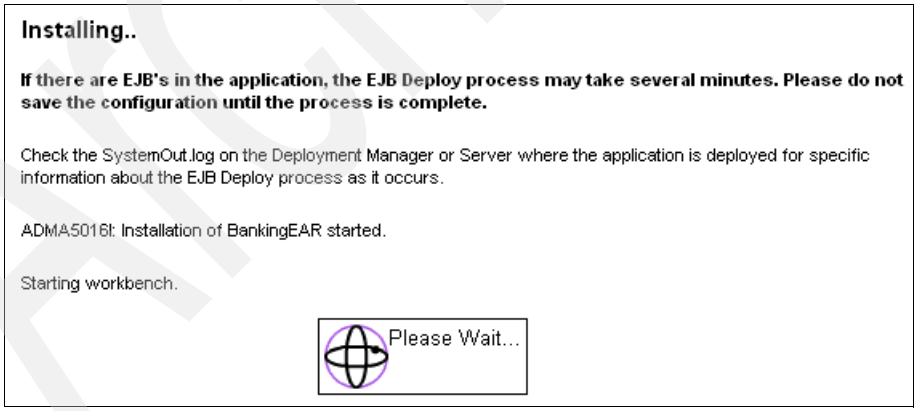



*Figure 11-12 Starting the EJB deployment process*

9. At the end of the installation, look in the log file to see whether any warning or error messages occurred during the EJB deployment process. As shown in

the example in Figure 11-13, you can see that there are no errors, warnings, or informational messages.

```
Check the SystemOut.log on the Deployment Manager or Server where the
application is deployed for specific information about the EJB Deploy
process as it occurs.
ADMA5016I: Installation of BankingEAR started.
Starting workbench.
Creating the project.
Building: /BankingRuleEJB.
Building: /BankingEJB.
Deploying jar BankingRuleEJB
Generating deployment code
Building: /BankingRuleEJB.
Invoking RMIC.
Generating DDL
Generating deployment code
Refreshing: /BankingEJB/ejbModule.
Building: /BankingEJB.
Invoking RMIC.
Writing output file
Shutting down workbench.
0 Errors, 0 Warnings, 0 Informational Messages
ADMA5005I: Application BankingEAR configured in WebSphere repository
ADMA5001I: Application binaries saved in
/WebSpherePA/V5R1M0/BS02/AppServer/wstemp/92668751/workspace/cells/cel662/ap
plications/BankingEAR.ear/BankingEAR.ear
ADMA5013I: Application BankingEAR installed successfully.
```

*Figure 11-13 Spot Loan: Sample enterprise application deployment log*

10. Click **Save to Master Configuration** to save the changes.
11. Click **Save** again to apply the changes.

You have now deployed the enterprise application.

## 11.4.2 Deploying the Spot Loan sample business rules

To deploy the Spot Loan sample business rules, follow these steps:

1. On the WebSphere Application Server console, select **Applications** → **Install New Application**.
2. Click **Browse** and select **BankingProcessBPELEAR.ear** on a local path. Click **Next**.
3. In the Step 1 pane, for Application Name, make sure that `BankingProcessBPELEAR` is specified as shown in Figure 11-14. Select the **Deploy EJBs** check box and then click **Next**.

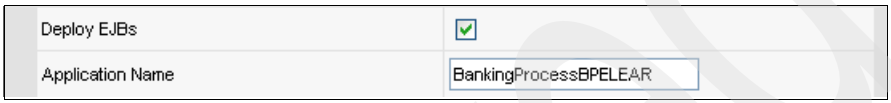



*Figure 11-14 Spot Loan: Defining the rules Application Name*

4. In the Step 2 pane, for Database Type, select **DB2UDBOS390_V7** as shown in Figure 11-10 on page 192. Click **Next**.
5. For Step 3 through Step 11, accept the default values and click **Next**.
6. Then you see a notification that the EJBs deployment process is starting (see Figure 11-12 on page 193).The EJBs deployment process takes a few minutes to finish.

7. Check the log after the EJBs deployment process to see if there are any warnings or error messages The example in Figure 11-15 shows that there aren't any warnings or error messages.

```
Check the SystemOut.log on the Deployment Manager or Server where the
application is deployed for specific information about the EJB Deploy
process as it occurs.
ADMA5016I: Installation of BankingProcessBPELEAR started.
BPEC0036I: Successfully bound service references of application
BankingProcessBPELEAR.
Starting workbench.
Creating the project.
Building: /BankingProcessBPELEJB.
Generating deployment code
Building: /BankingProcessBPELEJB.
Invoking RMIC.
Generating DDL
Shutting down workbench.
0 Errors, 0 Warnings, 0 Informational Messages
ADMA5005I: Application BankingProcessBPELEAR configured in WebSphere
repository
BPEC0031I: Completed updating the process choreographer databases with
processes for application BankingProcessBPELEAR.
ADMA5001I: Application binaries saved in
/WebSpherePA/V5R1M0/BS02/AppServer/wstemp/92668751/workspace/cells/cel662/ap
plications/BankingProcessBPELEAR.ear/BankingProcessBPELEAR.ear
ADMA5013I: Application BankingProcessBPELEAR installed successfully.
```

*Figure 11-15 Spot Loan: Sample rules application deployment log*

8. Click **Save to Master Configuration** to save the changes.
9. Click **Save** again to apply the changes.

You have now deployed the sample business rules.

## 11.4.3 Starting the Spot Loan sample application

After the enterprise and rules applications EJBs have been deployed successfully, we must start both applications.

1. From the WebSphere Application Server console, select **Applications** → **Enterprise Applications**.
2. Start the Spot Loan enterprise application. Select the **BankingEAR** check box, as shown in Figure 11-16. Click the **Start** button to start the application.

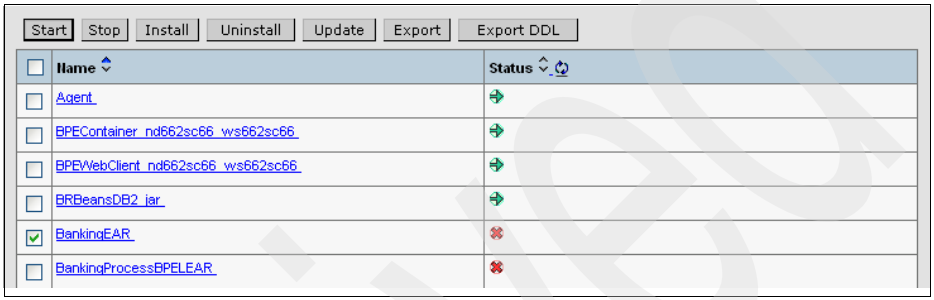



*Figure 11-16 Starting the Spot Loan enterprise application*

3. Select the **BankingProcessBPELEAR** check box, and then click the **Start** button to start the Spot Loan rules application.
4. Check to see that both applications were successfully started.

A green sign should appear in the status area for both applications indicating that they started successfully.

## 11.5 Deploying Spot Loan sample: the application support perspective

This section explains how to create and maintain the business management rules. This is the application support team perspective.

Use the rule management application to define the business rules.

1. Using the process outlined in 9.7, “Installing the Business Rules Management Client” on page 162, set up and start the rule management application.
2. Define the rules for the scenario as shown in Figure 11-17.
3. Close the rule management application by selecting **File** → **Close**.

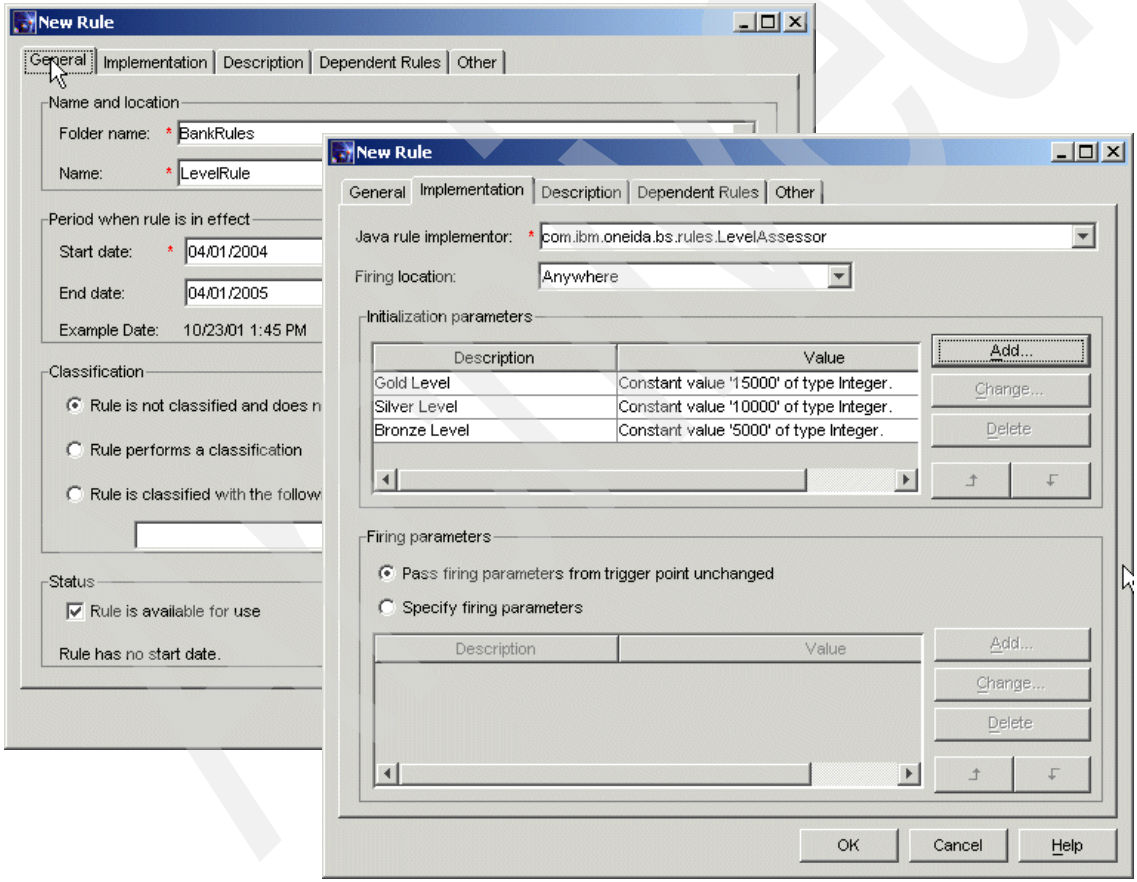



*Figure 11-17 Business rules for the Spot Loan scenario*

## 11.6 Starting the Spot Loan sample

Based on our business scenario, the Spot Loan sample, we need to handle a user request to transfer money from one account to another account. This business problem is solved by the implementing the makePayment choreographer process.

To run the Spot Loan sample, open a browser and launch the BP WebClient. In our example, we typed the following URL.

`http://wtsc66.itso.ibm.com:9015/bpe/webclient`

In this URL, note the following explanation:

- *wtsc66.itso.ibm.com* is our server host name.
- *9015* is the TCP/IP port that we are using.

> **Note:** We did not enable security for this sample. When we open the BP WebClient, in the top left corner of the window, we see that the user ID is UNAUTHENTICATED.

## 11.7 Spot Loan sample process: making a payment

Based on the Spot Loan business scenario, the process starts when a customer request a loan approval. To simulate this, we used the following steps:

1. In the left column of the WebClient GUI, select **My Templates**.

> **Important:** Process templates list contains a list of templates of all business processes that we have access implemented by Process Choreographer. This is not the same as a list of instances of business processes. An instance of a business process is created based on a particular Process Choreographer template (process template).

2. In the Template Name column, click **makePayment** (see Figure 11-18).

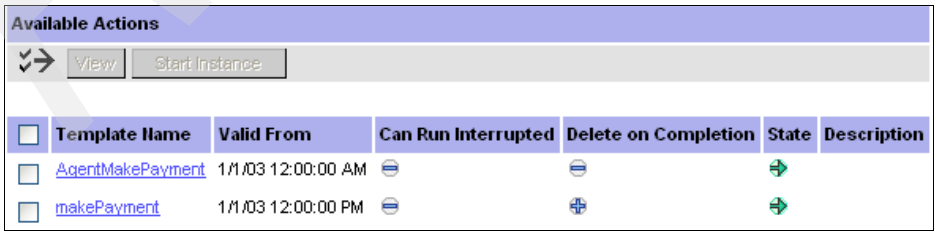



*Figure 11-18 Starting the makePayment business process*

3. In the Services area, click **makePayment**.
4. Enter the values required for the process and press Enter or click **Start Instance**, as shown Figure 11-19.

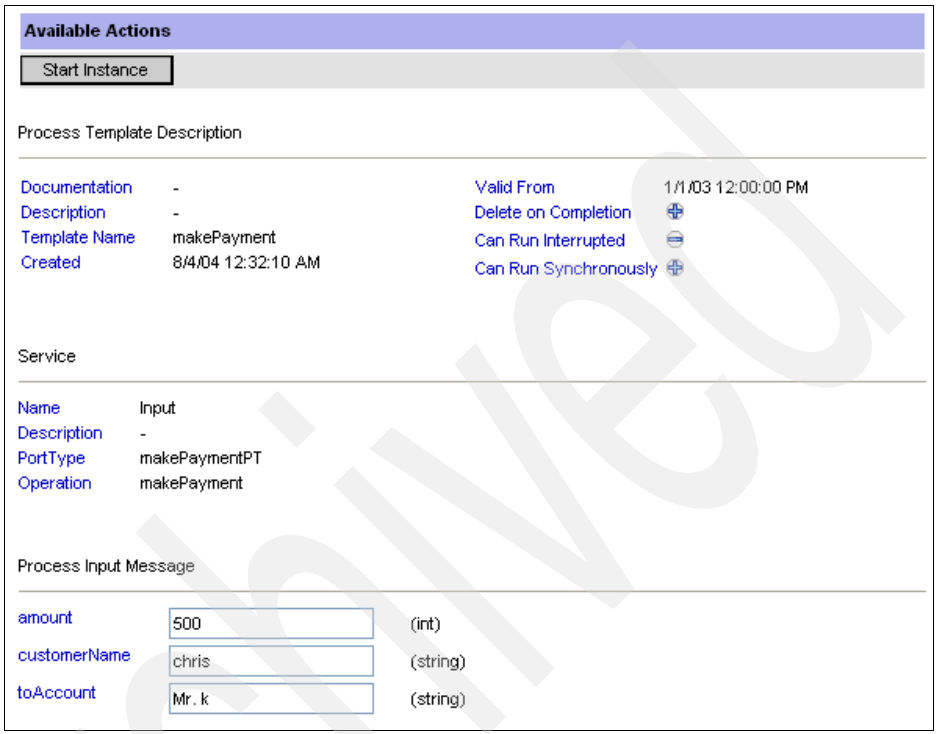



*Figure 11-19 Spot Loan: Starting the process instance*

The results depend on the values that you enter for the amount and customerName parameters. The toAccount parameter is not selected.

The customer data used to make the determination is hard-coded in the BankingEJB's BankingBean.java, as shown in Figure 11-20.

```
public class BankingBean implements javax.ejb.SessionBean {

private CustomerInfo customerList[] = {
// Customer Id, Account Balance, Credit Rating, Income
new CustomerInfo("chris", 100,   20000,   1000),
new CustomerInfo("linda",   5000,  10000, 100000),
new CustomerInfo("james", 10000, 1000,    500000)
};
private javax.ejb.SessionContext mySessionCtx;
/**
 * getSessionContext
 */
public javax.ejb.SessionContext getSessionContext() {
return mySessionCtx;
}
/**
 * setSessionContext
 */
public void setSessionContext(javax.ejb.SessionContext ctx) {
mySessionCtx = ctx;
}
```

*Figure 11-20 Spot Loan: CustomerInfo data implementation*

If the amount of money transferred is less than the account balance, the transfer is accepted automatically. If the amount is not less than the account balance, the credit rating value is compared to the rules defined by BankRules/LevelRule. If the customer rating is Gold, the loan is approved. Otherwise, the loan is rejected.

You can change these results by manipulating the rule management application console (see Figure 11-21).

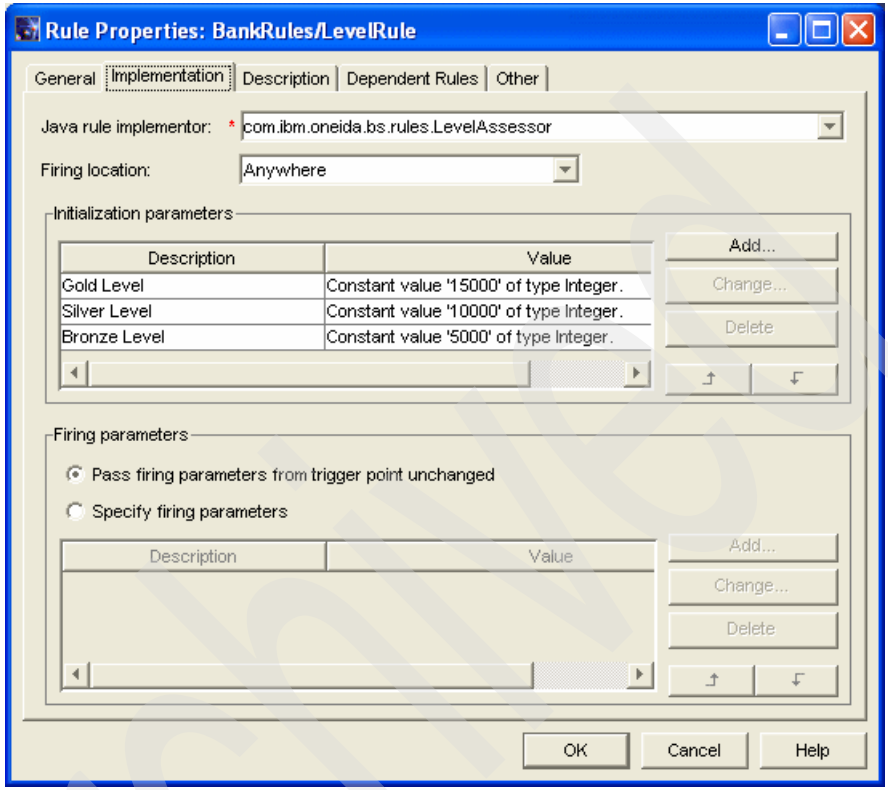



*Figure 11-21 Spot Loan: Business rules*

Based on the business rules in Figure 11-21, the customer named Chris (from Figure 11-19 on page 200) has a Gold Level rule, so his loan is approved. We receive the message `Payment transferred successfully`.

Table 11-5 shows other potential results for the loan applications.

*Table 11-5 Sample applications results*

| Customer ID | Amount transferred | Customer level | Loan approved or not approved |
|---|---|---|---|
| Chris | =>100 | Gold | Approved |
| Linda | =>5000 | Silver | Not approved |
| James | =>10000 | Bronze | Not approved |

12

# Applied Decomposition pattern with human interaction: Staff Assisted Loan scenario

This chapter describes a sample business scenario called Staff Assisted Loan for our imaginary client ITSO bank that uses the Decomposition pattern with human interaction. We outline the business scenario and present some design guidelines. We also explain how to prepare the sample application and the environment in which it will be deployed.

## 12.1 Business scenario

We continue with our imaginary client, the ITSO Bank, from Chapter 11, “Applied Decomposition pattern: Spot Loan scenario” on page 179. ITSO Bank wants to allow customers to obtain pre-approval for loans online. To improve customer satisfaction, the bank will allow customers to quickly check to see if a loan will be approved before the customer goes to the bank and completes a loan request.

Based on the ITSO Bank list of strategic business goals presented in Figure 12-1, the Staff Assisted Loan scenario will address:

- The business goal: To increase customer satisfaction
- A critical success factor: To provide quick and easy access to banking services

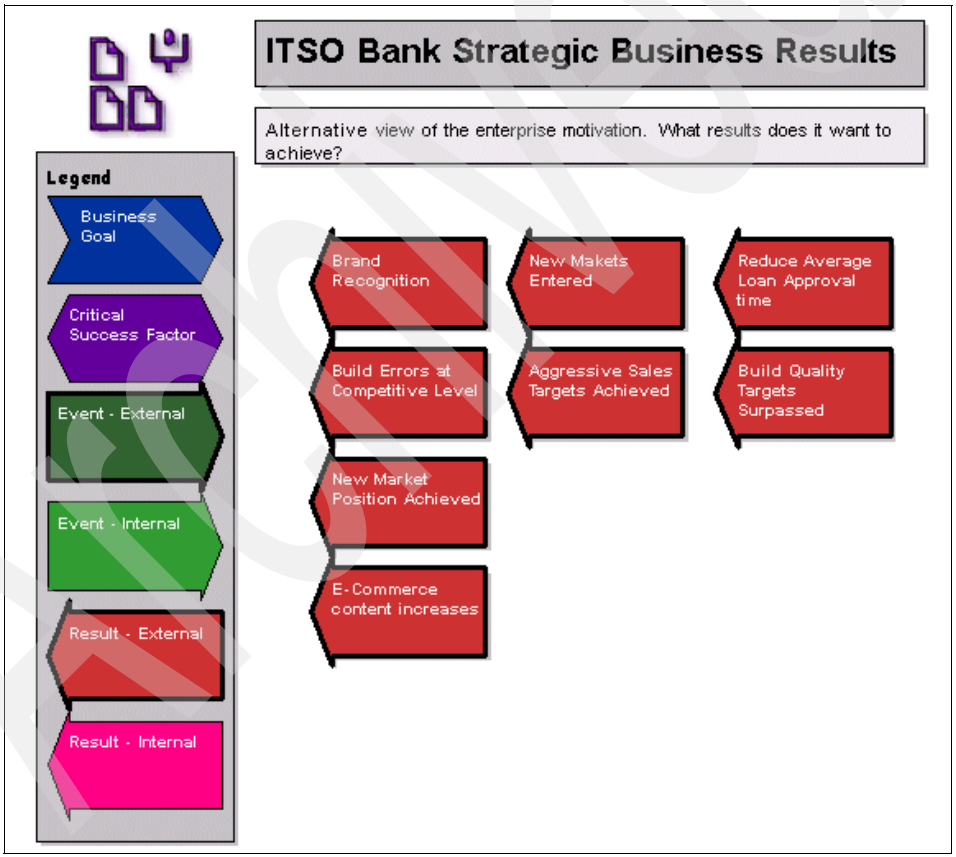



*Figure 12-1 ITSO Bank strategic business results*

## 12.2 Design guidelines

In this scenario, the same customer as in the Spot Loan scenario will have access the bank online and request a loan approval. When the request reaches the process, a risk assessment is done based on the customer's income. Two business situations can happen:

- If the customer has a low risk rating, the loan is approved automatically.
- If the customer has a medium or high risk rating, a chance is given for the loan manager to make a decision.

Some requests can be approved automatically. However, in some cases, human interaction in the form of a loan manager might be required.

### 12.2.1 Staff Assisted Loan scenario: pre-approval for loans

For this scenario, we identify the following actors:

- The customer (the same as in the Spot Loan scenario)
- The risk assessment service
- The loan manager

We can also identify an additional *use case*, the Staff Assisted Loan.

The customer actor details are provided in Table 11-1 on page 184. Table 12-1 provides details about the risk assessment service actor.

*Table 12-1 Risk assessment service actor details*

| | |
|---|---|
| **Actor name** | Risk assessment service |
| **Brief description** | An internal service that provides a loan risk assessment based on information the bank has about the customer |
| **Status** | Primary |
| **Relationships** | |
| **Associations to use cases** | 002 Staff Assisted Loan |

Table 12-2 provides details about the loan manager actor.

*Table 12-2 Loan manager actor details*

| | |
|---|---|
| **Actor name** | Loan manager |
| **Brief description** | A bank manager who is authorized to approve loans that might not be automatically approved |
| **Status** | Primary |
| **Relationships** | |
| **Associations to use cases** | 002 Staff Assisted Loan |

### 12.2.2 Use case 002: Staff Assisted Loan

Table 12-3 provides details about the Staff Assisted Loan use case.

*Table 12-3 Use case 002: Staff Assisted Loan*

| | |
|---|---|
| **Use case name** | 002 Staff Assisted Loan |
| **Subject area** | Loan management |
| **Business event** | A customer requests a loan approval online. |
| **Actors** | Customer, risk assessment service, loan manager |
| **Use case overview** | The customer requests pre-approval of a loan. Information about the customer is passed to the risk assessment service, which uses it to assess the credit risk.<br><br>If the risk is low, the banking service approves the loan. If the risk is not low, the request is passed to a loan manager who has the authority to approve or reject the loan. |
| **Preconditions** | The customer supplies the amount of the loan requested. |
| **Termination outcome 1** | The outcome of the request is recorded for use by a loan officer. |

Figure 12-2 illustrates the use case model for this scenario.

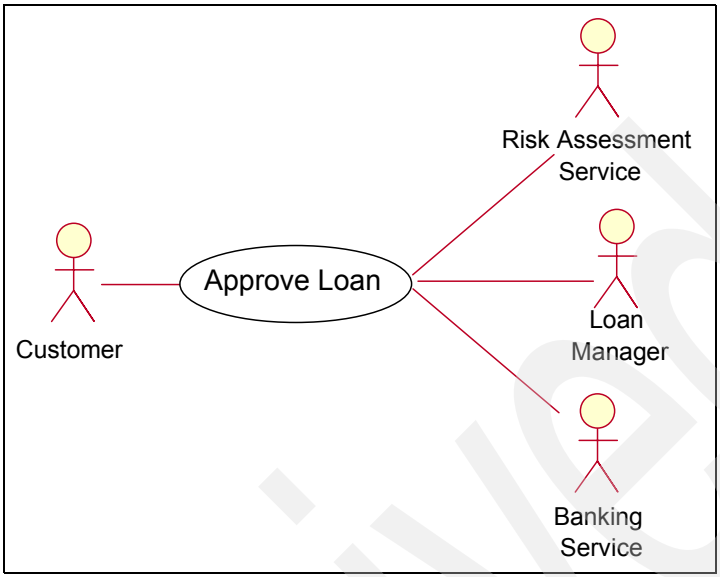



*Figure 12-2 Staff Assisted Loan scenario use case model*

We can also use the traditional business process methodology to describe our sample business scenario as shown in Figure 12-3.

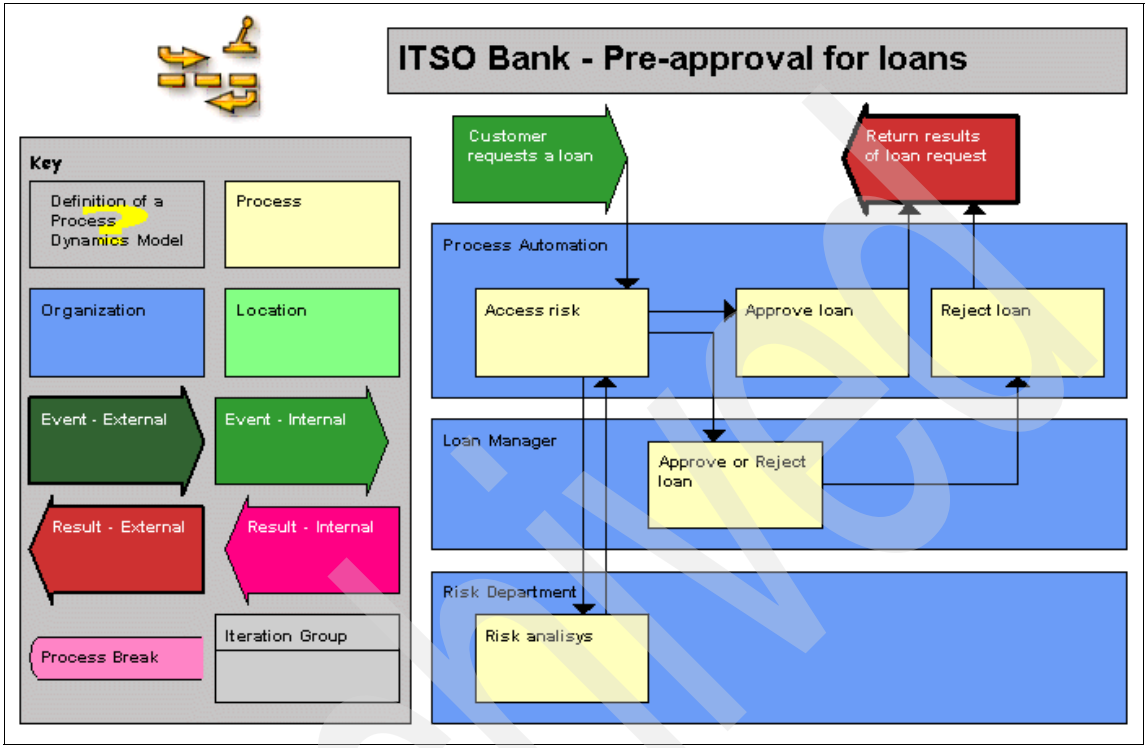



*Figure 12-3 ITSO Bank pre-approval for loans*

The business process involves the following flow:

1. The customer logs into the bank portal and requests a loan. The input is:
   a. Customer name
   b. Amount of the loan
2. A risk assessment is performed based on the customer's income level by the risk assessment department.
   – If the customer has a low risk rate, the approval is automatically granted.
   – If the customer is high or medium risk, the bank loan officer is allowed to make the decision to approve or reject the loan.

In this business scenario, there is a need for human activity, making this an interruptible process. For more information about interruptible and non-interruptible processes, see 7.1, "Non-interruptible versus interruptible processes" on page 96.

Keep in mind that no parallel processing is required. All the information about the request is obtained directly from the Spot Loan scenario customer and from the data available directly to the process.

> **Important:** The process results are determined using business rules, and potentially from input by the loan manager.

### 12.2.3 Application pattern selected

This scenario can be implemented using the Decomposition pattern with a human interface. The introduction of human interaction into the process is within the capabilities of the process manager.

## 12.3 Preparing the Staff Assisted Loan sample

To prepare the application for deployment, you need to set up the development environment, import the business process, and generate deploy code for the process. This is the Staff Assisted Loan sample deployment process from the developer perspective.

Like the Spot Loan sample, the Staff Assisted Loan scenario also operates in a common WebSphere Studio Application Development environment.

> **Attention:** You must install WebSphere Studio Application Developer Integration Edition 5.1 fixpack 2 to use this sample.

To set up the Staff Assisted Loan sample in the development environment, you must perform the following steps:

1. Import the existing banking enterprise applications.
2. Import the business process.
3. Generate deploy code for the process.
4. Define the business rules.

## 12.3.1 Importing the existing banking applications

To import the existing banking applications, use these steps:

1. Open the WebSphere Studio workbench.
2. Select **File** → **Import** and select the **EAR file**. Click **Next**.
3. In the Enterprise Application Import window, complete the following actions (see Figure 12-4):
   a. For EAR File, click **Browse** and select the **BankingStaffEAR.ear** file.
   b. For Project name, specify `BankingStaffEAR`.
   c. For Target server, select **Integration Server v5.1**.
   d. Click **Finish**.

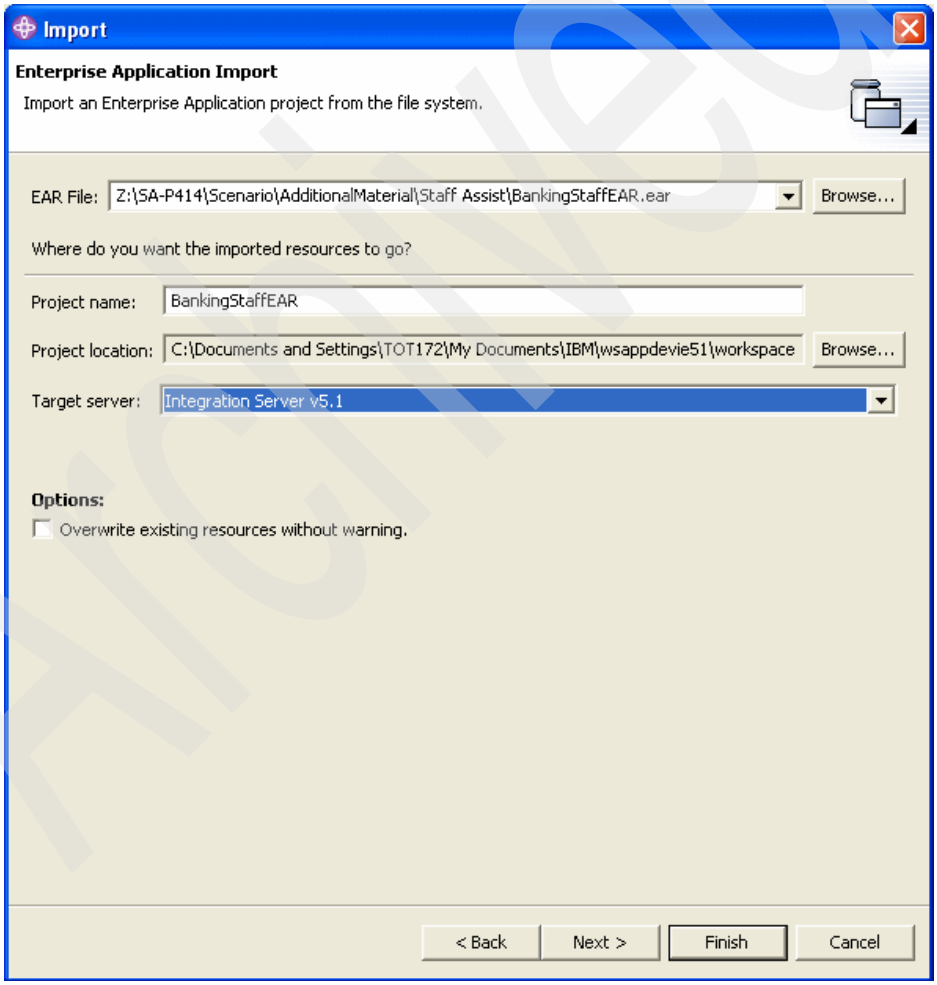



*Figure 12-4 Importing the Staff Assisted Loan enterprise application*

4. In the Project Navigator view, select **BankingRuleStaffEJB**, right-click, and select **Properties**.
5. In the Properties window, select **Java Build Path** and complete the following steps:
   a. Click the **Libraries** tab.
      i. Click **Add Variables.**
      ii. Select **WAS_EE_V51**. Click **Extend**.
      iii. From the lib folder, select **brbRuleMgmtApp.jar**, as shown in Figure 12-5. Click **OK**.
   b. Select the **Project** tab.
      i. Select **BankingEJB** to add it to your Java Build Path.
      ii. Click **OK**.

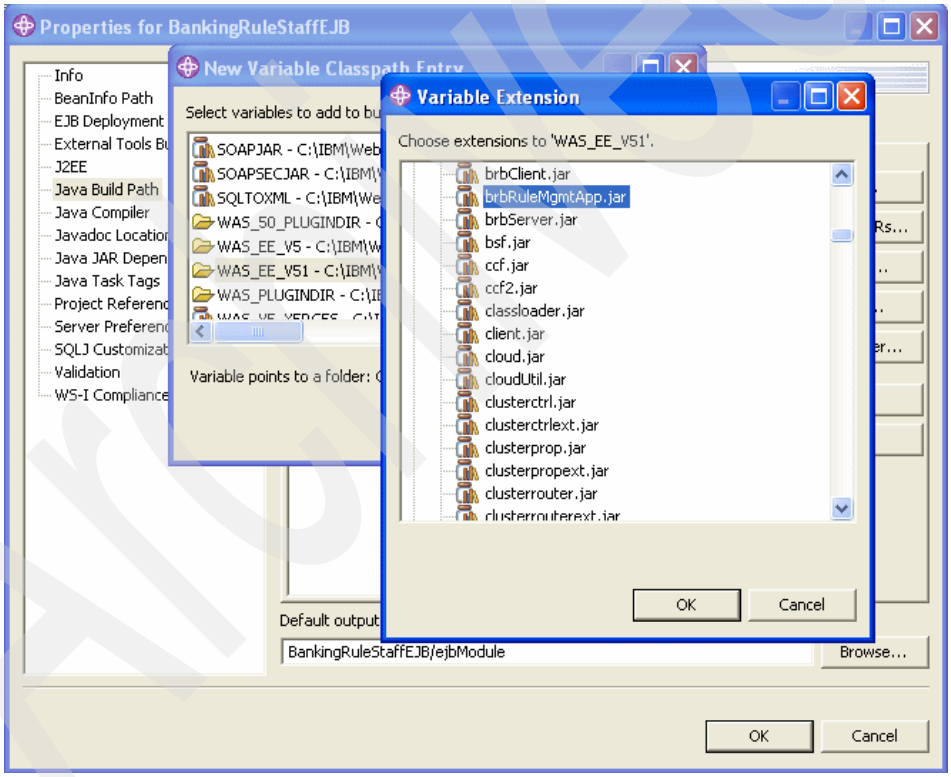



*Figure 12-5 Adding brbRuleMgmtApp*

You have now imported the existing banking applications.

### 12.3.2 Importing the business process

Next you need to import the business process into the service project that you created.

1. Unzip the BankingStaffBPEL.zip file directly into the workspace.
2. In WebSphere Sudio, select **File** → **Import** → **Existing Project into Workplace**.
3. In the window that opens, click **Next**.
4. In the next window, click **Browse**, locate your workspace, and select **BankingStaffBPEL**. Click **Finish**.

### 12.3.3 Generating deployment code for the process

Now create the deployment for the business process.

1. In the Services view, expand **BankingStaffBPEL** → **com.ibm.oneida.bs.flow.approveLoan**. Select the **approveLoan.bpel** business process file, right-click, and select **Enterprise Services** → **Generate Deploy Code**, as shown in Figure 12-6.

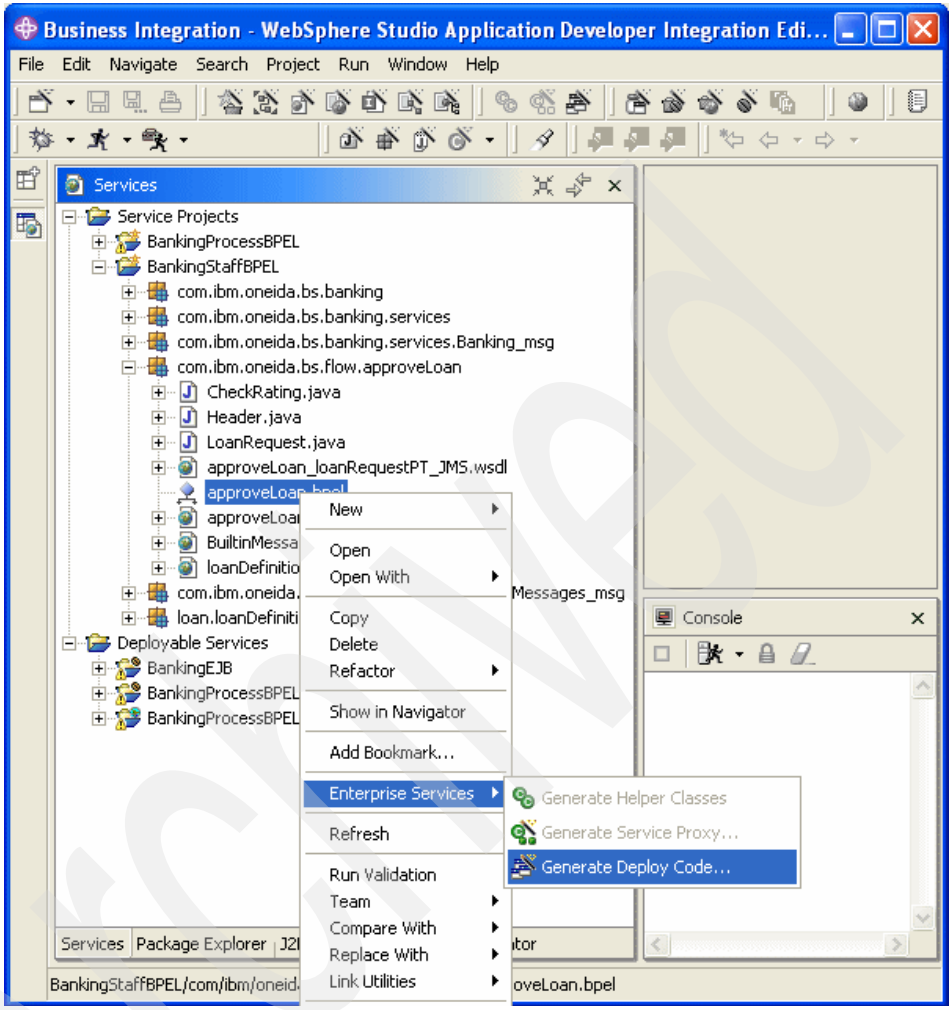



*Figure 12-6 Generating the deploy code for the approve loan process*

2. In the message window that opens, click **OK**.

An EAR file is generated for deployment. This operation takes a few minutes.

## 12.4 Preparing the Staff Assisted Loan sample environment

To deploy the Staff Assisted Loan sample into WebSphere Application Server, you must perform the following actions:

1. Turn on security for WebSphere Application Server.
2. Define the security policies for EJB.
3. Deploy the applications to the server.

This section concentrates on the first two tasks. To create and define resources, and to deploy the enterprise and rules management applications into the WebSphere Application Server, see 11.4, “Deploying the Spot Loan sample: the WebSphere for z/OS administrator perspective” on page 191.

### 12.4.1 Turning on security for WebSphere Application Server

To run the Assisted Loan sample, we need to turn on security for WebSphere Application Server.

1. On the WebSphere Application Server console, select **Security** → **Global Security**.
2. Select the **Enabled** check box to enable security for this WebSphere domain.
3. Deselect the **Enforce Java 2 Security** check box.

> **Attention:** If Java 2 Security is enabled and the application policy file is not set up correctly, the application may fail to run.

4. Specify the Active Protocol, Active Authentication Mechanism, and Active User Registry as shown in Figure 12-7.

> **Note:** For the purpose of this sample, we used Resource Access Control Facility (RACF) as our security server repository.

5. Click **OK**.
6. Click **Save** to apply save changes.
7. Click **Save** again to apply changes to the master configuration.
8. Restart the WebSphere Application Server to activate the global security changes.

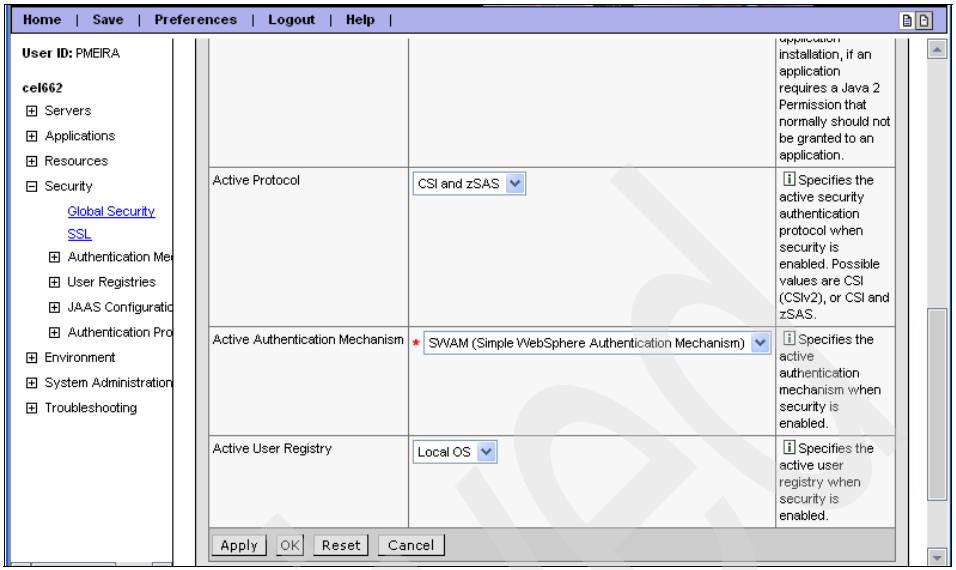



*Figure 12-7 Activating security for WebSphere Application Server*

## 12.4.2 Defining the security policies for EJBs

For the purpose of this sample, there are two options for defining EJB security policies:

- Let WebSphere Application Server manage EJB security policies itself.
- Define and activate the EJB security policies in the RACF.

We selected the first option to deploy this sample.

### EJB security policies managed by WebSphere for z/OS

You need to deactivate EJB SAF authorization.

1. From the WebSphere Application Server Admin Console, select **User Registries** → **Local OS**.
2. Deselect **Ignore Case**, if you don't consider case-sensitive user IDs and passwords.
3. From Additional Properties pane, select **Custom Properties**.
4. Under General Properties (Figure 12-8), complete the following information:
   a. For Name, select **com.ibm.security.SAF.authorization**.
   b. Change the Value field to **False**.
   c. Click **OK**.

5. Click **Save** to apply save changes.
6. Click **Save** again to apply changes to the master configuration.

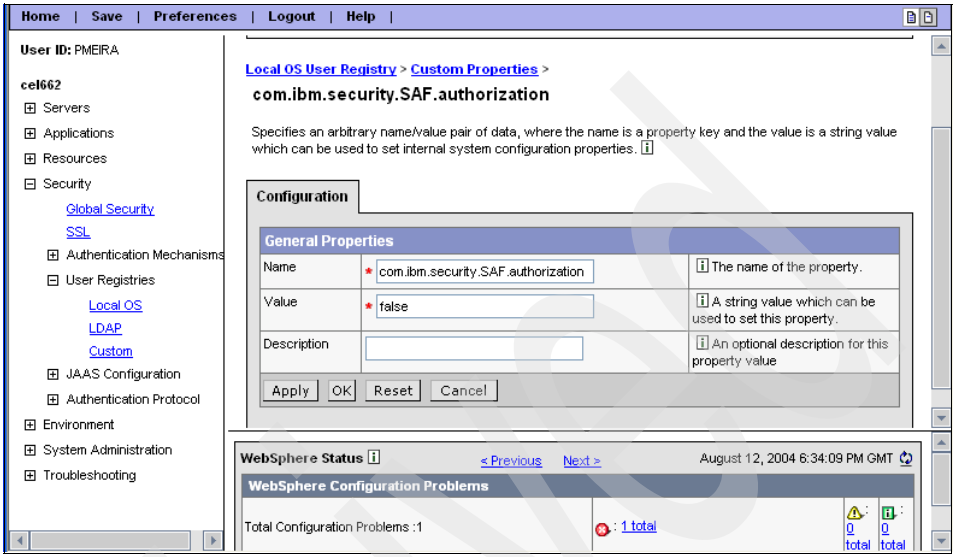



*Figure 12-8 Disabling SAF authorization*

## Defining and activating the RACF EJB security policies

Instead of using the WebSphere EJB security policies, you can define the EJB policies in RACF.

1. Set up WebSphere EJB roles in RACF, as shown in Example 12-1.

*Example 12-1 Sample RACF commands to define WebSphere EJB roles*

```
SETROPTS CLASSACT(EJBROLE)

RDEFINE EJBROLE [domain_prefix.]administrator UACC(NONE)
RDEFINE EJBROLE [domain_prefix.]monitor UACC(NONE)
RDEFINE EJBROLE [domain_prefix.]configurator UACC(NONE)
RDEFINE EJBROLE [domain_prefix.]operator UACC(NONE)

PERMIT [domain_prefix.]administrator CLASS(EJBROLE) ID(default_WS_CFG_group)
ACCESS(READ)
PERMIT [domain_prefix.]monitor CLASS(EJBROLE) ID(default_WS_CFG_group)
ACCESS(READ)
PERMIT [domain_prefix.]configurator CLASS(EJBROLE) ID(default_WS_CFG_group)
ACCESS(READ)
PERMIT [domain_prefix.]operator CLASS(EJBROLE) ID(default_WS_CFG_group)
ACCESS(READ)
```

```
RDEFINE EJBROLE [domain_prefix.]CosNamingRead UACC(READ)
RDEFINE EJBROLE [domain_prefix.]CosNamingWrite UACC(READ)
RDEFINE EJBROLE [domain_prefix.]CosNamingCreate UACC(READ)
RDEFINE EJBROLE [domain_prefix.]CosNamingDelete UACC(READ)

PERMIT [domain_prefix.]CosNamingRead CLASS(EJBROLE)
ID(default_WS_unauth_userid) ACCESS(READ)

SETROPTS RACLIST(EJBROLE) REFRESH
```

In Example 12-1, note the following explanations:

– *domain_prefix* is the name of your security domain if you have defined it in the Admin Console (see Figure 12-9) by using the following steps.

  i. Select **Security** → **Global Security**.
  ii. Select **Custom properties**.
  iii. Under General Properties, for Name, type `security.zOS.domainName`.

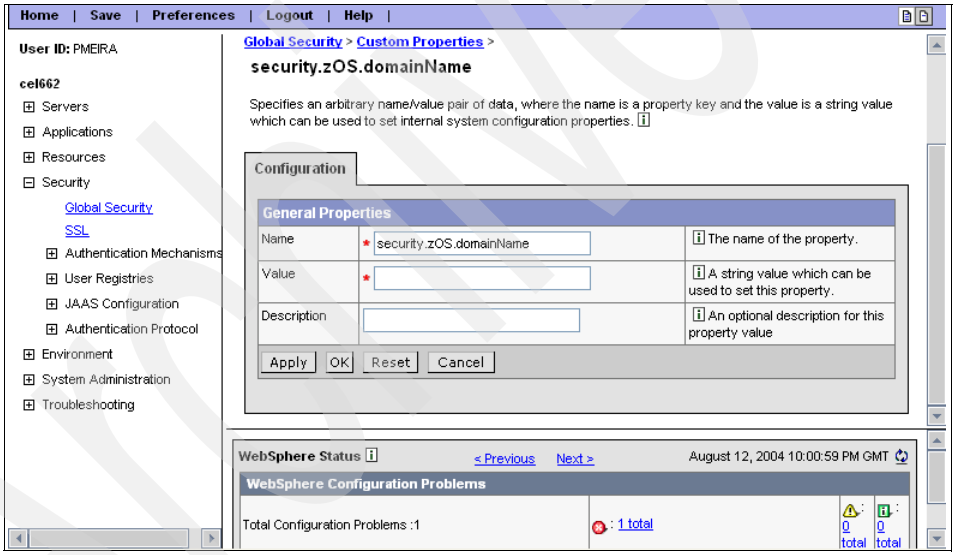



*Figure 12-9 Setting up the z/OS security domain name*

– *default_WS_CFG_group* is the WebSphere Application Server configuration group name; check the group name owning your configuration files in the hierarchical file system (HFS).

– *default_WS_unauth_userid* is the local default identity. To see your local default identity:

  i. Select **Security** → **Global Security**.
  ii. Select **z/OS Security Options**.
  iii. Under General Properties, specify a value for Local Identity as shown in Figure 12-10.

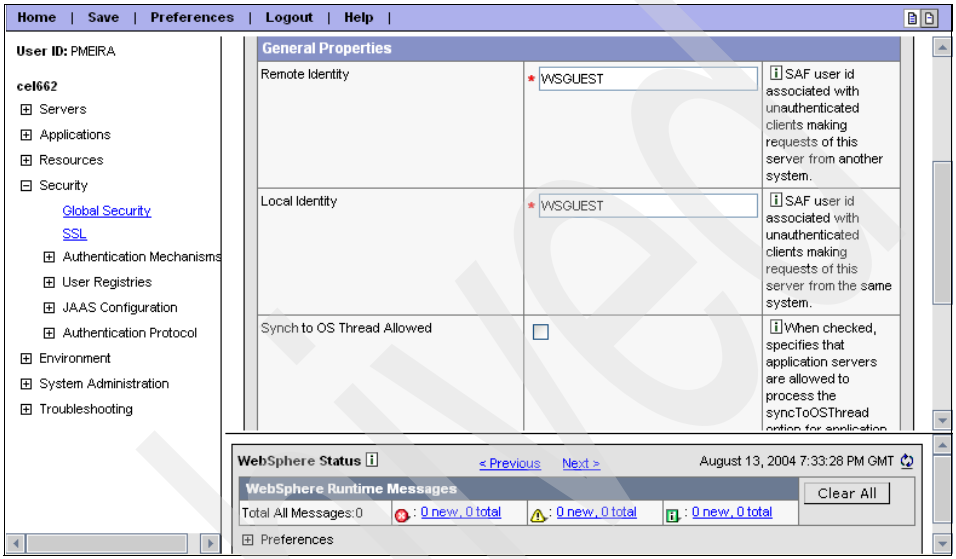



*Figure 12-10 Changing local default identity*

2. Set up the sample application roles in RACF. Create the RACF EJBROLE profiles for RuleManager, RuleUser. In our environment, we defined the users as shown in Example 12-2.

*Example 12-2 RACF profiles for Business Rule Beans*

```
PERMIT RuleManager CLASS(EJBROLE) ACCESS(READ) ID(PEDRO)
PERMIT RuleUser    CLASS(EJBROLE) ACCESS(READ) ID(BARTV)
```

13

# Run-time scenario on z/OS Parallel Sysplex

In this chapter, we illustrate how we can map a Self-Service runtime pattern to a z/OS Parallel Sysplex environment. We document our architectural design decisions for leveraging the requirements of the business process scenario, for example high availability and scalability.

Based on our imaginary client, the ITSO Bank, from Chapter 11, “Applied Decomposition pattern: Spot Loan scenario” on page 179, we develop and describe a more complex business scenario: the Optimized Spot Loan. First we describe the Runtime pattern nodes for scalability and high availability and list the details about our specific test environment. Then we explain the deployment process of the sample application into a z/OS Parallel Sysplex environment with WebSphere Business Integration Server Foundation (WBISF) V5.1.

To implement CICS business logic into a Web services-based business process in a Parallel Sysplex, you need to generate a CICS Web service first, and then integrate the Web service into the business process, from which you can generate the deployment code. We explain these tasks step-by-step and help you to prepare the infrastructure in connection with CICS.

## 13.1 Business scenario

ITSO Bank has recently acquired one of its competitors, ZYX Finance, which will now also operate under the ITSO Bank label. Some business drivers for this merger are to:

- Improve and extend service to the customer
- Reduce operational costs and improve efficiency

### 13.1.1 Business goals

These business drivers have been translated into the following strategic business decisions:

- IT infrastructure will be consolidated into a single environment.
- ZYX Finance currently does not offer self-service functions to its customers. The ITSO Bank self-service facilities will now be enabled across all business entities in the new group.
- The Spot Loan business process that existed at ITSO Bank will be optimized according to business logic that is available from ZYX Finance.
- The Spot Loan application availability to customers must be increased to a 24x7 service level.

These decisions imply the need for a scalable and high availability infrastructure.

### 13.1.2 Business environment

While ITSO Bank is already running WBISF for z/OS, the applications at ZYX Finance are all based on CICS Transaction Server for z/OS. The operational implementation of the strategic business decisions entails the following actions:

- The zSeries servers of ZYX Finance will be merged into the z/OS Parallel Sysplex of ITSO Bank. Each member of the new Parallel Sysplex setup will be able to run both the WebSphere Application Server profiles and the CICS Transaction Server regions.
- The CICS components that implement the business logic for optimized funding of payments will be integrated as a service in the Spot Loan business process, to leverage the improved cost effectiveness of the Spot Loan self-service.
- The WebSphere for z/OS setup will be reviewed to accommodate a scalable solution for the expected increase in workload and to guarantee high availability for 24x7 operations.

### 13.1.3 Adapting the business process model

ITSO Bank has audited the business processes at ZYX Finance. The audit found that the ZYK Finance equivalent of the Spot Loan process can be much more efficient for some kinds of payments. For payments that are transferred to an account within the same financial group, ZYX Finance does not create the overhead of reserving funds for the payment. The business audit has shown that this generates a major reduction of operational costs.

### 13.1.4 Pattern selected

Our business scenario requires a run-time implementation of the Decomposition Self-Service application pattern that leverages scalability and high availability. The high availability and scalability solution provided by WebSphere Application Server from Version 5 onward is *clustering*. z/OS fully supports this solution.

## 13.2 Design guidelines

The business logic that implements the decision whether funding for a Spot Loan can be optimized exists as a CICS application. We therefore extend the existing Spot Loan business process described in 11.2.1, “Spot Loan scenario: online funds transfer with Spot Loan” on page 184.

### 13.2.1 Optimized Spot Loan scenario

The CICS application is considered as new actor in the business process, the *Funds Optimizer Service*. It verifies the target account for the payment. If it is an account within the ITSO Bank group, then it indicates that no reservation of funds is required to process the loan request.

Table 13-1 lists the characteristics for the Funds Optimizer Service.

*Table 13-1 Funds Optimizer Service actor details*

| | |
|---|---|
| **Actor name** | Funds Optimizer Service |
| **Brief description** | The Funds Optimizer Service provides decides if reservation of funds is required for processing the loan request. |
| **Status** | Primary |
| **Relationships** | |
| **Associations to use cases** | 004 Optimized Spot Loan |

We identify the new use case the Optimized Spot Loan. Figure 13-1 illustrates the adapted business process, in which the specific activities (represented as swim lanes) for this use case have been highlighted.

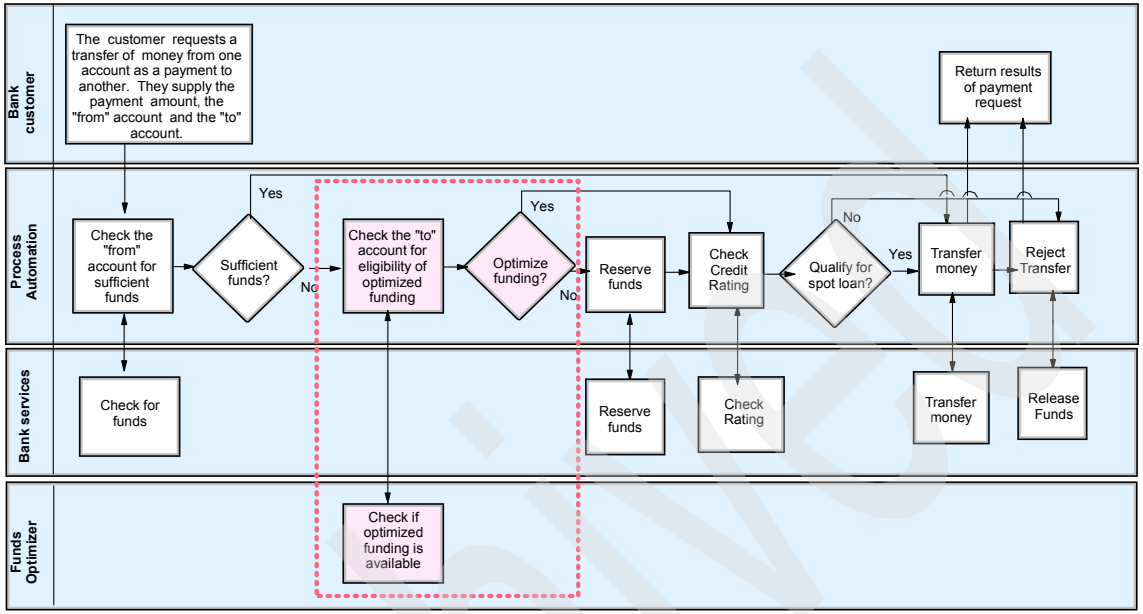



*Figure 13-1 Swimlanes diagram for the Optimized Spot Loan use case*

### 13.2.2 Use case 004: Optimized Spot Loan

The use case is summarized in Table 13-2.

*Table 13-2 Optimized Spot Loan use case*

| | |
|---|---|
| **Use case name** | 004 Optimized Spot Loan |
| **Subject area** | Online funds transfer |
| **Business event** | A customer transfers money from one account to another to manage account balances or to pay a bill. |
| **Actors** | Customer, credit rating service, banking service, funds optimizer service |
| **Use case overview** | The customer requests a transfer of money from one account to another. If the account has sufficient funds, the banking service transfers the money. If the account doesn't hold sufficient funds, the funds optimizer service will decide whether to reserve funds for a loan. If necessary, the bank will reserve the funds. The bank uses the customer's pre-calculated credit rating held on file as input to the credit rating service, which determines if the customer meets the requirements for a spot loan.<br><br>If the credit rating is high enough, the loan is approved, and banking services transfers the money from the reserved bank funds to the account, or directly if optimized funding was decided. If the loan is not approved, the reserved bank funds are released, if any. |
| **Preconditions** | The customer supplies the amount of funds, the account to transfer from, and the account to transfer the money to. |
| **Termination outcome 1** | The bank service transfers the money and records the transaction. |

Before continuing with the design and implementation of this use case, we describe the Parallel Sysplex setup that will leverage the business decisions presented in 13.1, "Business scenario" on page 220.

## 13.3 Runtime pattern mapping for scalability and availability in a z/OS Parallel Sysplex

The objective of a high availability implementation is to eliminate single points of failure in the architected solution. The z/OS Parallel Sysplex is designed with this concern specifically in mind.

A high level approach to the elimination of single points of failures in an application architecture is mapped to the z/OS environment in Table 13-3.

*Table 13-3 Elimination of single points of failure in a z/OS Runtime pattern implementation*

| Runtime pattern node | Single point of failure elimination | z/OS run-time solutions in Parallel Sysplex |
|---|---|---|
| Web server or Web server redirector | Network path redundancy up to Web server (redirector) | If this node is implemented on z/OS, then two logical partitions (LPARs) are required. These LPARs must be separated from the LPARs running the application server. In 13.4, “Overview of the Parallel Sysplex deployment environment” on page 226, we use two Linux for zSeries partitions. |
| | At least two HTTP servers on dedicated hardware platforms | |
| Network load balancer | Hardware (router) or software solution | z/OS provides Sysplex Distributor software as a unique feature to work with z/OS Workload Manager (WLM) and dynamic virtual IP addresses (DVIPAs). Intra-Sysplex TCP/IP traffic can use the high-performance zSeries HiperSocket feature. |
| | Must support both HTTP and IIOP protocols | |
| Application server | Network path redundancy up to application server | |
| | Cluster of two cloned application servers | WebSphere Application Server for z/OS clusters are WLM-enabled, through the specific design of control and scalable servant regions. |
| Process manager | WBISF Business Process Choreographer requires hot standby database and queue manager | WebSphere BI Server Foundation on z/OS can exploit DB2 Data Sharing in a sysplex for the BPC database, giving each hardware platform access to the data. The storage media can be mirrored using PPRC. |
| Existing applications and data | Duplication of hardware and software in hot standby | Traditional z/OS resource and application managers CICS, DB2, IMS, and WebSphere MQ, and RRS, are fully sysplex-enabled using coupling facility features. |

The z/OS specific features that are listed in Table 13-3 are discussed in Chapter 5, “Selected patterns for e-business on zSeries” on page 65, and in

Chapter 8, “Value of WBISF on z/OS” on page 117. Also see Chapter 14, “Best practices for high availability on z/OS” on page 255, in which we briefly summarize how the specific z/OS features map onto the high availability run-time requirements.

A Parallel Sysplex for z/OS uses the unique hardware feature of a coupling facility to which all the z/OS systems in the sysplex, local or remote, are attached. Usually this coupling facility is a separate hardware, and a second one is set up for failover support. All z/OS subsystems, which are set up to participate in a sysplex sharing mode, use the high performance coupling facility to exchange and lock information about current work in the sysplex. It is the core of sysplex high availability.

In addition, the orchestrator for scalability on the z/OS platform is the Workload Manager. It is a base component of z/OS that is embedded in the operating system. WLM manages the dispatch priorities of all the transactions and other workloads on the system, delivering a service level that meets the performance goals that have been set. It can increase dynamically the number of application servers if this contributes to achieving these goals.

Plus, the service goals are assigned per individual WebSphere transaction, through the means of a *WLM enclave*. The enclave guarantees that all the work that is being run from WebSphere on behalf of a client request, including business processes, is classified under a single WLM manageable object for that request.

**Important:** WLM manages the priorities of dispatchable work units to achieve service-level goals. However the priorities are managed at the micro-level of WebSphere transactions, not at the macro-level of WebSphere Application Server. This is why on z/OS, a WebSphere Application Server is capable of running applications that have different performance profiles. You will often find a mixture of applications deployed in one application server. Manageability of workload service levels is not likely to be the major driver for deploying applications to dedicated servers on z/OS.

For more information about the setup of WLM, see Chapter 10, “Workload management on z/OS” on page 165. You can find details about the complete z/OS and WebSphere setup in *Enabling High Availability e-business on @server zSeries,* SG24-6850.

## 13.4 Overview of the Parallel Sysplex deployment environment

This section describes our specific Parallel Sysplex environment in which we deployed our sample application. Figure 13-2 shows the Parallel Sysplex members.

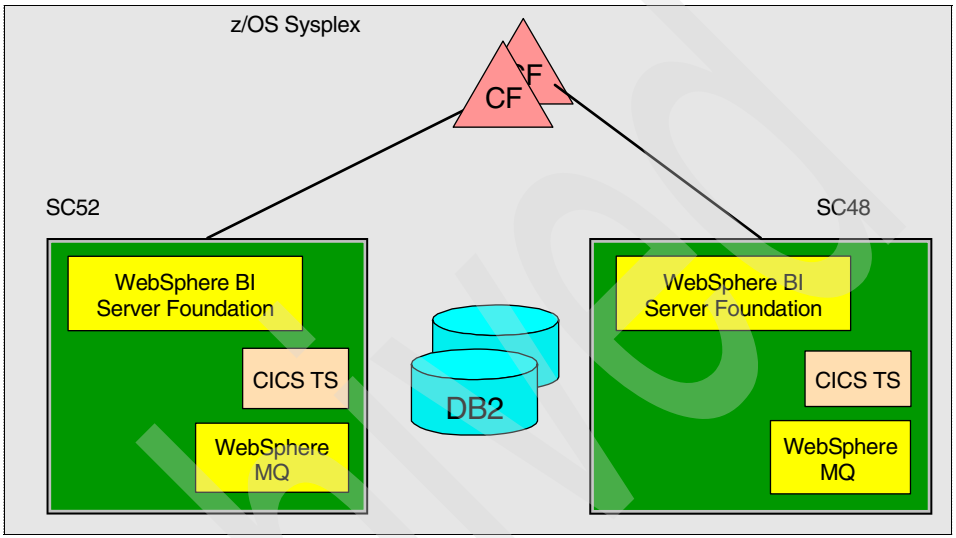



*Figure 13-2 Parallel Sysplex members in the run-time setup*

The systems SC52 and SC48 are on a zSeries 990 (z990) Model 2084-B16, while SC66 is on a zSeries 900 (z900) Model 2064-1C7. All LPARs are defined with 4 GB memory and run z/OS V1.5. You can find more details about our test configuration in Appendix A, “ITSO test configurations” on page 273.

### 13.4.1 Network Deployment configuration

The workload balancing mechanism for WebSphere Application Server is based on *clustering* over different application server nodes. We installed a Network Deployment configuration of WBISF. See the redbook *WebSphere Business Integration Server Foundation V5.1 on z/OS* SG24-6382, for details about the setup of such an infrastructure.

As shown in Figure 13-3, our infrastructure consists of:

- One cell, called *sdcell*, that spans the two sysplex LPARs SC52 and SC48
- One node on each LPAR: *sdnodea* on SC52 and *sdnodeb* on SC48
- On each node, one active application server: *sdsr03a* on SC52 and *sdsr03b* on SC48
- One cluster defined across both LPARs: *sdsr03cluster*, which includes both application servers *sdsr03a* and *sdsr03b*
- On SC52, the node *sddmnode*, which runs the Deployment Manager

**Note:** On z/OS, all servers have a *long name* and a *short name*. In this text, the long names are used. In Figure 13-3, the servers are labelled by their short names.

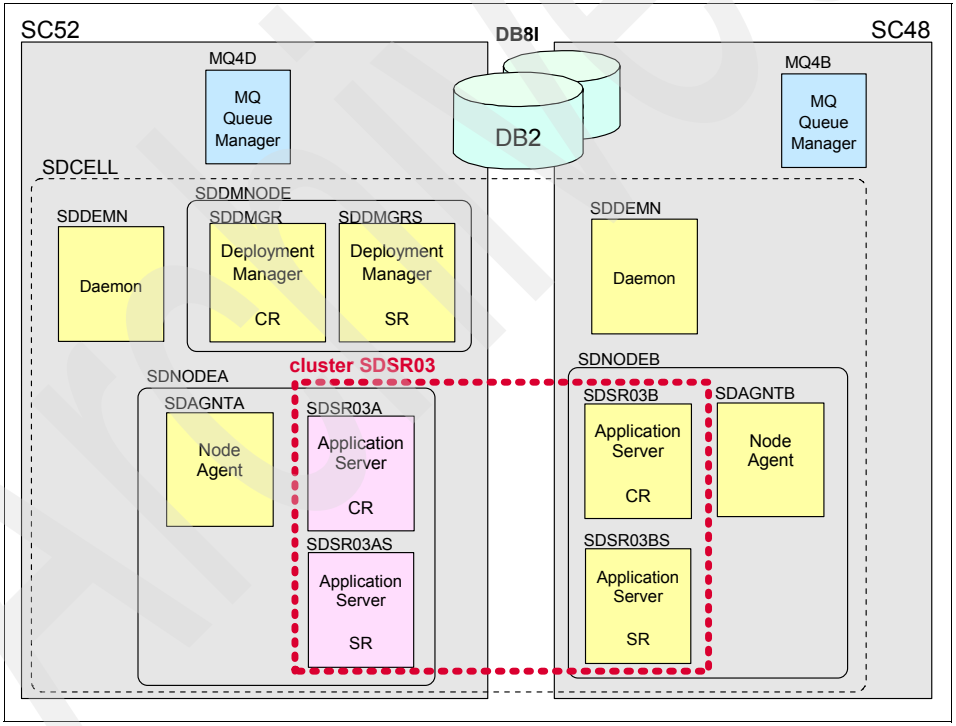



*Figure 13-3 Network deployment configuration on the sysplex*

From the Admin Console in the Deployment manager, you can see the cluster topology like the example in Figure 13-4.

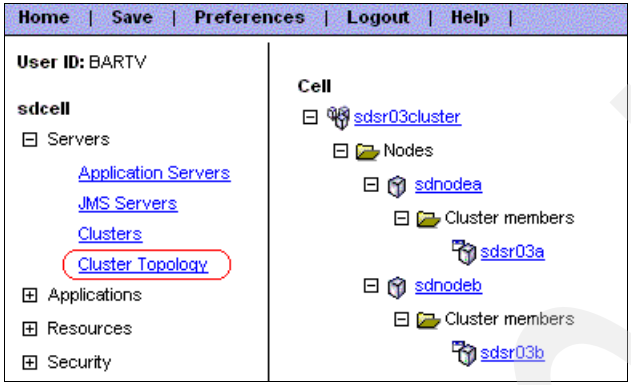



*Figure 13-4 Cluster topology from the Admin Console*

## 13.4.2 DB2 subsystems

The systems SC52 and SC48 are running DB2 V8.1 in Data Sharing mode. Therefore, the data is available from both systems. We use this DB2 database server for both the business data and the system tables to support Business Process Choreographer.

## 13.4.3 WebSphere MQ queue managers

On both SC52 and SC48, a WebSphere MQ V5.3.1 subsystem is installed. However, keep in mind that both queue managers are stand-alone systems. For the purpose of this book, no shared queues have been set up in the coupling facility.

> **Note:** In WebSphere Application Server for z/OS V5.1, shared queues cannot be used for the setup of Business Process Choreographer on this run-time setup.

### 13.4.4 CICS Transaction Server region

One of the activities in the business process is implemented in CICS. However, for the purpose of this document, we do no discuss high availability solutions for this back-end application. Traditionally, you should use CICSPlex® SM to implement such a solution. For our sample scenario, we use a remote CICS system on a third LPAR, SC66 running CICS Transaction Server V2.2.

## 13.5 Generating a Web service from the CICS business logic

In the following sections, we document the design and development process of a Web service from the CICS business logic for our specific business scenario (see Figure 13-1 on page 222). Its implementation requires two components:

- The existing business process workflow

  This workflow has been described as the Spot Loan scenario in Chapter 11, “Applied Decomposition pattern: Spot Loan scenario” on page 179. For the developer or designer, it is represented as a Business Process Execution Language (BPEL) file which can be manipulated graphically in WebSphere Studio Application Developer Integration Edition (WebSphere Studio). Figure 13-5 shows the makePayment.wsdl file that represents the Spot Loan process.

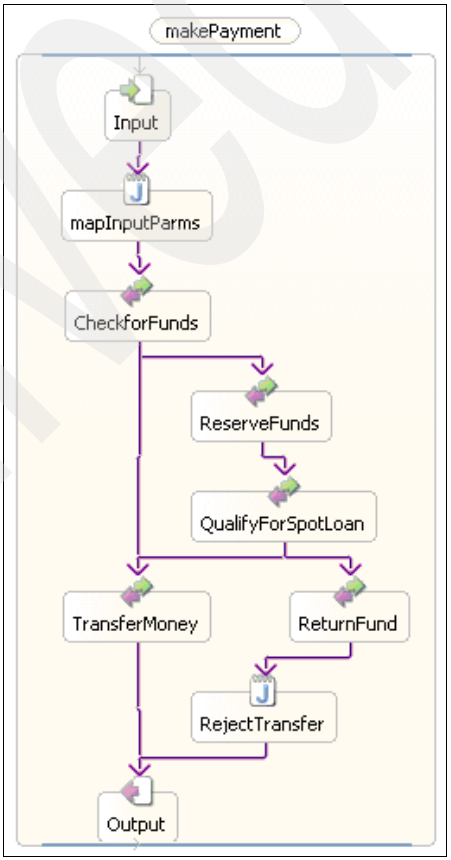



*Figure 13-5 The Spot Loan business process flow*

- The CICS business logic that implements the Funds Optimizer Service

  This logic has to be represented by a Web Services Definition Language (WSDL) interface for Web services. We created a WSDL description of the CICS application using WebSphere Studio.

### 13.5.1 Importing the commarea copybook into WebSphere Studio

As part of the additional material of this redbook, we supply the commarea COBOL copybook of the CICS program, which is the interface for that business logic. Figure 13-6 shows the commarea layout.

```
01 ACCOUNTINFO.
         03 RETURNCODE                PIC X(02).
         03 USERID                    PIC X(60).
         03 PASSWORD                  PIC X(10).
         03 CORRELID                  PIC X(32).
         03 ACCOUNTNO                 PIC X(08).
         03 FILLER                    PIC X(17).
```

*Figure 13-6 COBOL copybook for CICS business logic*

To generate a WSDL file that represent the CICS business logic, we import the commarea copybook into WebSphere Studio by using the following steps.

1. Go to the Business Integration perspective.
2. Select **File** → **New** and choose the **Create new service project** option.
3. Set Project Name to `FundOptimizer`, and click **Finish**.
4. Select the **FundOptimizer** project.
5. Select **File** → **New** to create a new package.
6. For Name, type `com.ibm.oneida.cics`, and click **Finish**.
7. Import the commarea.ccp file into this project.
   a. Click **File** → **Import**.
   b. For Import location, select **File system** and click **Next**.
   c. Locate the **commarea.ccp** file that you obtained from the additional material. Make sure that Into folder field is set to `FundOptimizer/com/ibm/oneida/cics`.
   d. Click **Finish** to complete the import operation.

## 13.5.2 Generating a new WSDL file

You are now ready to generate the WSDL file.

1. Highlight the **FundOptimizer** project, and select **File** → **New**, and choose **Service built from ...**
2. The New Service wizard opens. In the Create Service panel (Figure 13-7):
   - If you see CICS ECI in the window, as in this example, skip to step 3.
   - If you do not see CICS ECI, complete the following steps:
     i. Click the **Import resource adapter...** button.
     ii. In the Connector import window, browse for the adapter file in the Connector file field. The default path is C:\Program Files\IBM\WebSphere Studio\Application Developer IE\v5.1\resource adapters\ctg510\cicseci.rar.
     iii. Click **Finish** to return to the New Service wizard.

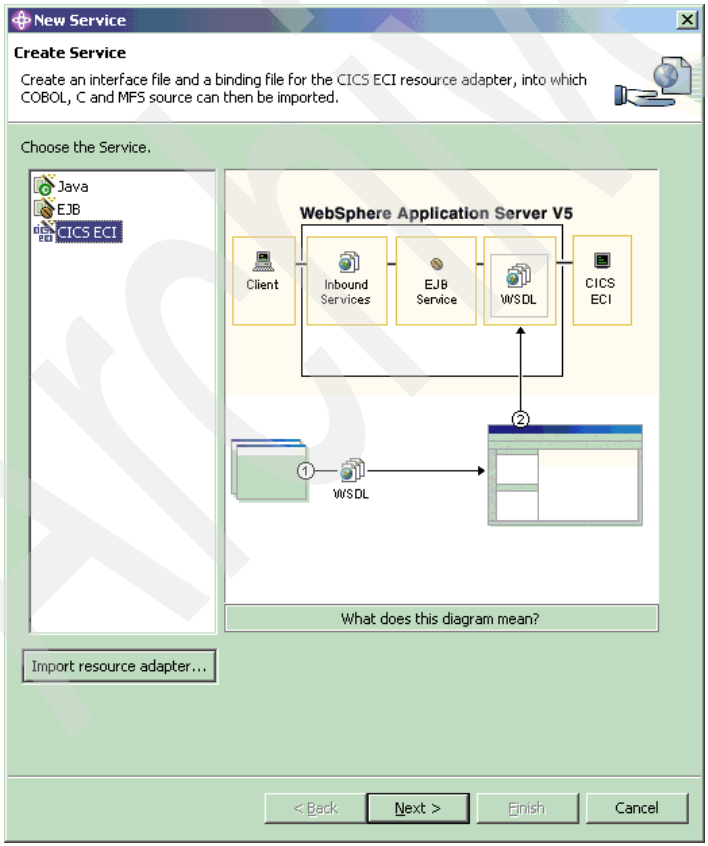



*Figure 13-7 Create New Service wizard*

3. In the New Service panel (Figure 13-7), select **CICS ECI service**. Click **Next**.
4. The Connection Properties panel (Figure 13-8) opens, in which you should complete the location of the CICS Transaction Gateway (CTG) daemon and target CICS region.
   a. For Gateway address, specify the host name or IP address URL for the CTG daemon.
   b. For Port number, specify the CTG listener port.
   c. For CICS server name, type the VTAM® application ID of the target CICS region.
   d. Click **Next**.

   Figure 13-8 shows the details for our environment. This information enables you to test the CICS program from WebSphere Studio.

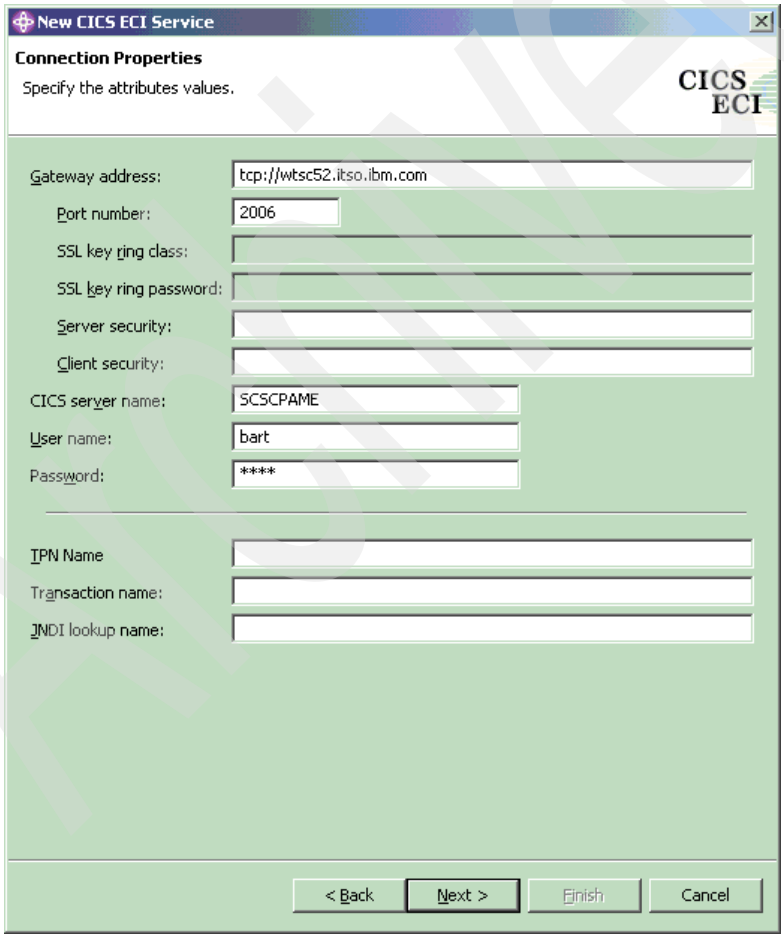



*Figure 13-8 CICS ECI service connection properties*

5. In the Service Binding panel (Figure 13-9), for Interface file name, type `FundOptimizer` and keep the default values for all other fields. Click **Finish**.

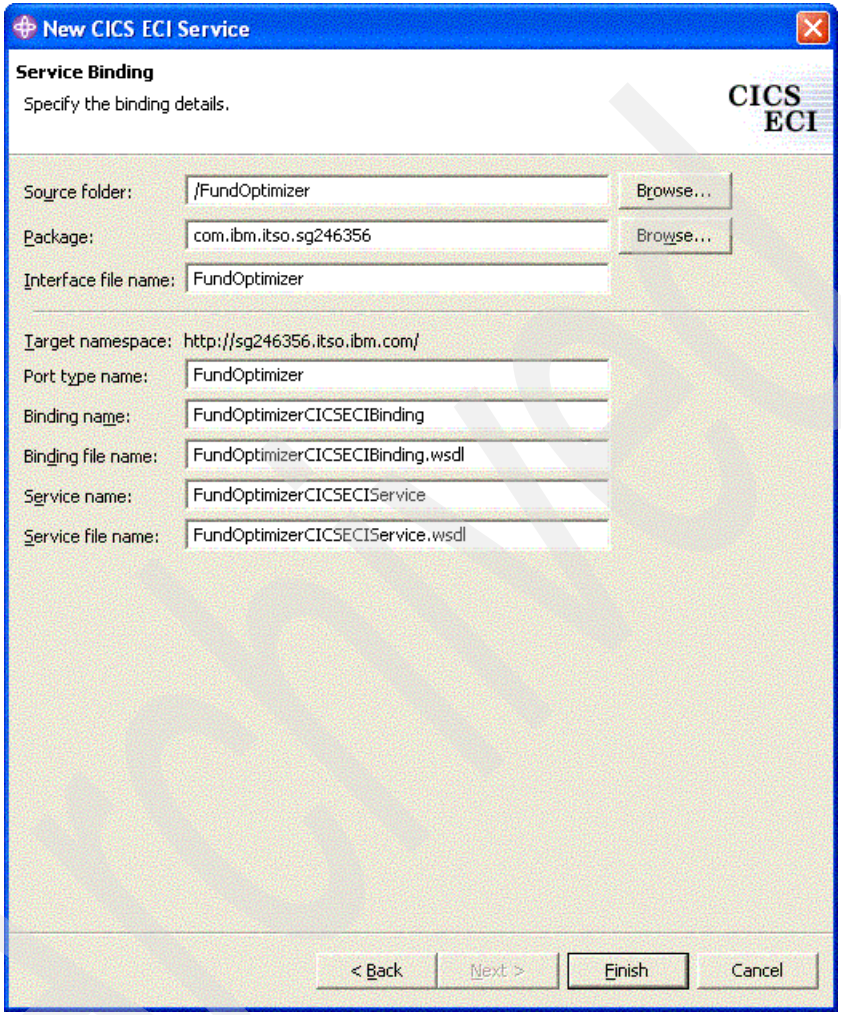



*Figure 13-9 Service bindings for the CICS ECI service*

## 13.5.3 Binding WSDL files to the CICS business logic

At this point, we have created an empty WSDL file for a CICS Web service. Rather than one WSDL file, there are three files. One of these files, the Binding WSDL file, will be opened for edit when you complete step 5.

Now complete the following steps to bind the freshly created WSDL files to the CICS business logic.

1. From the Binding WSDL editor window (Figure 13-10), select **FundOptimizerCICSECIBinding**, right-click, and select **Generate Binding Content...**.

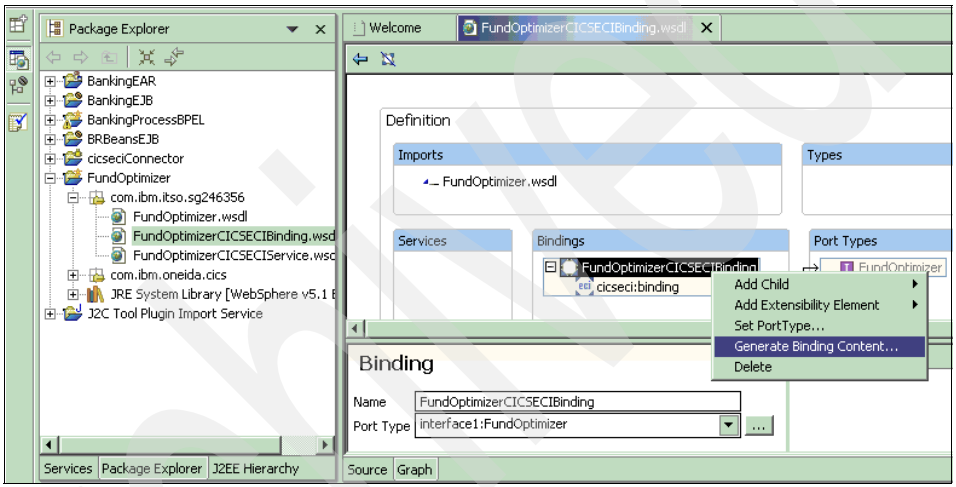



*Figure 13-10 Generate binding content for the CICS Web service*

2. In the Specify Binding Details window, click the **Add** button to create a new binding operation.

3. The Operation Binding window opens. The operation represents the ECI call into CICS, and is of the type REQUEST_RESPONSE. We name the operation `optimumFunding`. Click **Next** to continue.

4. In the Operation Binding Properties window (Figure 13-11), supply the functionName value. The value must be the name of the commarea-based CICS program that the service will implement. In our case, it is named `BPECOB`. Click **Next** to continue.

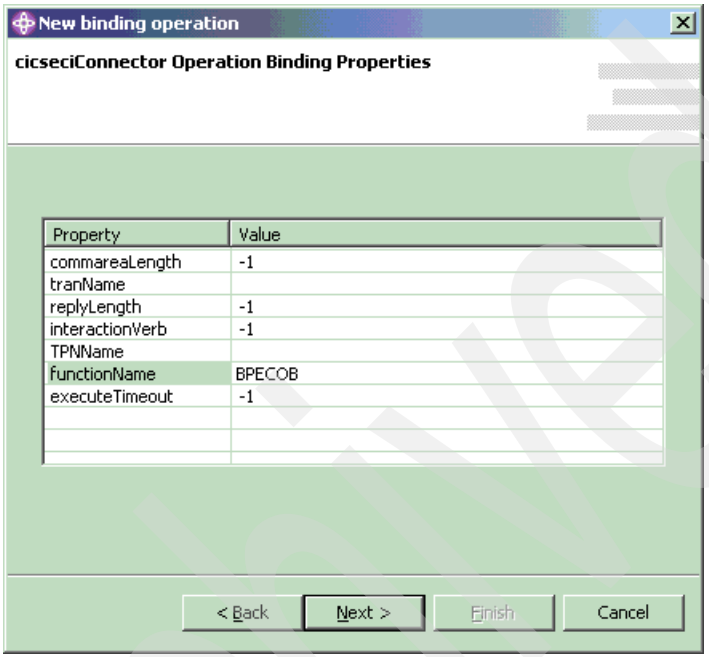



*Figure 13-11 CICS connector operation binding properties*

5. At this stage, specify the operation binding by means of the message format that the service will exchange between the WebSphere application and the CICS program. In the Operation Binding window, click the **Import** button to import the commarea copybook that we saved earlier.

6. Navigate through the class path in the FundOptimizer project as shown in Figure 13-12. In the File Selection Page panel, select the **commarea.ccp** file and click **Next**.

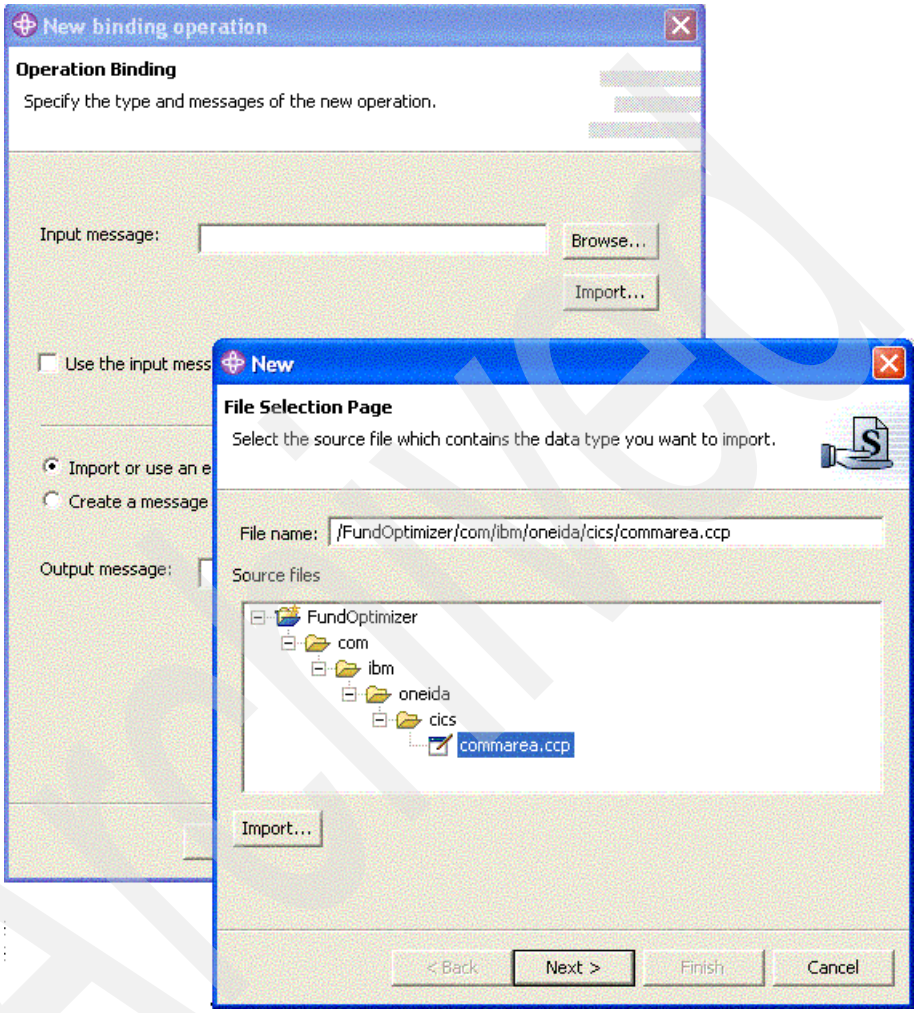



*Figure 13-12 Importing the CICS program copybook into WSDL operation binding*

7. In the COBOL Import Properties panel (Figure 13-13), for Platform, specify **z/OS** as the target platform. Also for Code Page Selection, choose the correct character code page of the CICS region. Figure 13-13 reflects the values for our system. Click **Next**.

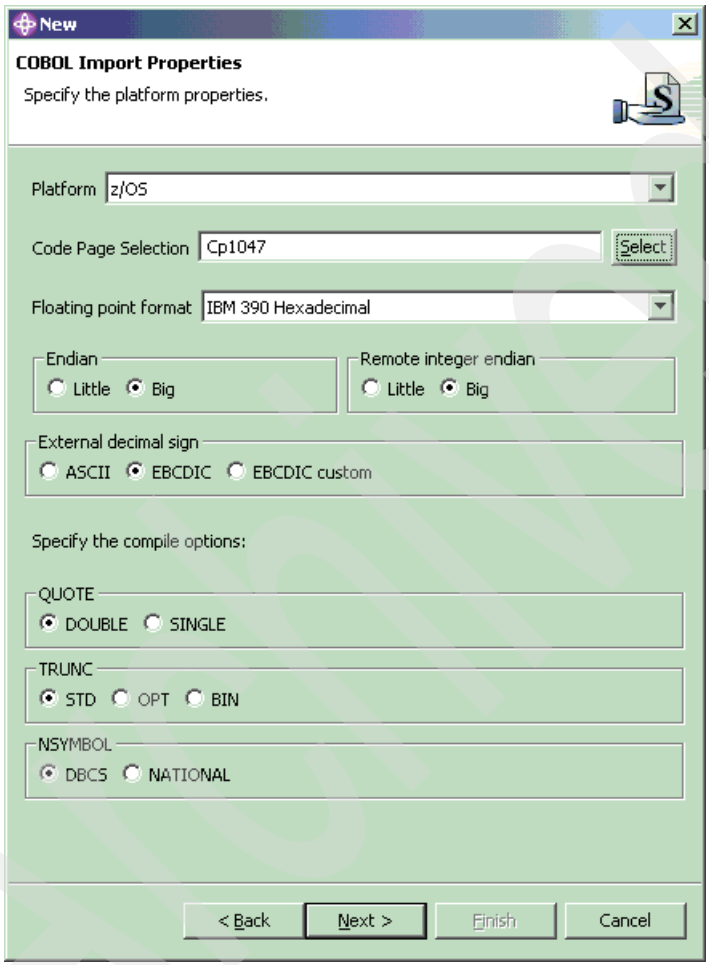



*Figure 13-13 COBOL platform characteristics*

8. The COBOL Importer window shows the top-level declarations from the imported copybook. We select **ACCOUNTINFO** and click **Finish**.
9. You return to the Operation Binding window. Select the **Use the input message for output** check box, and click **Finish**.
10. In the Specify Binding Details window, click **Finish** one more time.

This completes the creation of the CICS service interface in WSDL.

## 13.5.4 Generating deploy code

At this stage, the WSDL files describe the CICS ECI service. We need to call this service as executable code. We generate this code as explained in the following steps.

1. In the Business Integration perspective, select the **FundOptimizerCICSECIService.wsdl** file, right-click, and select **Enterprise Services** → **Generate Deploy Code...** as shown in Figure 13-14.

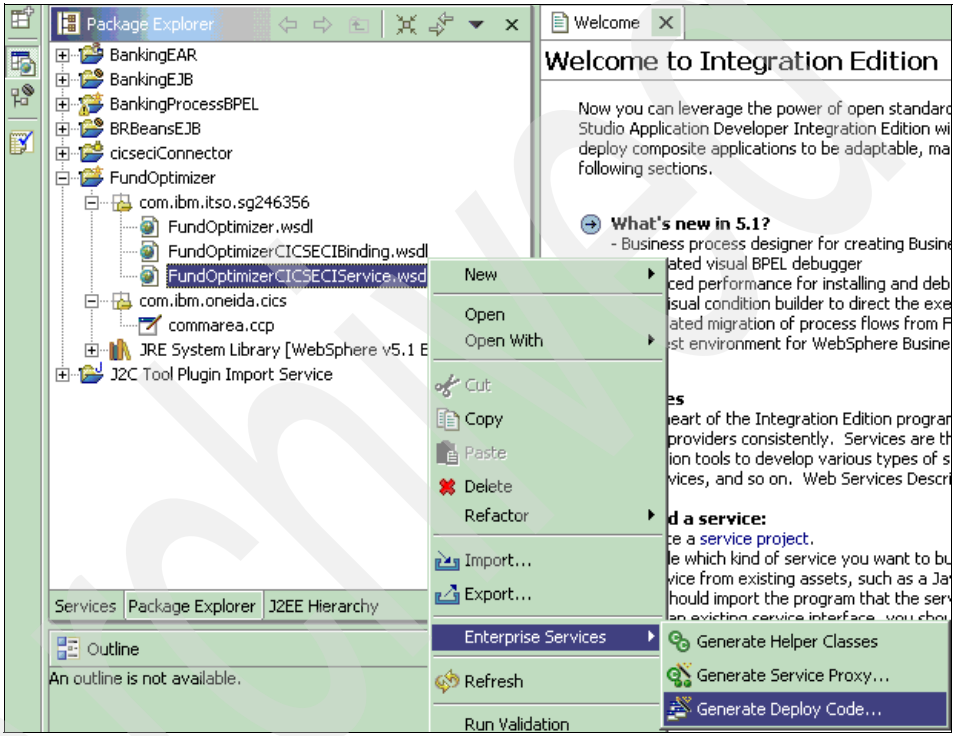



*Figure 13-14 Generating deploy code for CICS ECI service*

2. Set Inbound binding type to **EJB**. We choose EJB because we want to execute the CICS ECI service as part of a WebSphere transaction.
3. Accept the defaults for all other fields and click **Finish**.

As a result of these activities, you now have two additional projects in WSAD IE:

- FundOptimizerEJB, which has the EJB deployment descriptor and code
- FundOptimizerEAR, which has the application descriptor, and which you should export for deployment into the application server

**Note:** When you deploy EJBs, choose your transaction settings carefully. As shown in Figure 13-15, we set the transaction type for the optimumFunding operation in the FundOptimizerService session EJB to Supports. Doing this, prevents a new transaction from being created each time the method is called.

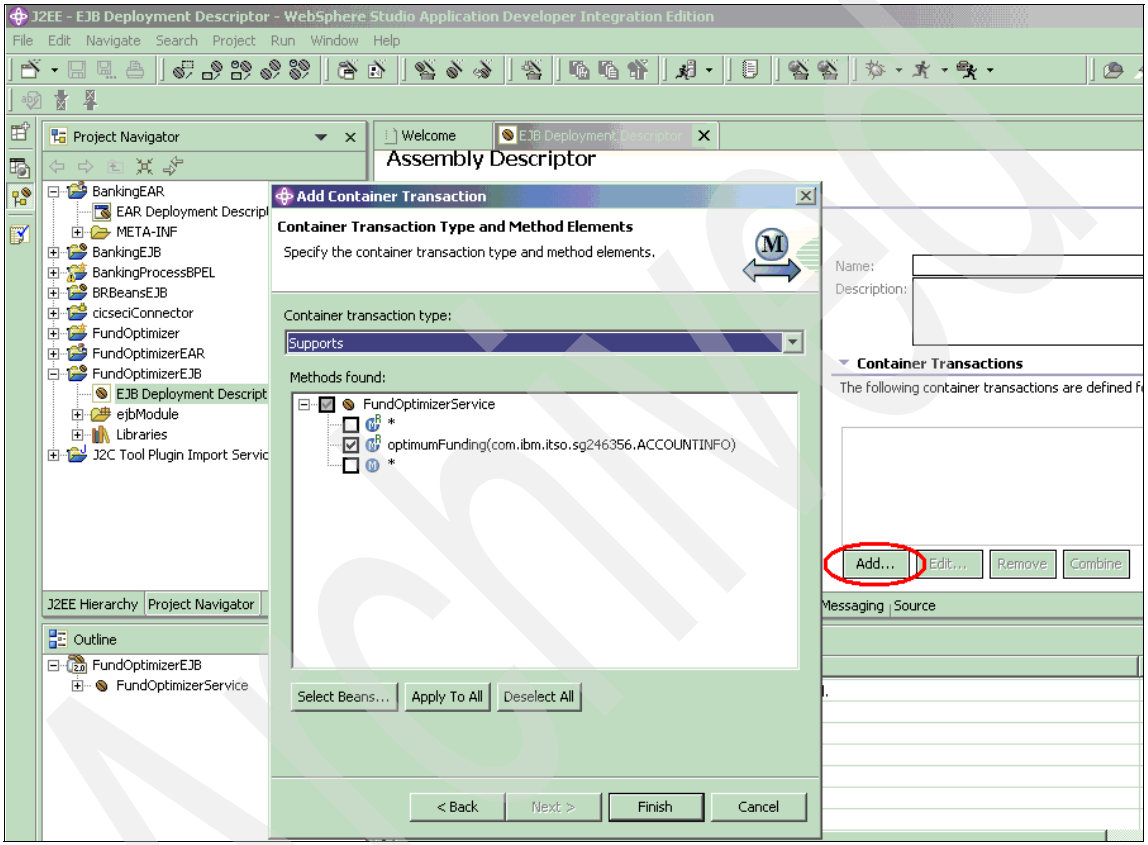



*Figure 13-15 Transaction type for the CICS service operation*

We now need to integrate the CICS EJB service into the existing makePayment.bpel business process.

## 13.6 Integrating the CICS Web service in the business process

This section takes you step by step through the process to integrate the new CICS component, the CICS Web service, into the business process, and the makePayment.bpel file. The starting point is a WebSphere Studio workspace.

We supply our WebSphere Studio workspaces in a zip file with the additional material of this redbook. See Appendix D, “Additional material” on page 323, for instructions to set up this workspace on your workstation. Take the workspace supplied in the wksp_zOS_before sub-directory of the sysplex_version directory. Unzip the workspace wksp_zOS_before into a directory path of your workstation. Use the unzipped location as the workspace for your WSAD IE session. If you are interested only in the result, then use the workspace from wksp_zOS_after sub-directory, or import from the deployable EAR files into WebSphere Studio.

In comparison to the Spot Loan business flow as shown in Figure 13-5 on page 229, the target business flow for this section is represented in Figure 13-16. This is the business process for the Optimized Spot Loan use case defined in 13.2.2, “Use case 004: Optimized Spot Loan” on page 223.

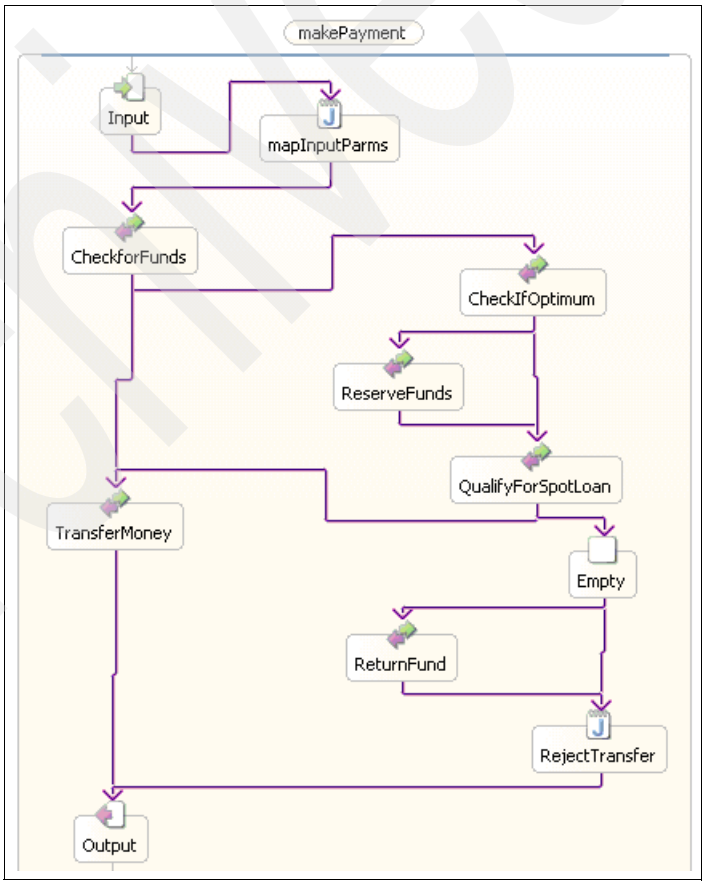



*Figure 13-16 Optimized Spot Loan business process flow*

To integrate the CICS Web service into the business process, follow these steps:

1. Go to the Services view in the Business Integration perspective of your workspace. Open the **Service Projects** folder. Expand the **BankingProcessBPEL** project and **com.ibm.oneida.bs.flow.makePayment**. Double-click the **makePayment.bpel** file to bring up the editor.
2. Complete the following tasks:
   a. Click the plus sign (**+**) in the Partner Links area of the drawing canvas, and add the CICS service as a new PartnerLink. Name it `FundOptimizer`.
   b. Select the **Implementation** tab, along the lower left side of the canvas.
   c. Click the **New...** button to create a new partner link type.

   Refer to Figure 13-17.

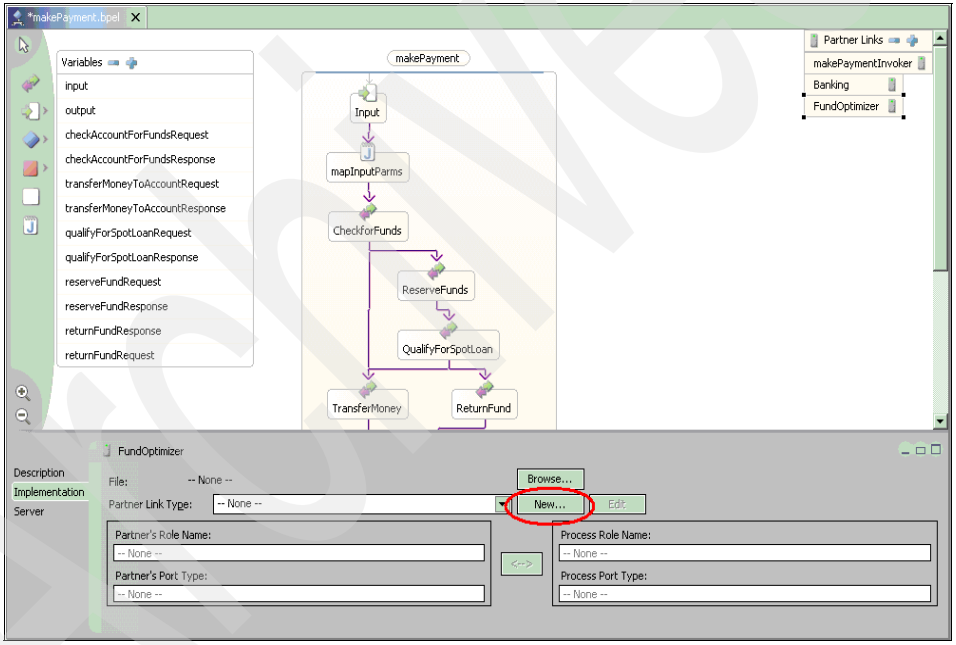



*Figure 13-17 Adding the CICS ECI service as a partner link in BPEL*

3. In the New Partner Link Type window, complete the following steps:
   a. In the First role field, enter a value, for example `Service`.
   b. Click the **Browse...** button to select a role from a port type file.
   c. Select the **FundOptimizer.wsdl** file from the FundOptimizer service project and click **OK**.
   d. Click **OK** again to close save the new Partner Link Type.

   Verify your results with those shown in Figure 13-18.

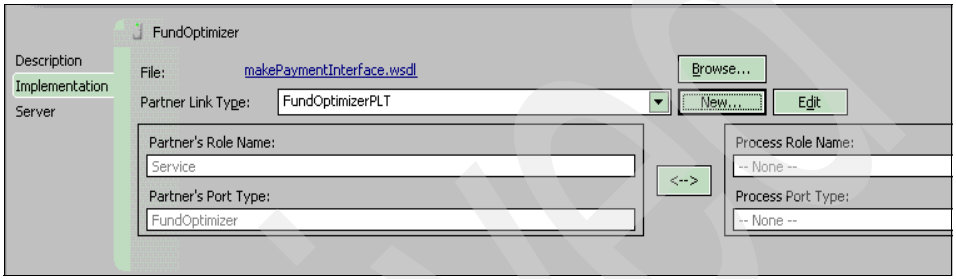



*Figure 13-18 Partner Link Type implementation*

4. Add two new variables.
   a. Under Variables, click the plus icon (+).
   b. Name the variables `optimumFundRequest` and `optimumFundResponse` respectively.
   c. Associate the variables with the correct message implementation from the FundOptimizer service, as shown in Figure 13-19.
   d. Since we used the same message for input and output of the CICS service operation, you can only select **optimumFundingRequest** as the value for the Message field.

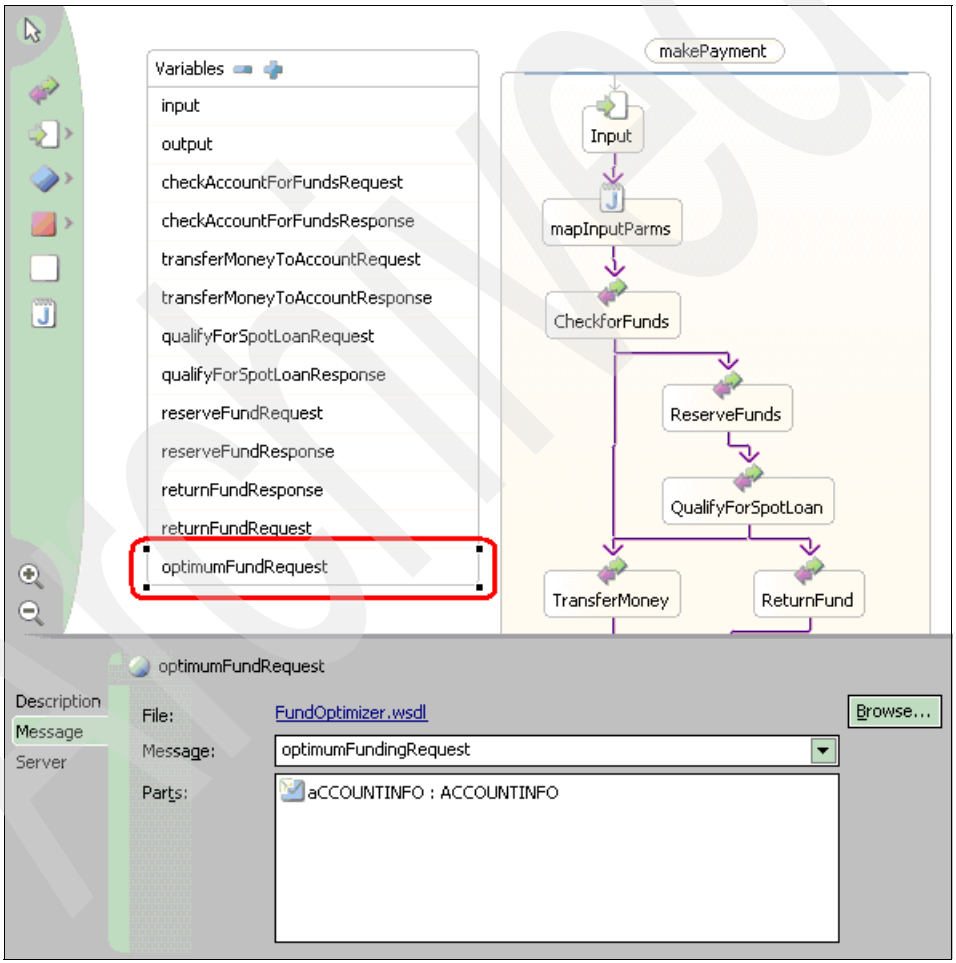



*Figure 13-19 Associating the BPEL variable with the WSDL message*

5. Drop a new Invoke activity onto the canvas. Name the activity `CheckIfOptimum`.
   a. Click the **Implementation** tab.
   b. For Partner Link, select **FundOptimizer**.
   c. For Request and Response, select the newly created variables (see Figure 13-20).
   d. Save your changes.

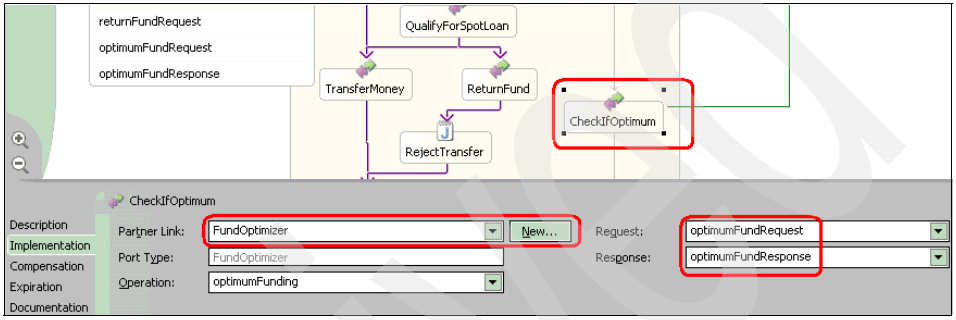



*Figure 13-20 Creating a new Invoke activity*

6. Remove some of the existing links from the canvas and create new links to integrate the CheckIfOptimum activity in the flow. Use care in defining the condition that is associated with the link. An example in shown in Figure 13-21, where the Condition value is set to Otherwise.

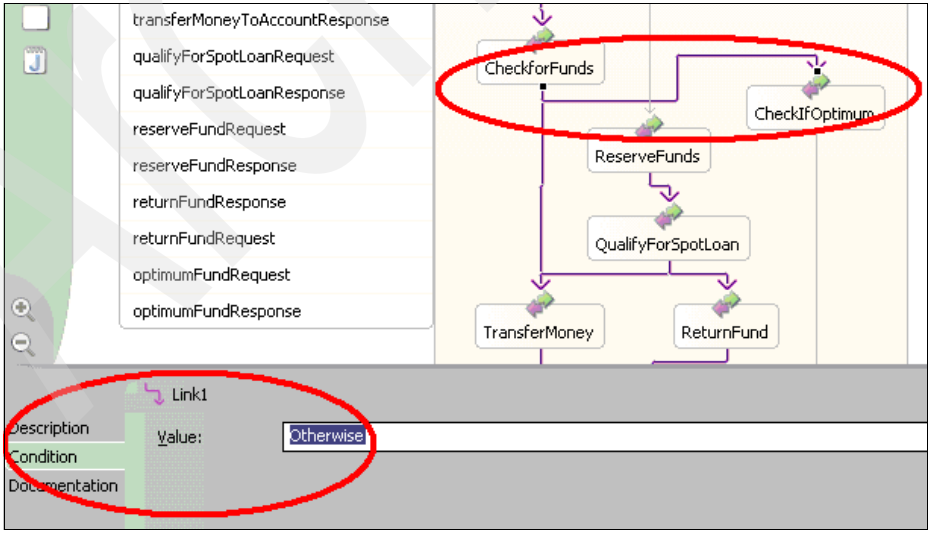



*Figure 13-21 Specifying link conditions in BPEL*

Figure 13-22 shows the result of removing some of the existing links. See Table 13-4 for the conditions that are associated with the numbered links in Figure 13-22. Notice that we have coded an Empty activity for convenience.

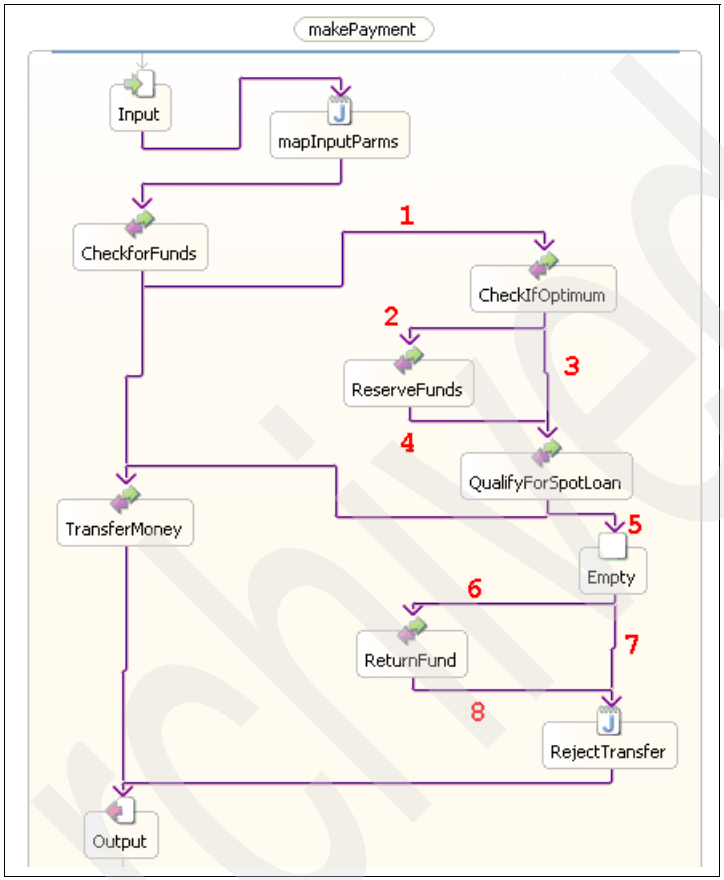



*Figure 13-22 Integration of the CICS service in the business flow*

*Table 13-4 Setting the conditions for the links between BPEL activities*

| Link | Condition |
|---|---|
| 1 | **Otherwise** |
| 2 | **Expression**<br>`OptimumFundingRequestMessage cicsOut = getOptimumFundResponse();`<br>`ACCOUNTINFO commareaOut = cicsOut.getACCOUNTINFO();`<br>`String rc = commareaOut.getReturncode();`<br>`if (rc.equals("00"))`<br>`  return false;`<br>`else return true;` |
| 3 | **Otherwise** |
| 4 | **True** |
| 5 | **Otherwise** |
| 6 | **Expression**<br>`OptimumFundingRequestMessage cicsOut = getOptimumFundResponse();`<br>`ACCOUNTINFO commareaOut = cicsOut.getACCOUNTINFO();`<br>`String rc = commareaOut.getReturncode();`<br>`if (rc.equals("00"))`<br>`  return false;`<br>`else return true;` |
| 7 | **Otherwise** |
| 8 | **True** |

Also notice that you will get some errors while entering the Java snippets for the conditions of type *Expression*. You can correct this by clicking the **Import** tab, at the top of the canvas of the makePayment process, and inserting the following two lines:

```
import com.ibm.itso.sg246356.*;
import com.ibm.itso.sg246356_msg.*;
```

7. Assign a value to the CICS commarea. We do this by assigning a value to the OptimumFundingRequestMessage which is input to the CICS activity. We include the lines that are highlighted in Figure 13-23 to the Java snippet of the MapInputParms activity.

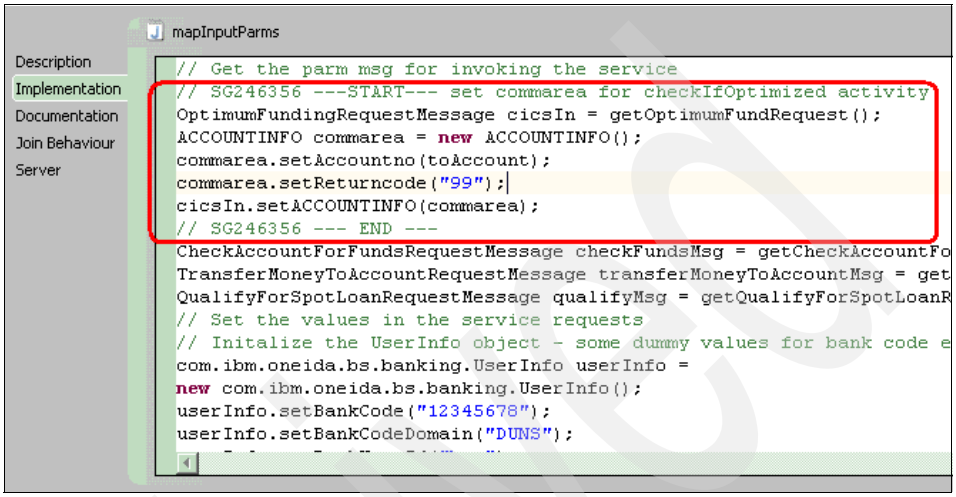



*Figure 13-23 Assigning the value for the CICS commarea in BPEL*

8. Close the editor window to save the makePayment.bpel file.

The business process in the BankingProcessBPEL project has now been updated and the deployment code can be generated. To generate the code:

1. Right-click the **makePayment.bpel** file and select **Enterprise Services** → **Generate Deploy Code**.
2. In the Generate BPEL Deploy Code window (Figure 13-24), specify the port for the FundOptimizer partner.

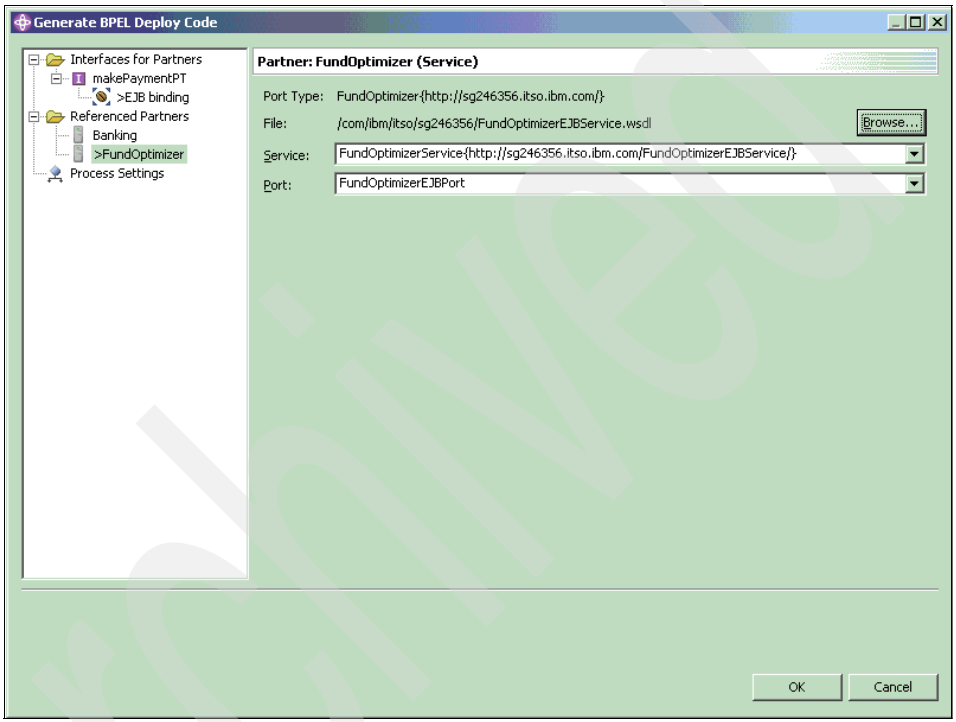



*Figure 13-24 Specifying ports for referenced BPEL partners*

The code is generated into three new projects: BankingProcessBPELEJB, BankingProcessBPELWeb, and BankingProcessBPELEAR. The latter should be exported as an EAR file and deployed into WBISF on z/OS.

## 13.7 Configuring the infrastructure for the sample application

In the following sections, we explain the key steps to prepare the infrastructure into which our sample application will be deployed.

### 13.7.1 Installing the Business Process Container

Refer to 9.2, “Configuring the BP Container” on page 142, which explains how to install the Business Process Container.

### 13.7.2 Setting up the resource adapter for CICS

Our J2EE application will connect to CICS, which requires us to set up an adapter. For the installation instructions of the CICSECI resource adapter in a WebSphere Application Server, refer to the redbook *WebSphere for z/OS V5 Connectivity Handbook,* SG24-7064.

When you install the CICSECI resource adapter, be aware that the resource adapter is installed in a single node (see Figure 13-25), even if your Admin Console scope is currently set to the cell level. This means that, in your sysplex environment, you have to install the RAR file twice.

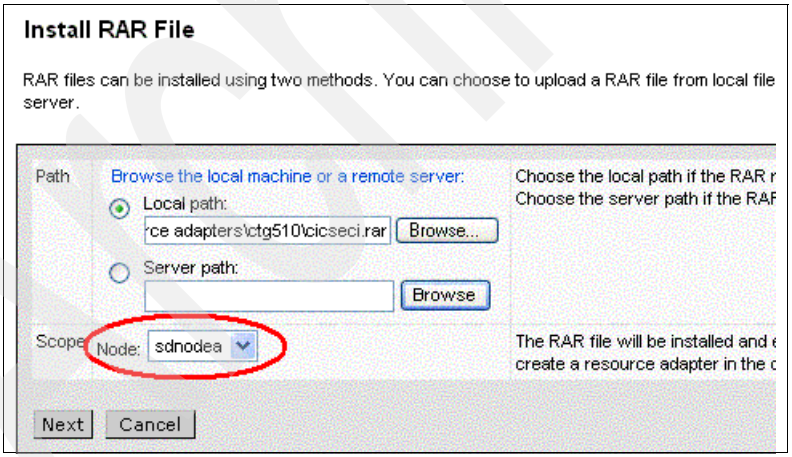



*Figure 13-25 CICS resource adapter installation*

You must also set the native classpath. The CICS Transaction Gateway uses Java Native Interface (JNI) calls and requires this path setting (see Figure 13-26) to find the libCTGJNI.so module. This is only required if you use the local protocol to connect from the application directly to CICS, rather than through a CICS Transaction Gateway daemon.

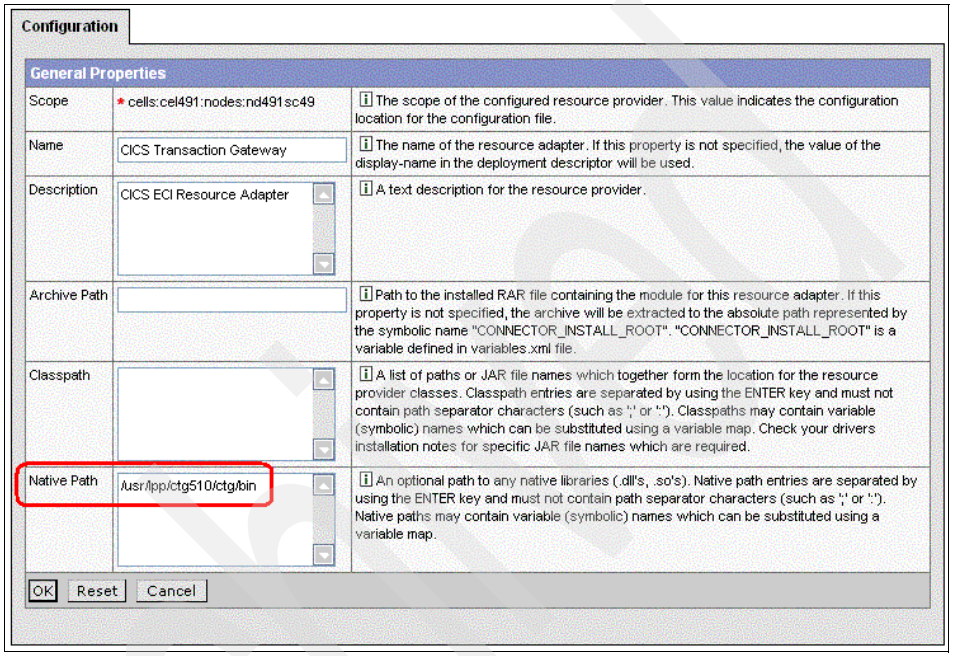



*Figure 13-26 CICS resource adapter native path*

**Note:** Since at least build level c502-20040302, CICS Transaction Gateway for z/OS now includes the cicsecitools.jar utilities classes in the CICSECI resource adapter archive. The cicseci.rar file is usually located in /usr/lpp/ctg510/ctg/deployable. Therefore, it is no longer required to install the cicseci.rar file that is shipped with WebSphere Studio Application Developer. The utility classes are required to support WSIF format handlers for the CICS Web service.

Once the CICSECI resource adapter has installed correctly, you can create a CICS Connection® Factory.

1. From the navigator pane on the Admin Console, select **Resources** → **Resource Adapters** and select the **CICS Transaction Gateway** from the list.
2. Go to Additional Properties at the bottom of the resource adapter pane, and select **J2C Connection Factories**.
3. As shown in Figure 13-27, create a new Connection Factory and name it the same as the CICS region to which you will connect. Click **OK** to save the new Connection Factory.

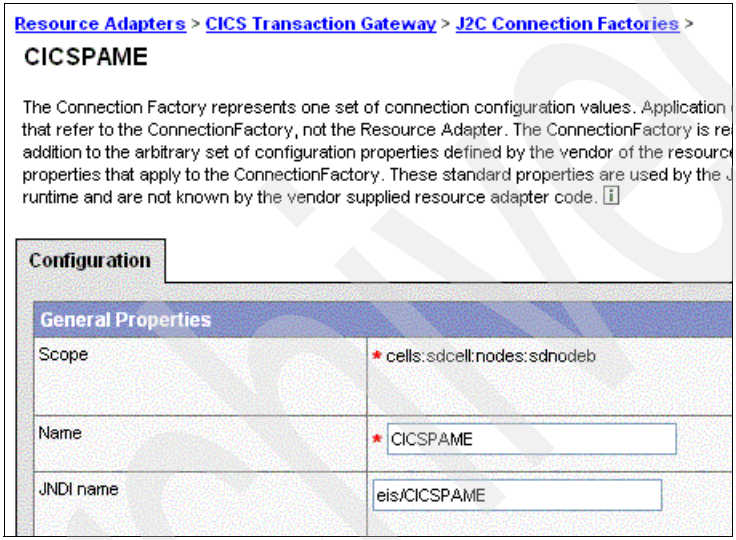



*Figure 13-27 CICS Connection Factory*

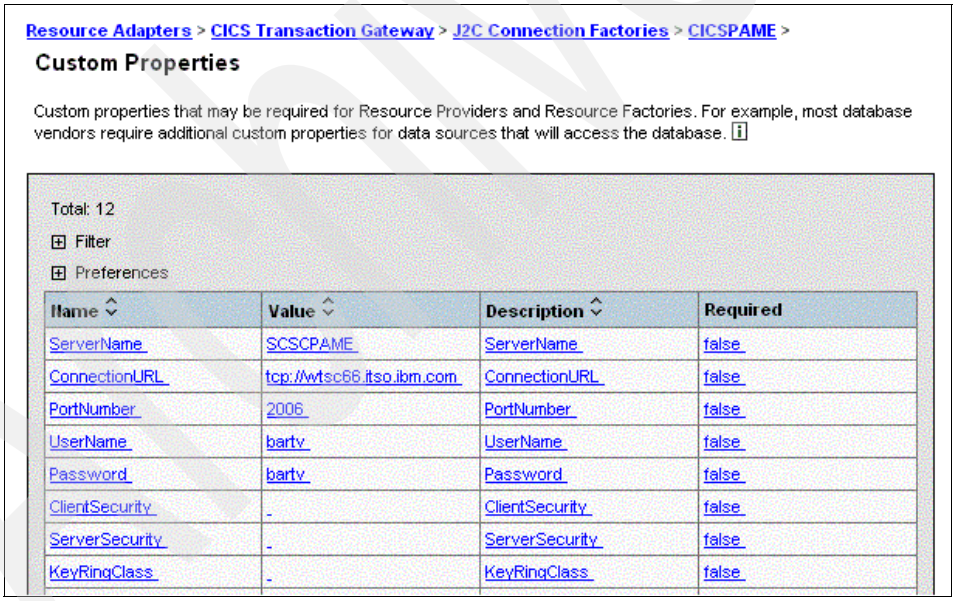
4. After you create the new Connection Factory, you can change its properties. Select the new connection factory from the list.
5. Then you see the Custom Properties panel, like in Figure 13-28, which shows the settings that we used for our tests.

   Make sure to use the correct protocol in the ConnectionURL setting. It is used to establish the network connection from the WebSphere container to the CTG daemon, which itself is set up to connect to the target CICS region, based on the CICS VTAM application ID specified as ServerName in the connection factory.

In Figure 13-28, the protocol is tcp:. The supported values are:

- **tcp**: This is a basic TCP/IP socket protocol. The ECI call from the application is packaged and exchanged with the CTG using TCP/IP socket APIs.
- **http**: The ECI call is packaged and exchanged with the CTG using the HTTP protocol.
- **ssl**: The ECI call is packaged and exchanged with the CTG using the SSL encryption on top of the basic TCP/IP socket protocol.
- **local**: The ECI call is packaged and exchanged with CICS using CICS's proprietary EXCI protocol. In this case, no intermediate CTG daemon is required. However, it can only be used if the application and CICS are on the same MVS image. EXCI is a cross-memory rather than a network protocol, and it is RRS-enabled. It is the preferred choice when possible.

The protocol that you choose in the connection factory must match the CTG daemon configuration. A number of listeners for each protocol must be defined in the CTG. This number can be 0.




| Name | Value | Description | Required |
|---|---|---|---|
| ServerName | SCSCPAME | ServerName | false |
| ConnectionURL | tcp://wtsc66.itso.ibm.com | ConnectionURL | false |
| PortNumber | 2006 | PortNumber | false |
| UserName | barty | UserName | false |
| Password | barty | Password | false |
| ClientSecurity | - | ClientSecurity | false |
| ServerSecurity | - | ServerSecurity | false |
| KeyRingClass | - | KeyRingClass | false |

*Figure 13-28 CICS Connection Factory properties*

## 13.7.3 Configuring the CICS Transaction Gateway

For the purpose of our tests, we have not installed a CICS region on each of the sysplex LPARs. Rather than the configuration in Figure 13-2 on page 226, we have worked with the sysplex configuration as shown in Figure 13-29.

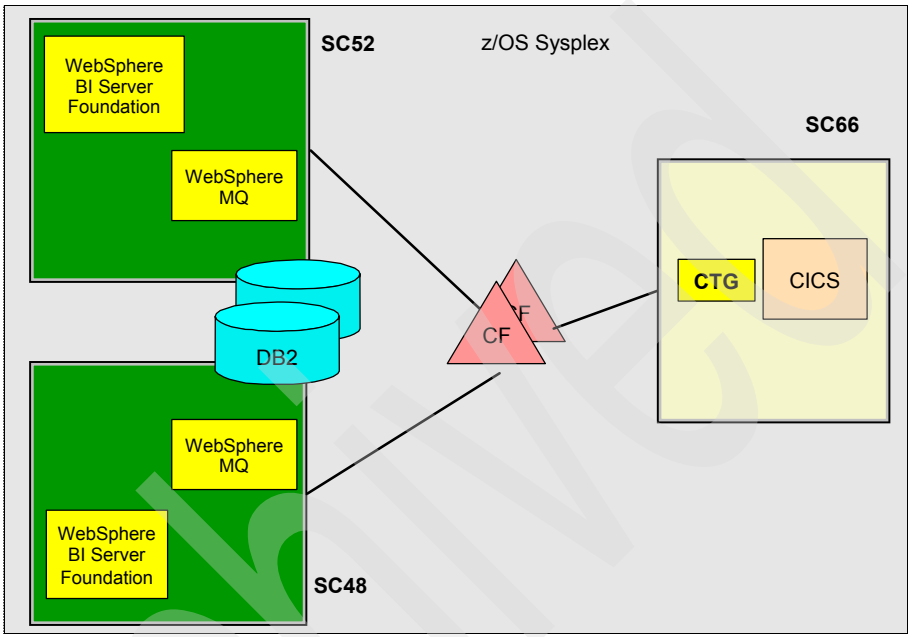



*Figure 13-29 Accessing a remote CICS server from a WebSphere cluster*

This effectively separates the WebSphere Application Server tier from the CICS back-end tier. Note that the transactions going to a remote CICS systems from a WebSphere application cannot coordinate through RRS.

In this scenario, a CICS call from a WebSphere application follows this flow:

1. A J2EE application uses the JCA API to open a connection to CICS from a CICS Connection Factory.
2. This CICS Connection Factory is configured to use the tcp: protocol to the CTG daemon, as shown in Figure 13-28.
3. The CTG for z/OS daemon must run on the same MVS image as the target CICS system and use the EXCI protocol to connect to CICS.

For further installation and configuration guidelines for the CICS Transaction Gateway, see *CICS Transaction Gateway V5 - The WebSphere Connector for CICS,* SG24-6133.

14

# Best practices for high availability on z/OS

This chapter provides some guidelines and best practices for high availability on z/OS. We summarize how specific z/OS features map onto the high availability run-time requirements of this pattern.

We provide details about:

- TCP/IP traffic on z/OS
- z/OS Workload Manager (WLM) setup for WebSphere Business Integration Server Foundation (WBISF)
- DB2 Data Sharing
- Resource Recovery Services (RRS)
- Session affinity for Web applications regarding security and performance

## 14.1 TCP/IP traffic on z/OS

We chose the following z/OS features for implementing the Runtime pattern:

- Dynamic Virtual IP Addressing (DVIPA)
- Sysplex Distributor
- Multi-Node Load Balancer
- HiperSockets™

For details about the implementation of WebSphere for z/OS on a system with multiple TCP/IP stacks, refer to the manual *WebSphere Business Integration Server Foundation for z/OS V5.1: Servers and Environment*, GA22-7977.

### 14.1.1 Dynamic Virtual IP addressing (DVIPA)

Virtual IP Addressing (VIPA) is a z/OS solution to detach a host IP address allocation from its physical network devices. Without VIPA, connections are bound to an IP address of a specific zSeries network interface. Failure of this interface would disrupt service.

A VIPA address, however, is mapped to a virtual device that never fails. Incoming connections to this address are moved by TCP/IP internally up the IP stack for the next available interface on the system. This way, redundancy in the network interfaces of the system can be implemented transparently to remote applications.

While VIPA is a solution on a single z/OS image, DVIPA extends it to multiple z/OS images, providing:

- Automatic takeover on a VIPA backup IP stack, typically on another logical partition (LPAR) of the sysplex
- Dynamic activation of VIPA

### 14.1.2 Sysplex Distributor

DVIPA alone does not provide workload distribution of network traffic. For this purpose, we recommend the *Sysplex Distributor* solution. Rather than the mere automatic takeover provided by DVIPA, Sysplex Distributor adds coordination with z/OS Workload Manager to distribute work to one of several available DVIPA addresses in the sysplex. It also provides the management of server affinity once a connection from the client is established.

While Sysplex Distributor adds workload balancing capabilities, it also introduces a problem with applications that have network connection affinity to a particular application server. Two application components are prone to this problem:

- HTTP sessions in Web applications
- Stateful session EJBs

See 14.5, “Session affinity for Web applications” on page 265, and 14.5, “Session affinity for Web applications” on page 265, which briefly present the problem and provide solutions.

### 14.1.3 Multi-Node Load Balancer

This solution is based on CISCO technology. Where the Sysplex Distributor resides on the sysplex, the Multi-Node Load Balancer is embedded in the CISCO router. It balances the workload dynamically based on WLM information that it receives from an agent on the sysplex. Refer to the redbook *Architecting High Availability e-business on IBM @server* zSeries, SG24-6850, for an introduction to this technology.

### 14.1.4 HiperSockets

HiperSockets are a unique zSeries hardware feature. Implemented in microcode, HiperSockets enables TCP/IP traffic between LPARs to be routed through the system memory of the processor rather than through the network adapters and the local network. This allows TCP/IP traffic at much faster speeds. In a sysplex environment, HiperSockets also improve TCP/IP traffic between the z/OS images, especially on the same server, and between different servers if the DYNAMICXCF option is coded in the TCP/IP profile. For further details, refer to the *zSeries HiperSockets*, SG24-6816.

This is the reason why we have chosen to implement the Web server nodes on a Linux for zSeries front end to the application server nodes. Although the operating system is different, we are still using the zSeries hardware throughout, giving us faster network connectivity.

## 14.2 z/OS WLM setup for WBISF

The Workload Manager for z/OS is a base component in the system. It is still possible to set it up in compatibility mode, which is the way performance management was done in the older OS/390 versions. For WebSphere, however, WLM must be set up in goal mode. Refer to the product manual *z/OS MVS Planning: Workload Management*, SA22-7602, for details.

All the workload management in the z/OS environment is handled by WLM, especially in a sysplex. We describe its major interactions with WebSphere in the following topics.

### 14.2.1 WebSphere and WLM on z/OS

WBISF for z/OS has a slightly different implementation from other platforms to specifically leverage the z/OS WLM capabilities. On other platforms, an application server is implemented as a single process. On z/OS, however, an application server is implemented as at least three processes, or *started tasks*, on the same MVS system:

- The *controller region* is a workload queue manager. It takes the incoming workload and places it on a work queue for a specific servant region, consulting WLM for its decisions. It may also start additional servant regions if workload cannot be handled by the current regions.
- The *servant region* takes the request from the WLM queue that is being fed by the controller region. The servant region runs the Web and J2EE containers. Several cloned servant regions can run work for one controller region.
- The *location service daemon* provides the CORBA location service in support of Remote Method Invocation and Internet Inter-ORB Protocol (RMI/IIOP). When a client makes a remote call to an enterprise bean, the daemon determines, with WLM, which server or servers are eligible to process the request. The daemon then routes the request to the selected server, which establishes a CORBA session with the client. Subsequent calls to the same enterprise bean flow directly over the established session.

The controller and its servant regions define the *server instance* of the application server on that system. Since multiple servant regions can be started by the controller region, the application server instance can process work in several Java Virtual Machines (JVMs).

In a clustered environment, the *server* can consist of several server instances, one in each node that participates in the cluster. The ability for the controller region to start multiple servants must be enabled. You can do this in the Administrative Console, by selecting **Servers** → **Application Servers** → ***server name*** → **Server Instance**. Under General Properties for Server Instance, select the **Multiple Instances Enabled** option (see Figure 14-1).

> **Note:** The term *Instances* in the Multiple Instances Enabled option is misleading, and does not refer to the server instance. Rather it refers to the servant region clones.

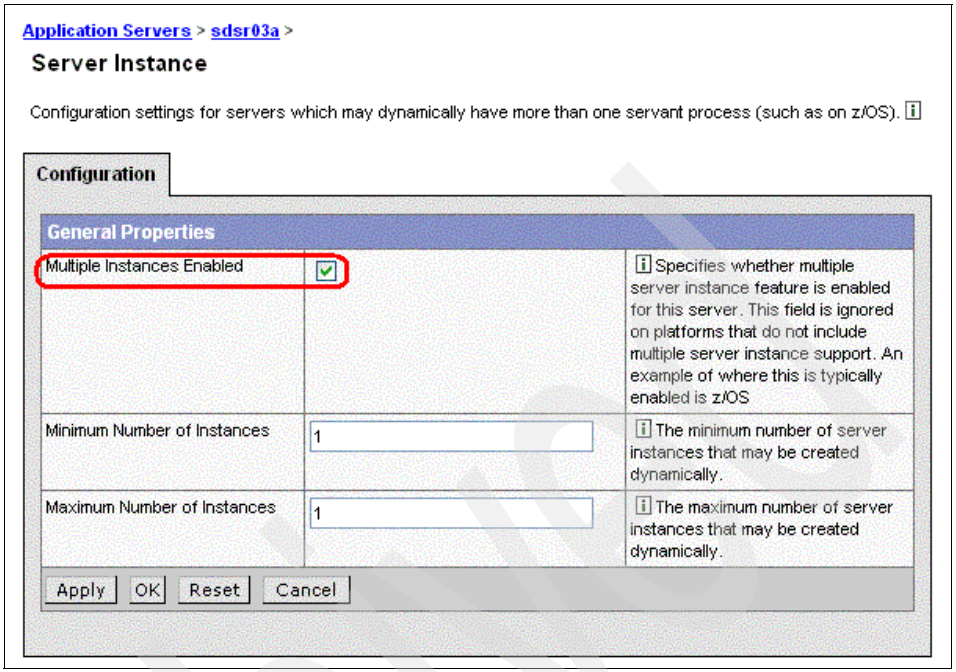



*Figure 14-1 Enabling multiple servants for an application server on z/OS*

All the previously mentioned WebSphere for z/OS processes have to be classified with sufficient performance goals for WLM. In general, controller regions are work routers and should have high priority. Servant regions are started and stopped dynamically by WLM, and require high priority so that initialization is performed quickly. After initialization is complete, the work is run in the servant region according to the goals of the client requests, and the servant region priority itself has no significance.

WebSphere for z/OS propagates the performance context of work requests through the use of WLM *enclaves*. Each transaction has its own enclave and is managed according to its enclave service class. For further guidelines, see Chapter 10, “Workload management on z/OS” on page 165.

## 14.2.2 WLM support for DNS lookups

Client applications connect to WebSphere Application Server through TCP/IP, using either HTTP or IIOP, with or without Secure Sockets Layer (SSL) encryption. While WLM may already have been involved in the routing of the request, using Sysplex Distributor or Multi-Node Load Balancer, here we discuss the usage of Domain Name System (DNS) workload balancing after the request has reached its destination z/OS system.

Upon receiving an incoming request that uses a generic host name common to the sysplex members, the DNS asks WLM for the best available system to which to route the request. WLM's decision is based upon the z/OS performance measurements of the available systems, including CPU, memory, and I/O utilization. We highly recommend that you use DNS workload balancing, also known as *DNS/WLM* and *Connection Optimization*, if you do not use a Sysplex Distributor or equivalent solution.

### 14.2.3 WLM and EJB lookups

EJB client applications run name server lookups to obtain direct references to the EJB object. A pure client uses a lookup string like the one in Example 14-1.

*Example 14-1 Java Naming and Directory Interface (JNDI) naming server lookup URL*

```
initialContext.lookup(
   "corbaname:iiop:hostname:bootstrap_port#
      cell/clusters/MyCluster/com/mycompany/accounting/AccountEJB");
```

In this example, *hostname* and *bootstrap port* correspond to those of the cluster. In practice, the host name is a generic host name for all the nodes in the cluster, or a common DVIPA address. WLM resolves it to a specific naming server in the cluster that listens on the specified bootstrap port, for example a Node Agent.

Figure 14-2 shows the sequence of communication exchanges between cluster components to provide the client application with a direct reference to the required EJB. The numbers in Figure 14-2 correspond to the following events:

1. A client requests a lookup to a generic bootstrap host.
2. WLM resolves the generic name to a specific node agent in the cluster.
3. The node agent returns an indirect interoperable object reference (IOR) to the client. It contains the generic host name of the location service daemons in the cluster.
4. The client Object Request Broker (ORB) sends the lookup request with the indirect ORB. WLM passes the indirect ORB to a suitable location service daemon.
5. The daemon consults WLM to determine a suitable application server that should handle the request.
6. The daemon returns a direct IOR to the client. It contains the forwarded address of the designated cluster member.
7. The client ORB uses the direct IOR to send the request directly to that cluster member.

This sequence of events is started when specific IIOP lookup requests are sent by clients, such as shown in Example 14-1.

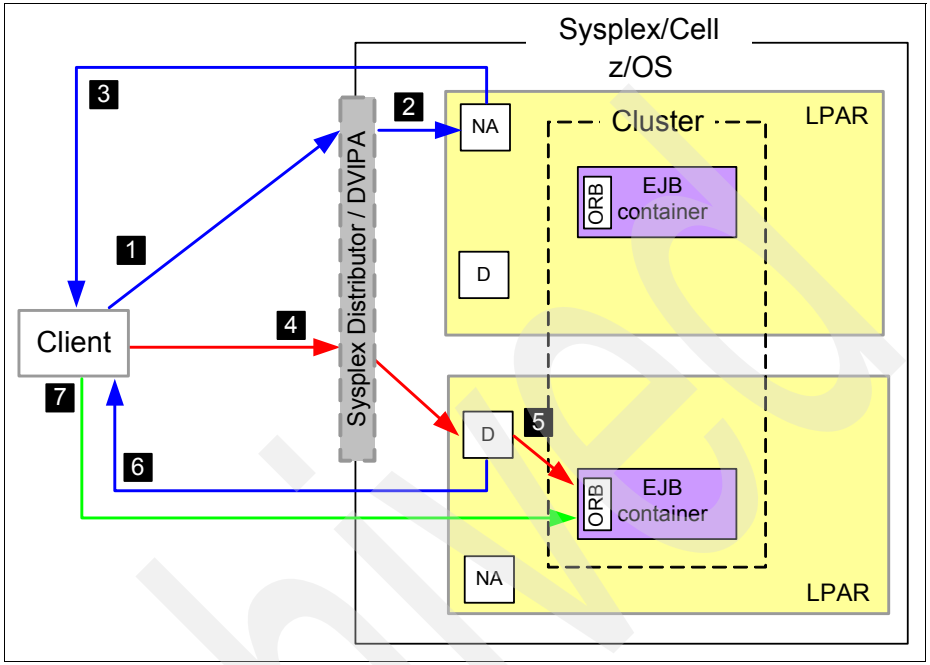



*Figure 14-2 Flow of EJB lookup events in a sysplex cluster*

**Note:** The WLM behavior shown in Figure 14-2 for z/OS is slightly different from the behavior of WebSphere Application Server on distributed platforms. On non-z/OS platforms, workload management of the EJB lookups is performed by a plug-in to the ORBs. The location service daemon returns the cluster configuration with the direct IOR to the WLM plug-in at the client's ORB. The actual destination is then chosen by this WLM plug-in. On z/OS, all workload decisions are made at the server level by z/OS Workload Manager.

If relative lookups are done by a server application in the cluster, for example as shown in Example 14-2, then the naming server of the server instance running the application will handle the initial lookup. Still, the location service daemon can supply a direct IOR to a different cluster member, based upon WLM information.

*Example 14-2 JNDI lookups relative to the server name space*

```
java.lang.Object ejbHome = initialContext.lookup(
      "cell/clusters/MyCluster/com/mycompany/accounting/AccountEJB");
```

The recommended coding of naming server lookups for server applications is to use object references in the deployment descriptor. In that case, the syntax uses the **`java:`** name space as a JNDI lookup provider, as shown in Example 14-3. The container that runs the application sets up this name space based on information from the deployment descriptor of the application.

*Example 14-3 JNDI lookups in the java: name space*

```
initialContext.lookup(
        "java:comp/env/com/mycompany/accounting/AccountEJB");
```

When working with EJB workload management, the transport mechanism that WebSphere Application Server for z/OS uses depends on whether the client is local or remote. If the client is remote (that is, not running on the same z/OS system), the transport is TCP/IP. If the client is local, the transport is through a program call. Local transport is fast because it avoids the physical trip over the network and eliminates data transforms. It also simplifies the marshalling of requests and uses optimized Resource Access Control Facilities (RACF) for security rather than having to invoke Kerberos or SSL.

Because of this, note the importance of the HiperSockets facilities on z/OS, which we discussed earlier. When the EJB client is an application in the server cluster, the preference to resolve EJB lookups to the local server can be indicated in the cluster definition. To do this, in the Administrative Console, navigate to **Servers** → **Clusters** → ***cluster name***. On the Configuration tab, make sure that the **Prefer local** check box is selected as shown in Figure 14-3.

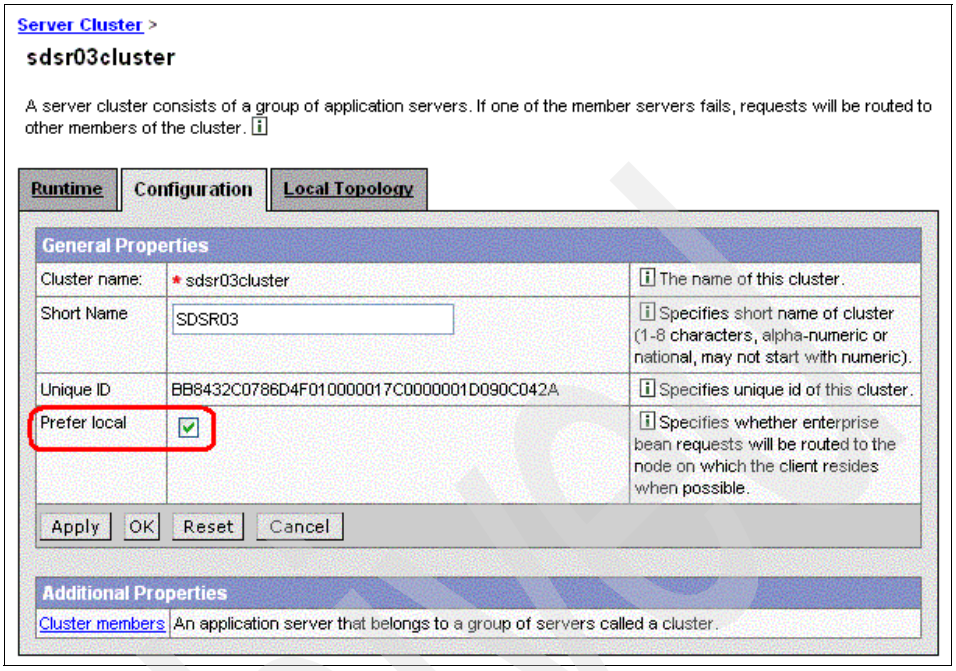



*Figure 14-3 Selecting the option Prefer local EJB requests in a cluster*

## 14.3 DB2 Data Sharing

The possibility of sharing information between sysplex members through the coupling facility is one of the major advantages of the z/OS run-time environment. One of the subsystems to exploit this feature is DB2.

Both application and system data can be stored in DB2 databases. The unique data sharing feature of DB2 for z/OS makes it possible to have a single catalog and directory for all the DB2 subsystems that participate in the Data Sharing Group that is defined in the sysplex. All the data for the DB2 members of the Data Sharing Group is stored on shared media that is attached to all systems in the sysplex. All the participating members can read and update the data simultaneously. The coupling facility guarantees cross-system locking of the resources, by means of so-called *group buffer pools*.

In case of failure of a DB2 subsystem on one member of the Data Sharing Group, another member can recover or back out the units of work that were in flight.

In the WebSphere environment discussed in this book, DB2 Data Sharing is exploited to store the databases of the BP Container and the BRBeans, as well as the application data. As a consequence, in our scenario we do not need data replication or other data failover solutions. Refer to 14.5, “Session affinity for Web applications” on page 265, to see how DB2 Data Sharing can resolve HTTP session affinity problems in a server cluster.

### Cloudscape not recommended in a z/OS production or sysplex environment

WBISF for z/OS includes a fully supported version of Cloudscape, an object-relational database system implemented in Java. On z/OS, this database system uses a UNIX file system, that is an HFS, to store the data.

Although we successfully tested our application in a WebSphere Base Server topology to store the BP Container and BRBeans system data, we recommend that you do not use Cloudscape in a z/OS production or sysplex environment. The major reason for this is that the embedded Cloudscape does not support simultaneous data access from different JVMs. While this is resolved by the Cloudscape Network Server, which is also shipped with the WebSphere product, this network server only supports non-XA network access with a JDBC Type 4 driver. Business Process Choreographer requires XA-enabled JDBC connections though.

## 14.4 Resource Recovery Services

Resource Recovery Services (RRS) is a base z/OS component that provides services to coordinate update and recovery of distributed resources within a sysplex. Since transactions running in WBISF for z/OS can access multiple resource managers such as DB2, CICS, IMS, and WebSphere MQ, there is a need for a resource coordinator with a sufficient subsystem scope. For several years now, the z/OS policy has indicated to centralize such resource coordination in the RRS of the operating system.

Rather than including its proprietary transaction coordination, WebSphere also exploits RRS. WebSphere supports RRS-enabled adapters for all the previously mentioned subsystems.

From the J2EE development point of view, RRS is a z/OS extension to the JCA resource adapter specifications. WebSphere for z/OS supports the J2EE Connector Architecture (JCA) 1.0. In addition to the transaction support defined by JCA, WebSphere for z/OS supports *RRSTransactional* support, which is a z/OS only extension to the architecture. Resource adapters that are capable of using RRS, and that properly indicate to WebSphere Application Server for z/OS

that they are RRSTransactional, will be supported as RRS compliant resource adapters.

RRS two-phase commit support is only applicable in a local environment, where the back end must reside on the system. CICS and IMS resources adapters may use RRS transactional support only when these adapters are configured to use local interfaces to their back-end resource manager, which as stated earlier, must reside on the same system as the IBM WebSphere Application Server for z/OS. To learn more about RRS, see the WebSphere information center at:

http://publib.boulder.ibm.com/infocenter/ws51help/topic/com.ibm.websphere.wbifz.doc/info/wbifz/ae/cdat_rrs.html

RRS is a z/OS subsystem that requires several log streams to be defined. These log streams reside in the coupling facility of our sysplex setup. If RRS is not yet enabled when WebSphere is installed, RRS can be installed as part of the WebSphere installation process.

## 14.5 Session affinity for Web applications

One of the concerns in self-service Web applications is that the user may have several network interactions with the application server to complete one request. Therefore, the application may need to store state information. In Web application servlets, use HTTP session objects to achieve this. By default, these are in-memory objects in the application server region. With Sysplex Distributor, consecutive network interactions may result in different application server regions. To overcome this affinity problem, two solutions are possible:

- Persist the session information.
- Use the HTTP server plug-in file plugin-cfg.xml in the Web server (redirector) node.

If you decide to persist the session information, and require a persistent implementation in a high availability scenario, you can use one of these two solutions:

- Database session persistence using a Data Sharing database
- Memory-to-memory session replication (DRS)

Refer to the redbook *Architecting High Availability e-business on IBM @server zSeries*, SG24-6850, for a complete discussion about all the possible solutions.

### 14.5.1 Stateful session EJBs in a workload balanced environment

There are implications of workload balancing on stateful-session EJBs. Generally, stateful-session EJBs are not well suited to a workload balanced environment. An active instance of a stateful-session EJB is tied to a particular application server. Stateful-session EJB mobility is achievable only through the passivation mechanism.

The WebSphere Application Server for z/OS V5.1 InfoCenter states:

> "This product supports the cloning of stateful session bean home objects among multiple application servers. However, it does not support the cloning of a specific instance of a stateful session bean. Each instance of a particular stateful session bean can exist in just one application server and can be accessed only by directing requests to that particular application server. State information for a stateful session bean cannot be maintained across multiple members of a server cluster."

The use of stateful-session EJBs is not considered best practice programming due to both their potential for misuse and apparent lack of scalability. Many application server providers, including IBM (via WebSphere on z/OS V5) provide enhanced support for stateful-session EJBs in workload balanced environments.

In particular, consider the situation of a long running transaction which results in an EJB being passivated (see the following section for explanation) on one server in the cluster. The transaction is resumed some time later, and the EJB is re-activated on a different server, perhaps on a different system in the sysplex. Both the EJB container and the developer must handle such situations correctly.

### 14.5.2 Passivation and activation

When a client application requires the use of an EJB, usually as the result of a business function, the application invokes the create() method of the EJB. In response to this, the EJB container instantiates the EJB, causing it to be added to the application server's EJB Session Cache. At this point, the EJB methods are available for use.

If there is an extended period where the EJB methods are not invoked, and the EJB containers' Instance Session Cache is full, then the EJB container chooses the EJBs to passivate, usually based on the last reference. Figure 14-4 illustrates this process.

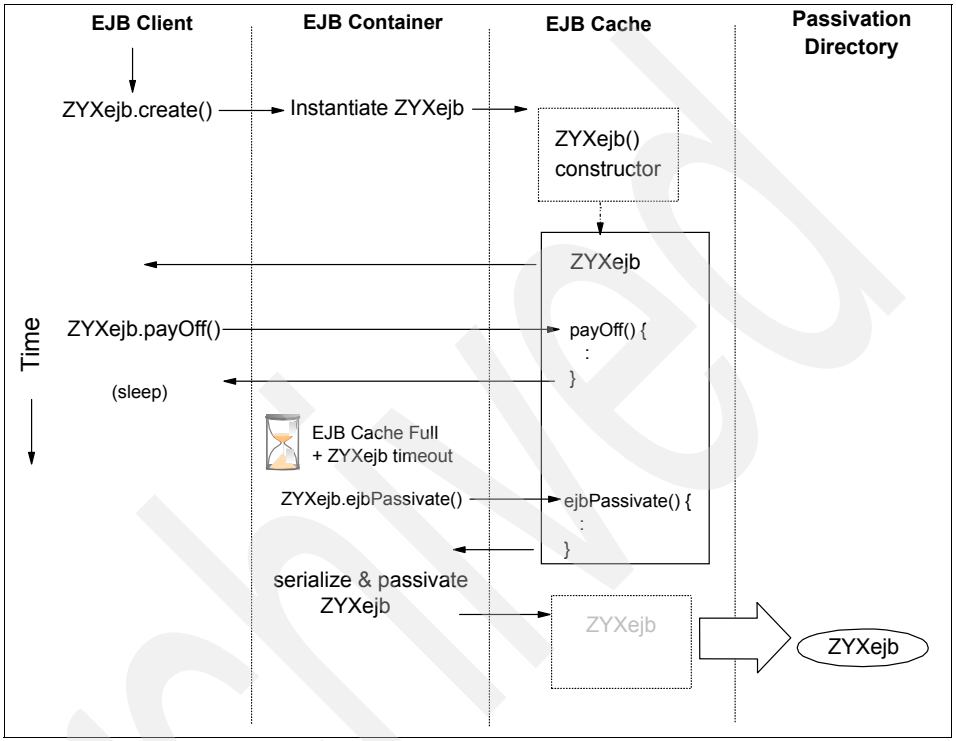



*Figure 14-4 EJB passivation*

When the EJB container passivates the EJB, it is serialized and written to disk. If and when it is required later, an arbitrary reference to one of the EJB methods triggers the EJB container to re-activate the EJB. This results in the EJB being deserialized, and made available to the client, as shown in Figure 14-5.

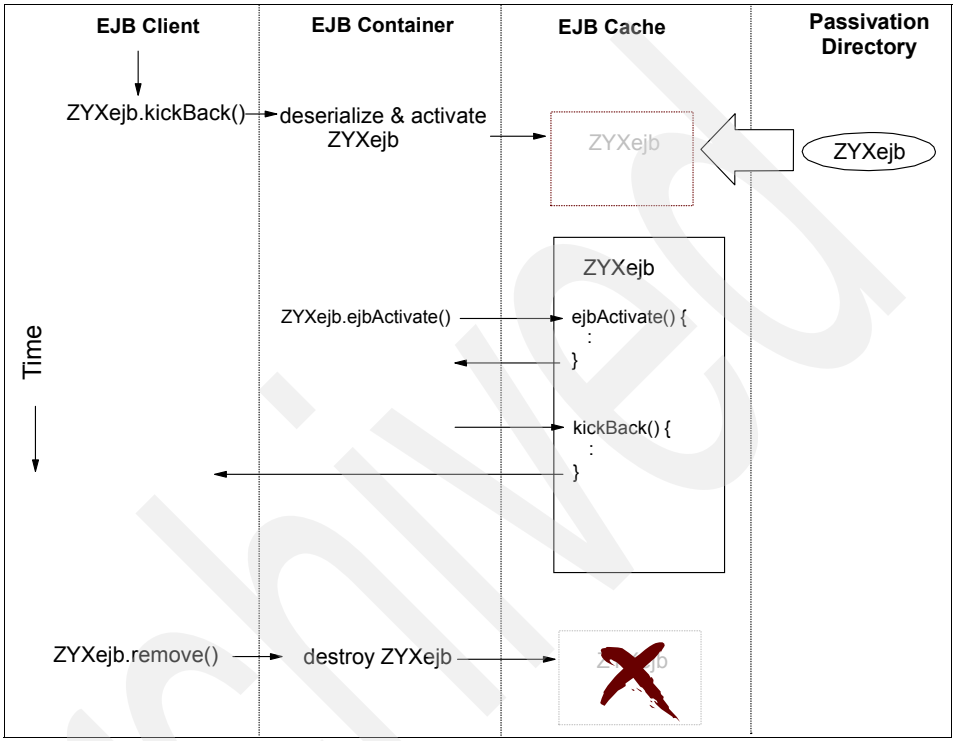



*Figure 14-5 EJB activation*

Clearly the passivation and activation process has some overheads, and you should avoid it if possible. The first step is to ensure the EJB Session Cache has sufficient capacity for the typical workload.

The recommendation in the WebSphere InfoCenter is to set the Cache size to the expected peak number of sessions in a typical workload. Setting this value too high will impact the application server's JVM free memory. Setting it too low greatly will increase the likelihood of unnecessary EJB passivations. Figure 14-6 shows the default settings in WebSphere Application Server.

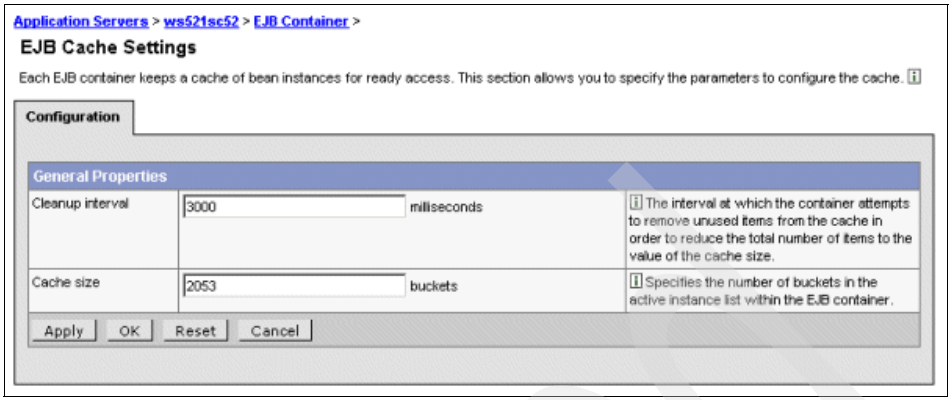



*Figure 14-6 Instance Session Cache Settings*

If the EJB session cache becomes full, the EJB container chooses a number of EJB instances to write-out to the passivation directory. The choice of EJB is left to the developer of the EJB container. In most cases, this is based on the least recently used (LRU) algorithm. This assumes the passivation directory is set to a valid directory location. You specify this setting by selecting **Servers** → **Application Servers** → ***server name*** → **EJB Container**. Figure 14-7 shows the EJB Container pane. By default, the passivation directory is set to ${USER_INSTALL_ROOT}/temp which is not appropriate if:

- It is a read-only file system.
- It is not large enough.
- The application server is part of a cluster supporting stateful session EJBs.

In these cases, you can perform either of the following actions:

- Create a new writable file-system to be mounted over ${USER_INSTALL_ROOT}/temp.
- Change the value of the passivation directory setting to a writable file system.

When the EJB container cannot passivate EJBs, WebSphere produces the warning message `CNTR0001W: A Stateful SessionBean could not be passivated`.

> **Note:** Message CNTR0001W is only a warning. You should check this, and if you don't, the application server JVM may eventually produce a java.lang.OutOfMemoryException and stall.

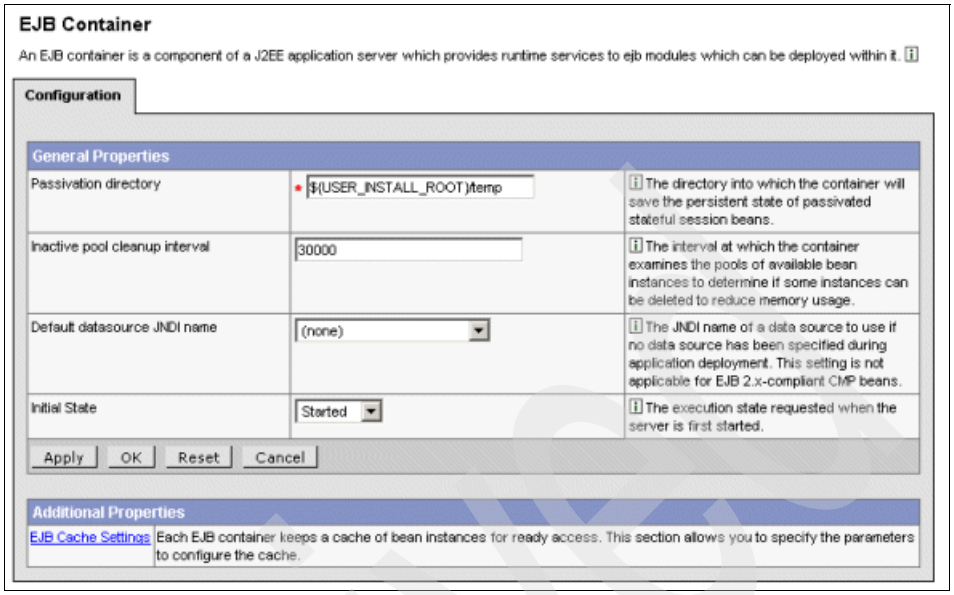



*Figure 14-7 EJB Container settings*

The EJB developer needs to address the possibility that a stateful session EJB will be passivated. In a workload balancing scenario, it is possible the reactivated EJB will run on a different application server, possibly even on a different member of the sysplex. Therefore, it makes no sense to expect such resources as databases connections, sockets, and threads to function as they did following reactivation.

Therefore the EJB specification provides for two methods: ejbPassivate() and ejbActivate() to be implemented by the bean developer. The ejbPassivate() method is called by the EJB container immediately prior to passivation of that EJB to that allow the EJB developer to release resources such as threads and database connections.

Non-serializable objects and very large objects in the EJB require special care. Consider the use of the Java modifier *transient* on such objects, which means they will be skipped during serialization. Supplementary code to restore or recreate such objects is required at reactivation.

The EJB is then serialized and written out as a flat file in the passivation directory until needed. When a reference is made to a passivated EJB, the EJB container deserializes the EJB back into the Instance Session Cache, and then invokes the ejbActivate method of the EJB. This gives the EJB developer the opportunity to re-establish those resources it was made to relinquish on passivation. This includes any non-serializable objects and transient-flagged resources.

This type of resource establishment is similar to processing that is normally done in the constructor method of the EJB. However, the constructor method is called only once in an EJB instance's lifetime. Rather than recode the resource initialization, factor out the initialization code into a new helper method, which is capable of being invoked from both the constructor and ejbActivate.

### 14.5.3 Security and performance implications of passivation

As indicated earlier, there are some significant overheads, both at development and run time to make passivation happen. Not much can be done to stop passivation from happening, apart from defining a very large instance session cache and an invalid passivation directory. Both of these approaches are risky. However, passivation is necessary to the integrity of the application servers, and only stateful-session EJBs can move between application server instances in their lifetime.

It is necessary to remember that Java object serialization is relatively slow, because it involves disk I/O. Add to this the extra work required to re-establish non-serializable resources, and it soon becomes apparent that passivation can be an expensive process. Defining a suitably sized EJB Session Cache and using good programming practices can alleviate unnecessary EJB retention resulting in unnecessary passivations.

One such way is described on the IBM Best Practices Web site, suggesting that EJB clients always invoke the EJB's remove() method once the business transaction is complete. See "Best Practice: Removing stateful session beans" on the Web at:

http://www-106.ibm.com/developerworks/websphere/library/bestpractices/removing_stateful_session_beans.html

Consider also the security of the passivation directory. Serialized objects are not encrypted or encoded. They are more or less clear text files in the passivation directory. If these objects contain potentially sensitive or secret data, then in the first instance, the passivation directory should be protected from public view. At a minimum, take the following precautions:

- Ensure that the permissions are 600 on the passivation directory itself.
- Set the WebSphere user ID to have suitable *umask*. The value 0077 is a good starting point. It removes all access to newly created files at both the group and public levels.

For extra security, consider making the bean implement the *externalizable interface* instead of the *serializable* interface. This places the burden of creating serialization methods on the EJB developer, but allows for encryption of the serialized object.

### 14.5.4 Using a shared HFS for the passivation directory on z/OS

You must ensure that a suitably sized shared hierarchical file system (HFS) is defined on OS/390 2.9 or newer operating systems. With SYSPLEX(YES) specified in the BPXPRMxx member of SYS1.PARMLIB, you set the passivation directory to point to this location, because sharing is on by default.

Creation of a shared HFS is straightforward and is well documented in *OS/390 V2R10.0 UNIX System Services Planning*, SC28-1890. You simply create the HFS and mount it over a directory in the USS file system. Figure 14-8 shows how to update the passivation directory setting in the EJB Container settings.

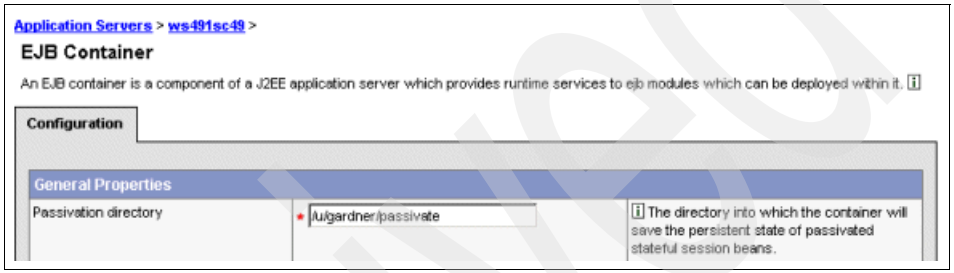



*Figure 14-8 EJB Container settings with shared HFS in place*

A

# ITSO test configurations

In this appendix, we describe the hardware configuration of our test environment. We also explain the software that we installed and any relevant configuration data.

# A.1 Monoplex systems (SC49, SC66)

Our monoplex systems were constructed as a stepping stone to the full sysplex enabled WebSphere Application Server Network Deployment. We used two separate logical partitions (LPARs) for this test.

## A.1.1 Hardware and operating systems

Table A-1 shows the specific zSeries models, z/OS versions, and the real memory that we used for our test scenarios for the two systems SC49 and SC66 in the monoplex environment.

*Table A-1 Hardware and operating system data*

| | **SC49** | **SC66** |
|---|---|---|
| Hardware | zSeries 900 model 2084-B16 | zSeries 900 model 2064-1C7 |
| Operating system | z/OS R1.5 | z/OS R1.5 |
| Real memory | 4096 MB | 4032 MB |

## A.1.2 Software configuration

This section specifies the software and versions that we used for our test scenarios.

### WebSphere Business Integration Server Foundation

Each server had WebSphere Application Server V5.1 at PTF level W510003. This included support for WebSphere Business Integration Server Foundation (WBISF), including Business Process Execution Language (BPEL) and extensions. Each WBISF server was configured as a stand-alone node.

### HFS structure

The base hierarchical file system (HFS) structure for WBISF was /SC49/WebSpherePB/V5R1M0/BS01.

### Configuration files

The following sections show the contents of a few of the many configuration files that were created during the installation and configuration of WBISF. You can find more information in the ASCII format XML files. Almost every level of WebSphere BI Server (that is cell, node, application server, application, and so on) has a resources XML file.

### *was.env*

The was.env file is the major configuration file used by WebSphere BI Server Foundation on startup. Example A-1 shows the contents of this file.

*Example: A-1 The was.env file*

```
#was.env  ::  ws491sc49  ::  Mon Sep 27 18:01:01 GMT 2004
#-------------------------------------
APP_INSTALL_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/installedApps
CLOUDSCAPE_JDBC_DRIVER_PATH=/WebSpherePB/V5R1M0/BS01/AppServer/cloudscape/lib
CONFIG_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/config
CONNECTOR_INSTALL_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/in
talledConnectors
DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH=/usr/lpp/db2/db2810/jcc/lib
DB2UNIVERSAL_JDBC_DRIVER_PATH=/usr/lpp/db2/db2810/jcc/classes
DEPLOY_TOOL_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/deploytool/itp
DRIVER_PATH=/WebSpherePB/V5R1M0/BS01/AppServer
ICU_DATA=/WebSpherePB/V5R1M0/BS01/AppServer/bin/JAVA_HOME=/usr/lpp/java/J1.4/
JOBNAME=WS491D
LOG_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/logs
MQJMS_LIB_ROOT=/usr/lpp/mqm/V5R3M1/java/lib
MQ_INSTALL_ROOT=/usr/lpp/mqm/V5R3M1
PATH=/bin
SERVER_LOG_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/logs/ws491sc49
TRANLOG_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer/tranlog
UNIVERSAL_JDBC_DRIVER_PATH=/WebSpherePB/V5R1M0/BS01/AppServer/universalDriver/lib
USER_INSTALL_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer
WAS_CELL=cel491
WAS_DEFAULT_CONTROL_PROC_NAME=WS5491C
WAS_DEFAULT_SERVANT_PROC_NAME=WS5491S
WAS_ETC_DIR=/WebSpherePB/V5R1M0/BS01/AppServer/etc
WAS_HOME=/WebSpherePB/V5R1M0/BS01/AppServer
WAS_INSTALL_ROOT=/WebSpherePB/V5R1M0/BS01/AppServer
WAS_LIBS_DIR=/WebSpherePB/V5R1M0/BS01/AppServer/lib
WAS_NODE=nd491sc49
WAS_PROPS_DIR=/WebSpherePB/V5R1M0/BS01/AppServer/properties
WAS_TEMP_DIR=/WebSpherePB/V5R1M0/BS01/AppServer/temp
cell_name=cel491
cell_short_name=CL491
clustered_server=0
com.ibm.security.useFIPS=0
com_ibm_CSI_claimClientAuthenticationRequired=0
com_ibm_CSI_claimClientAuthenticationSupported=1
com_ibm_CSI_claimClientAuthenticationtype=SAFUSERIDPASSWORD
com_ibm_CSI_claimIdentityAssertionSupported=0
com_ibm_CSI_claimIdentityAssertionTypeCert=0
com_ibm_CSI_claimIdentityAssertionTypeDN=0
com_ibm_CSI_claimIdentityAssertionTypeSAF=0
com_ibm_CSI_claimKeyringName=WASKeyring
```

```
com_ibm_CSI_claimMessageConfidentialityRequired=0
com_ibm_CSI_claimSecurityLevel=HIGH
com_ibm_CSI_claimStateful=1
com_ibm_CSI_claimTLClientAuthenticationRequired=0
com_ibm_CSI_claimTLClientAuthenticationSupported=1
com_ibm_CSI_claimTransportAssocSSLTLSRequired=0
com_ibm_CSI_claimTransportAssocSSLTLSSupported=1
com_ibm_CSI_claim_ssl_sys_v3_timeout=600
com_ibm_CSI_performClientAuthenticationRequired=0
com_ibm_CSI_performClientAuthenticationSupported=1
com_ibm_CSI_performClientAuthenticationtype=SAFUSERIDPASSWORD
com_ibm_CSI_performIdentityAssertionRequired=0
com_ibm_CSI_performIdentityAssertionSupported=0
com_ibm_CSI_performKeyringName=WASKeyring
com_ibm_CSI_performMessageConfidentialityRequired=0
com_ibm_CSI_performMessageConfidentialitySupported=1
com_ibm_CSI_performMessageIntegrityRequired=1
com_ibm_CSI_performMessageIntegritySupported=1
com_ibm_CSI_performSecurityLevel=HIGH
com_ibm_CSI_performStateful=1
com_ibm_CSI_performTLClientAuthenticationRequired=0
com_ibm_CSI_performTLClientAuthenticationSupported=0
com_ibm_CSI_performTransportAssocSSLTLSRequired=0
com_ibm_CSI_performTransportAssocSSLTLSSupported=1
com_ibm_CSI_perform_ssl_sys_v3_timeout=600
com_ibm_HTTP_claimClientAuthentication=0
com_ibm_HTTP_claimKeyringName=WASKeyring
com_ibm_HTTP_claimSecurityLevel=HIGH
com_ibm_HTTP_claim_sslEnabled=1
com_ibm_HTTP_claims_ssl_sys_v3_timeout=600
com_ibm_Server_Security_Enabled=1
com_ibm_authMechanisms_type_OID=No OID for this mechanism
com_ibm_security_SAF_EJBROLE_Audit_Messages_Suppress=0
com_ibm_security_SAF_unauthenticated=WSGUEST
com_ibm_userRegistries_type=security:LocalOSUserRegistry
control_region_classpath=/WebSpherePB/V5R1M0/BS01/AppServer/properties:/WebSpherePB/V5R1M0/BS01
/AppServer/lib/bootstrap.jar:/WebSpherePB/V5R1M0/BS01/AppServer/lib/j2ee.jar:/WebSpherePB/V5R1M
0/BS01/AppServer/lib/lmproxy.jar
control_region_jvm_properties_file=/WebSpherePB/V5R1M0/BS01/AppServer/config/cells/cel491/nodes
/nd491sc49/servers/ws491sc49/control.jvm.options
control_region_libpath=/usr/lpp/java/J1.4/bin:/usr/lpp/java/J1.4/bin/classic:/WebSpherePB/V5R1M
0/BS01/AppServer/lib
control_region_security_enable_trusted_applications=1
control_region_thread_pool_size=25
control_region_wlm_dispatch_timeout=300
daemonInstanceName=WS491D
daemonName=CL491
daemon_group_name=CL491
daemon_start_command=START WS5491D
```

```
daemon_start_command_args=JOBNAME=WS491D,ENV=CL491.CL491.WS491D
daemon_was_env_file=/WebSpherePB/V5R1M0/BS01/Daemon/config/cells/cel491/CL491/WS
491D/was.env
node_name=nd491sc49
node_short_name=ND491
nonauthenticated_clients_allowed=1
primordial_root=/WebSpherePB/V5R1M0/BS01
private_bboo_jvm_in_ctl=1
protocol_http_defaultIdentity=WSGUEST
protocol_http_enableSSLTracking=0
protocol_http_listenIPAddress=*
protocol_http_port=9015
protocol_http_session_persistent=0
protocol_http_timeout_input=900
protocol_http_transport_network_qos=NONE
protocol_https_default_identity=WSGUEST
protocol_https_listenIPAddress=*
protocol_https_port=9016
protocol_iiop_daemon_listenIPAddress=wtsc49.itso.ibm.com
protocol_iiop_daemon_port=9018
protocol_iiop_daemon_port_ssl=9019
protocol_iiop_listenIPAddress=*
protocol_iiop_local_timeout=1800
protocol_iiop_no_local_copies=0
protocol_iiop_port=9013
protocol_iiop_port_ssl=9014
protocol_iiop_server_session_keepalive=0
protocol_iiop_server_session_keepalive_ssl=0
ras_trace_BufferCount=4
ras_trace_BufferSize=128K
ras_trace_ctraceParms=50
ras_trace_defaultTracingLevel=1
ras_trace_outputLocation=SYSPRINT BUFFER
read_license_agreement=1
security.enablePluggableAuthentication=1
security_EnableRunAsIdentity=0
security_EnableSyncToOSThread=0
security_assertedID_IBM_accepted=0
security_assertedID_IBM_sent=0
security_kerberos_allowed=0
security_local_identity=WSGUEST
security_remote_identity=WSGUEST
security_sslClientCerts_allowed=0
security_sslType1=0
security_userid_passticket_allowed=1
security_userid_password_allowed=0
security_zOS_domainType=0
security_zSAS_ssl_repertoire=nd491sc49/DefaultIIOPSSL
server_configured_system_name=SC49
```

```
server_generic_short_name=CLU491server_generic_uuid=BB6035BF1FE130C7000001140000007090C041E
server_region_classpath=/WebSpherePB/V5R1M0/BS01/AppServer/properties:/WebSphere
PB/V5R1M0/BS01/AppServer/lib/bootstrap.jar:/WebSpherePB/V5R1M0/BS01/AppServer/lib/urlprotocols.
jar:/WebSpherePB/V5R1M0/BS01/AppServer/lib/j2ee.jar:/WebSpherePB/
V5R1M0/BS01/AppServer/lib/lmproxy.jar
server_region_dynapplenv_jclparms=JOBNAME=&IWMSSNM.S,ENV=CL491.ND491.&IWMSSNM
server_region_dynapplenv_jclproc=WS5491S
server_region_jvm_properties_file=/WebSpherePB/V5R1M0/BS01/AppServer/config/cells/cel491/nodes/
nd491sc49/servers/ws491sc49/servant.jvm.optionsserver_region_libpath=/usr/lpp/java/J1.4/bin:/us
r/lpp/java/J1.4/bin/classic:/WebSpherePB/V5R1M0/BS01/AppServer/lib
server_region_workload_profile=LONGWAIT
server_specific_name=ws491sc49
server_specific_short_name=WS491
server_specific_uuid=BB6035BF1EE0571200000114000000007090C041E
server_type=AppServer
shell_command_proc_name=BBOW5SH
topology_server_BB6035BF1EE0571200000114000000007090C041E=1
topology_server_servernamefromuuidBB6035BF1EE0571200000114000000007090C041E=WS491
transaction_defaultTimeout=300
transaction_maximumTimeout=300
was_env_file=/WebSpherePB/V5R1M0/BS01/AppServer/config/cells/cel491/nodes/nd491s
c49/servers/ws491sc49/was.env
wlm_dynapplenv_single_server=1
wlm_maximumSRCount=1
wlm_minimumSRCount=1
```

### *control.jvm.options*

Located deep in the HFS, the control.jvm.options file contains parameters to be passed to the Control Region JVM. Example A-2 shows the contents of this file.

*Example: A-2 The control.jvm.options file*

```
/WebSpherePB/V5R1M0/BS01/AppServer/config/cells/cel491/nodes/nd491sc49/servers/ws491sc49/control.jvm.options
-DtraceSettingsFile=config/cells/cel491/nodes/nd491sc49/servers/ws491sc49/trace.dat
-Dws390.nodename=nd491sc49
-Dwbisf.smpe.install.root=/SC49/zWBISF51
-Djava.security.policy=/WebSpherePB/V5R1M0/BS01/AppServer/properties/server.policy
-Djava.security.auth.login.config=/WebSpherePB/V5R1M0/BS01/AppServer/properties/wsjaas.conf
-Dwas.repository.root=/WebSpherePB/V5R1M0/BS01/AppServer/config
-Dfile.encoding=ISO8859-1
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
-Dws.ext.dirs=/usr/lpp/java/J1.4/lib:/WebSpherePB/V5R1M0/BS01/AppServer/classes:/WebSpherePB/V5
R1M0/BS01/AppServer/lib:/WebSpherePB/V5R1M0/BS01/AppServer/lib/ext:/WebSpherePB/V5R1M0/BS01/App
Server:/WebSpherePB/V5R1M0/BS01/AppServer/web/help:/WebSpherePB/V5R1M0/BS01/AppServer/deploytoo
l/itp/plugins/com.ibm.etools.ejbdeploy/runtime
-Dserver.root=/WebSpherePB/V5R1M0/BS01/AppServer
-Dcom.ibm.itp.location=/WebSpherePB/V5R1M0/BS01/AppServer/bin
-Dwas.install.root=/WebSpherePB/V5R1M0/BS01/AppServer
-Dwas.history.dir=/tmp
-Xms48m
-Dsecurity.use.localos.authorization=true
-Djava.awt.headless=true
-Djava.ext.dirs=/usr/lpp/java/J1.4/lib/ext:/WebSpherePB/V5R1M0/BS01/AppServer/java/jre/lib/ext
-Xmx128m
-Djava.home=/usr/lpp/java/J1.4/
-Duser.install.root=/WebSpherePB/V5R1M0/BS01/AppServer
-Dsmpe.install.root=/usr/lpp/zWebSphere/V5R1M0
```

### *servant.jvm.options*

The servant.jvm.options file contains the parameters that are passed to the Servant Region JVM at startup. Example A-3 shows the contents of this file.

*Example: A-3 The servant.jvm.options file*

**/WebSpherePB/V5R1M0/BS01/AppServer/config/cells/cel491/nodes/nd491sc49/servers/ws491sc49/servant.jvm.options**

```
-Xmx256m
-DtraceSettingsFile=config/cells/cel491/nodes/nd491sc49/servers/ws491sc49/trace.dat
-Dws390.nodename=nd491sc49
-Dwbisf.smpe.install.root=/SC49/zWBISF51
-Djava.security.policy=/WebSpherePB/V5R1M0/BS01/AppServer/properties/server.policy
-Djava.security.auth.login.config=/WebSpherePB/V5R1M0/BS01/AppServer/properties/wsjaas.conf
-Dwas.repository.root=/WebSpherePB/V5R1M0/BS01/AppServer/config
-Dfile.encoding=ISO8859-1
-Xifa:project2
-Ddb2.jcc.propertiesFile=/SC49/WebSpherePB/V5R1M0/BS01/AppServer/etc/db2jcc.properties
-Duser.dir=/WebSpherePB/V5R1M0/BS01/AppServer
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
-Dws.ext.dirs=/usr/lpp/java/J1.4/lib:/WebSpherePB/V5R1M0/BS01/AppServer/classes:/WebSpherePB/V5
R1M0/BS01/AppServer/lib:/WebSpherePB/V5R1M0/BS01/AppServer/lib/ext:/WebSpherePB/V5R1M0/BS01/App
Server:/WebSpherePB/V5R1M0/BS01/AppServer/web/help:/WebSpherePB/V5R1M0/BS01/AppServer/deploytoo
l/itp/plugins/com.ibm.etools.ejbdeploy/runtime
-Dserver.root=/WebSpherePB/V5R1M0/BS01/AppServer
-Dcom.ibm.CORBA.iiop.noLocalCopies=false
-DbrbPropertiesFile=/SC49/WebSpherePB/V5R1M0/BS01/AppServer/etc/PDK-brbeansDefaultProperties
-Dcom.ibm.itp.location=/WebSpherePB/V5R1M0/BS01/AppServer/bin
-Dwas.install.root=/WebSpherePB/V5R1M0/BS01/AppServer
-Dwas.history.dir=/tmp
-Dsecurity.use.localos.authorization=true
-Djava.awt.headless=true
-Djava.ext.dirs=/usr/lpp/java/J1.4/lib/ext:/WebSpherePB/V5R1M0/BS01/AppServer/java/jre/lib/ext
-Xms256m
-Djava.home=/usr/lpp/java/J1.4/
-Duser.install.root=/WebSpherePB/V5R1M0/BS01/AppServer
-Dsmpe.install.root=/usr/lpp/zWebSphere/V5R1M0
```

### *resources.xml*

The following resources.xml files are located in the HFS under the /WebSpherePB/V5R1M0/BS01/AppServer directory:

- resources.xml
- config/cells/cel491/resources.xml
- config/cells/cel491/nodes/nd491sc49/resources.xml
- config/cells/cel491/nodes/nd491sc49/servers/ws491sc49/resources.xml
- config/templates/default/nodes/servers/server1/resources.xml
- config/templates/system/nodes/servers/jmsserver/resources.xml

## DB2

The DB2 subsystem that we used was at version 8.1. It provided a datasharing environment under the group DB2 subsystem ID D8FG. Table A-2 specifies the DB2 related settings for the two test systems.

*Table A-2 The DB2 environment (monoplex)*

| | **SC49** | **SC66** |
|---|---|---|
| Subsystem ID | D8F1 | D8F2 |
| JDBC HFS | /usr/lpp/db2/db2810 | /usr/lpp/db2/db2810 |
| Location ID | DB8F | DB8F |
| TCPPort | 38050 | 38050 |
| RESPort | 38051 | 38052 |

## CICS

A CICS region named CICSPAME was created on SC66 only. This region was CICS Transaction Server 2.2. This single CICS region was also used for the sysplex configuration.

## CICS Transaction Gateway

The CICS Transaction Gateway ran on SC66 as a started task. On startup, it uses the values listed in Example A-4 from /ctg/scsctg51/CTG.INI.

*Example: A-4 CICS Transaction Gateway values*

```
SECTION GATEWAY
closetimeout=10000
ecigenericreplies=off
initconnect=10
initworker=10
maxconnect=90
maxworker=90
noinput=off
nonames=on
notime=off
#    tfile=ctg.trc
workertimeout=10000
protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
protocol@tcp.parameters=connecttimeout=2000;idletimeout=600000;pingfrequency=60000;port=2006;so
linger=0;sotimeout=1000;
#   protocol@systemssl_ssl.handler=com.ibm.ctg.server.GskSslHandler
#   protocol@systemssl_ssl.parameters=port=8050;sotimeout=1000;connecttimeout=2000;idletimeout=
600000;pingfrequency=60000;keyring=/ctg/scsctg3/systemssl.kdb;keyringpw=oct2000;clientauth=off;
#protocol@systemssl_ssl.parameters=port=8050;sotimeout=1000;connecttimeout=2000;idletimeout=600
000;pingfrequency=60000;keyring=/u/wakelin/ssl/systemssl2.kdb;keyringpw=oct2000;clientauth=off;
#
ENDSECTION
```

## WebSphere MQ

WebSphere MQ ran on SC49 and SC66 with the configuration specified in Table A-3. These WebSphere MQ subsystems are capable of queue sharing. However, this feature was not used for the WBISF examples in this book.

*Table A-3 WebSphere MQ configuration*

| | SC49 | SC66 |
|---|---|---|
| Subsystem names | MQFN | MQFI |
| Listener ports | 1490 | 1490 |
| WMQ Java HFS | /usr/lpp/mqm/V5R3M1 | /usr/lpp/mqm/V5R3M1 |

# A.2 Sysplex systems (SC48, SC52)

This section describes the physical setup of the full sysplex configuration used for our examples.

## A.2.1 Hardware and operating systems

Table A-4 shows the specific zSeries models, z/OS versions, and the real memory that we used for our test scenarios for the two systems SC49 and SC66 in the sysplex environment.

*Table A-4 Hardware and operating system data*

| | **SC48** | **SC52** |
|---|---|---|
| Hardware | zSeries 990 model 2084-B16 | zSeries 990 model 2084-B16 |
| Operating system | z/OS R1.5 | z/OS R1.5 |
| Real memory | 6144 MB | 6144 MB |

## A.2.2 Software configuration

This section specifies the software and versions that we used for our test scenarios. Figure A-1 shows an overview of the architecture.

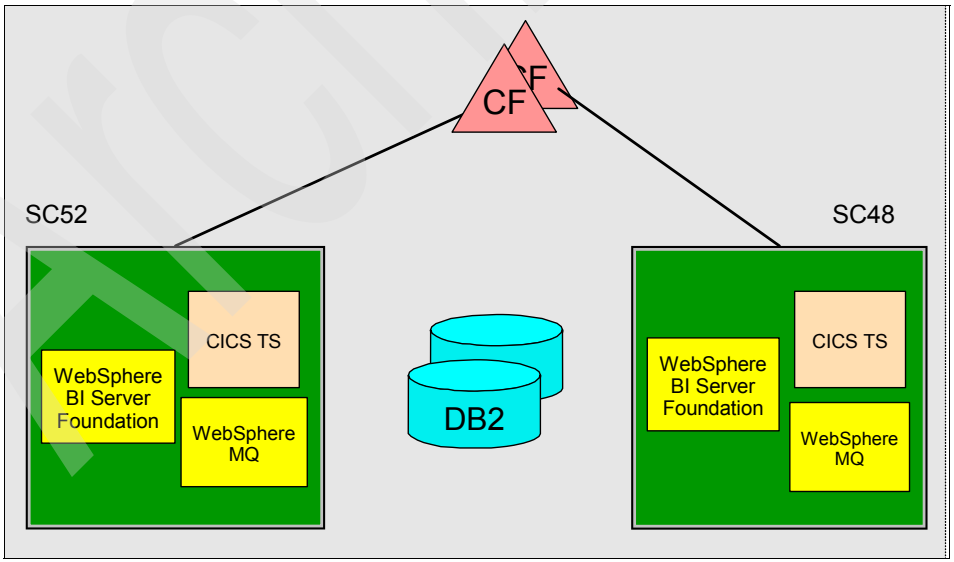



*Figure A-1 Parallel Sysplex members in the run-time setup*

## WebSphere Business Integration Server Foundation

We installed a Network Deployment configuration of WebSphere Business Integration Server Foundation. We refer to *WebSphere Business Integration Server Foundation V5.1 for z/OS*, SG24-6382, for details to set up such an infrastructure. As shown in Figure A-2, it consists of:

- One cell, called *sdcell*, spanning the two sysplex LPARs SC52 and SC48
- One node on each LPAR: *sdnodea* on SC52 and *sdnodeb* on SC48
- On each node, one active application server: *sdsr03a* on SC52 and *sdsr03b* on SC48
- One cluster defined across both LPARs: *sdsr03cluster*, which includes both application servers *sdsr03a* and *sdsr03b*
- One node, *sddmnode*, on SC52 running the Deployment Manager

**Note:** On z/OS, all servers have a *long name* and a *short name*. In this text, the long names are used. In Figure A-2, the servers are labelled by their short names.

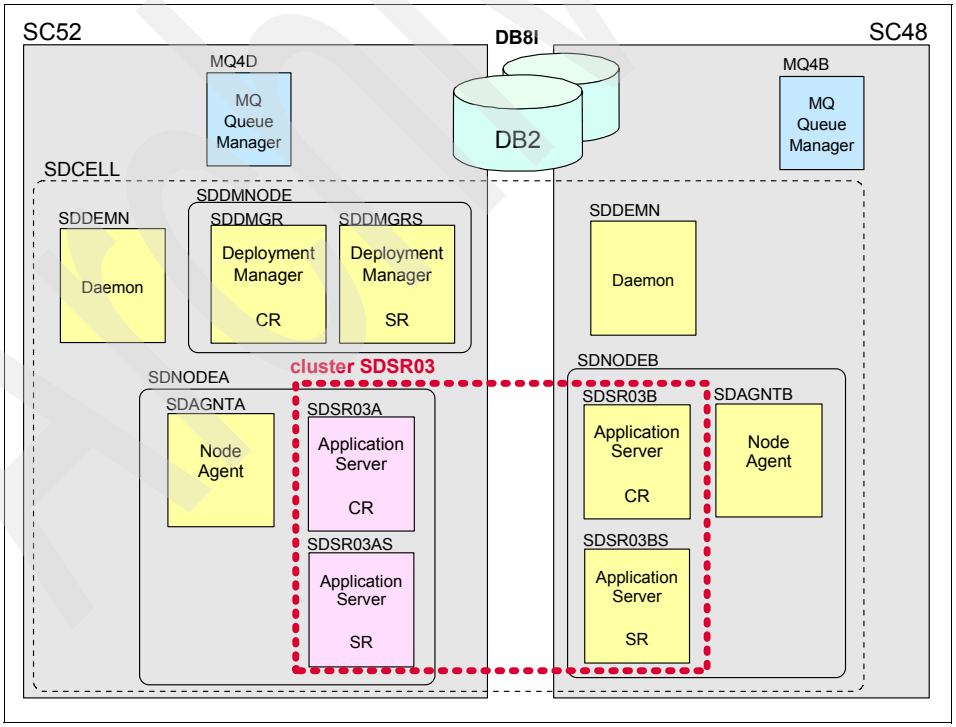



*Figure A-2 Network deployment configuration on the sysplex*

From the Administrative Console in the Deployment manager, you see the cluster topology as shown in Figure A-3.

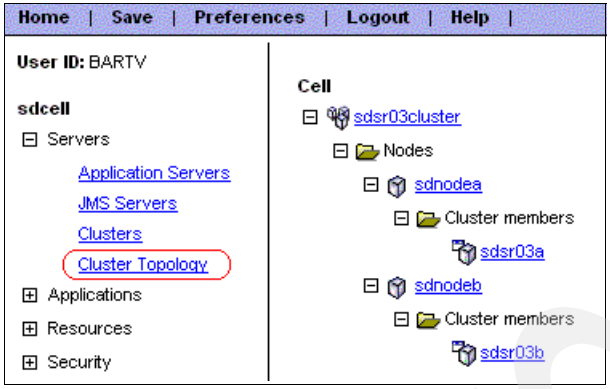



*Figure A-3 Cluster topology from the Admin Console*

## HFS structure

The base HFS for WebSphere BI Server Foundation is /WebSphereTV/V5R1M0/BS01.

## Configuration files

The following sections show the contents of a few of the configuration files created during the installation and configuration of WBISF. More information is contained in the ASCII format XML files. Almost every level of WBISF (that is cell, node, application server, application, and so on) has a resources XML file.

### *was.env*

The was.env file is the major configuration file used by WebSphere BI Server Foundation on startup.

*Example: A-5 The was.env file*

```
#was.env  ::  ws521sc52  ::  Wed Sep 29 20:01:41 GMT 2004
#-------------------------------------
APP_INSTALL_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/installedApps
CLOUDSCAPE_JDBC_DRIVER_PATH=/WebSphereTV/V5R1M0/BS01/AppServer/cloudscape/lib
CONFIG_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/config
CONNECTOR_INSTALL_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/installedConnectors
DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH=/usr/lpp/db2/db2810/jcc/lib
DB2UNIVERSAL_JDBC_DRIVER_PATH=/usr/lpp/db2/db2810/jcc/classes
DEPLOY_TOOL_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/deploytool/itp
DRIVER_PATH=/WebSphereTV/V5R1M0/BS01/AppServer
ICU_DATA=/WebSphereTV/V5R1M0/BS01/AppServer/bin/
JAVA_HOME=/usr/lpp/java/J1.4
```

```
JOBNAME=WS521D
LOG_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/logs
MQJMS_LIB_ROOT=/usr/lpp/mqm/V5R3M1/java/lib
MQ_INSTALL_ROOT=/usr/lpp/mqm/V5R3M1
PATH=/bin
SERVER_LOG_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/logs/ws521sc52
TRANLOG_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer/tranlog
UNIVERSAL_JDBC_DRIVER_PATH=/WebSphereTV/V5R1M0/BS01/AppServer/universalDriver/lib
USER_INSTALL_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer
WAS_CELL=cel521
WAS_DEFAULT_CONTROL_PROC_NAME=WS5521C
WAS_DEFAULT_SERVANT_PROC_NAME=WS5521S
WAS_ETC_DIR=/WebSphereTV/V5R1M0/BS01/AppServer/etc
WAS_HOME=/WebSphereTV/V5R1M0/BS01/AppServer
WAS_INSTALL_ROOT=/WebSphereTV/V5R1M0/BS01/AppServer
WAS_LIBS_DIR=/WebSphereTV/V5R1M0/BS01/AppServer/lib
WAS_NODE=nd521sc52
WAS_PROPS_DIR=/WebSphereTV/V5R1M0/BS01/AppServer/properties
WAS_TEMP_DIR=/WebSphereTV/V5R1M0/BS01/AppServer/temp
cell_name=cel521
cell_short_name=CL521
clustered_server=0
com.ibm.security.useFIPS=0
com_ibm_CSI_claimClientAuthenticationRequired=0
com_ibm_CSI_claimClientAuthenticationSupported=1
com_ibm_CSI_claimClientAuthenticationtype=SAFUSERIDPASSWORD
com_ibm_CSI_claimIdentityAssertionSupported=0
com_ibm_CSI_claimIdentityAssertionTypeCert=0
com_ibm_CSI_claimIdentityAssertionTypeDN=0
com_ibm_CSI_claimIdentityAssertionTypeSAF=0
com_ibm_CSI_claimKeyringName=WASKeyring
com_ibm_CSI_claimMessageConfidentialityRequired=0
com_ibm_CSI_claimSecurityLevel=HIGH
com_ibm_CSI_claimStateful=1
com_ibm_CSI_claimTLClientAuthenticationRequired=0
com_ibm_CSI_claimTLClientAuthenticationSupported=1
com_ibm_CSI_claimTransportAssocSSLTLSRequired=0
com_ibm_CSI_claimTransportAssocSSLTLSSupported=1
com_ibm_CSI_claim_ssl_sys_v3_timeout=600
com_ibm_CSI_performClientAuthenticationRequired=0
com_ibm_CSI_performClientAuthenticationSupported=1
com_ibm_CSI_performClientAuthenticationtype=SAFUSERIDPASSWORD
com_ibm_CSI_performIdentityAssertionRequired=0
com_ibm_CSI_performIdentityAssertionSupported=0
com_ibm_CSI_performKeyringName=WASKeyring
com_ibm_CSI_performMessageConfidentialityRequired=0
com_ibm_CSI_performMessageConfidentialitySupported=1
com_ibm_CSI_performMessageIntegrityRequired=1
com_ibm_CSI_performMessageIntegritySupported=1
```

```
com_ibm_CSI_performSecurityLevel=HIGH
com_ibm_CSI_performStateful=1
com_ibm_CSI_performTLClientAuthenticationRequired=0
com_ibm_CSI_performTLClientAuthenticationSupported=0
com_ibm_CSI_performTransportAssocSSLTLSRequired=0
com_ibm_CSI_performTransportAssocSSLTLSSupported=1
com_ibm_CSI_perform_ssl_sys_v3_timeout=600
com_ibm_HTTP_claimClientAuthentication=0
com_ibm_HTTP_claimKeyringName=WASKeyring
com_ibm_HTTP_claimSecurityLevel=HIGH
com_ibm_HTTP_claim_sslEnabled=1
com_ibm_HTTP_claims_ssl_sys_v3_timeout=600
com_ibm_Server_Security_Enabled=1
com_ibm_authMechanisms_type_OID=oid:1.3.18.0.2.30.2
com_ibm_security_SAF_EJBROLE_Audit_Messages_Suppress=0
com_ibm_security_SAF_unauthenticated=WSGUEST
com_ibm_userRegistries_type=security:LocalOSUserRegistry
control_region_classpath=/WebSphereTV/V5R1M0/BS01/AppServer/properties:/WebSphereTV/V5R1M0/BS01
/AppServer/lib/bootstrap.jar:/WebSphereTV/V5R1M0/BS01/AppServer/lib/j2ee.jar:/WebSphereTV/V5R1M
0/BS01/AppServer/lib/lmproxy.jar
control_region_jvm_properties_file=/WebSphereTV/V5R1M0/BS01/AppServer/config/cells/cel521/nodes
/nd521sc52/servers/ws521sc52/control.jvm.options
control_region_libpath=/usr/lpp/java/J1.4/bin:/usr/lpp/java/J1.4/bin/classic:/WebSphereTV/V5R1M
0/BS01/AppServer/lib
control_region_security_enable_trusted_applications=1
control_region_thread_pool_size=25
control_region_wlm_dispatch_timeout=300
daemonInstanceName=WS521D
daemonName=CL521
daemon_group_name=CL521
daemon_start_command=START WS5521D
daemon_start_command_args=JOBNAME=WS521D,ENV=CL521.CL521.WS521D
daemon_was_env_file=/WebSphereTV/V5R1M0/BS01/Daemon/config/cells/cel521/CL521/WS521D/was.env
node_name=nd521sc52
node_short_name=ND521
nonauthenticated_clients_allowed=1
primordial_root=/WebSphereTV/V5R1M0/BS01
private_bboo_jvm_in_ctl=1
protocol_http_defaultIdentity=WSGUEST
protocol_http_enableSSLTracking=0
protocol_http_listenIPAddress=*
protocol_http_port=19080
protocol_http_session_persistent=0
protocol_http_transport_network_qos=NONE
protocol_https_default_identity=WSGUEST
protocol_https_listenIPAddress=*
protocol_https_port=19443
protocol_iiop_daemon_listenIPAddress=wtsc52.itso.ibm.com
protocol_iiop_daemon_port=15655
```

```
protocol_iiop_daemon_port_ssl=15656
protocol_iiop_listenIPAddress=*
protocol_iiop_local_timeout=1800
protocol_iiop_no_local_copies=0
protocol_iiop_port=12809
protocol_iiop_port_ssl=0
protocol_iiop_server_session_keepalive=0
protocol_iiop_server_session_keepalive_ssl=0
ras_trace_BufferCount=4
ras_trace_BufferSize=128K
ras_trace_ctraceParms=50
ras_trace_defaultTracingLevel=1
ras_trace_outputLocation=SYSPRINT BUFFER
read_license_agreement=1
security.enablePluggableAuthentication=1
security_EnableRunAsIdentity=0
security_EnableSyncToOSThread=0
security_assertedID_IBM_accepted=0
security_assertedID_IBM_sent=0
security_kerberos_allowed=0
security_local_identity=WSGUEST
security_remote_identity=WSGUEST
security_sslClientCerts_allowed=0
security_sslType1=0
security_userid_passticket_allowed=1
security_userid_password_allowed=0
security_zOS_domainType=0
security_zSAS_ssl_repertoire=nd521sc52/DefaultIIOPSSL
server_configured_system_name=SC52
server_generic_short_name=CLU521
server_generic_uuid=BBD395FAF0FC66070000100C00000001090C042A
server_region_classpath=/WebSphereTV/V5R1M0/BS01/AppServer/properties:/WebSphereTV/V5R1M0/BS01/
AppServer/lib/bootstrap.jar:/WebSphereTV/V5R1M0/BS01/AppServer/lib/urlprotocols.jar:/WebSphereT
V/V5R1M0/BS01/AppServer/lib/j2ee.jar:/WebSphereTV/V5R1M0/BS01/AppServer/lib/lmproxy.jar
server_region_dynapplenv_jclparms=JOBNAME=&IWMSSNM.S,ENV=CL521.ND521.&IWMSSNM
server_region_dynapplenv_jclproc=WS5521S
server_region_jvm_properties_file=/WebSphereTV/V5R1M0/BS01/AppServer/config/cells/cel521/nodes/
nd521sc52/servers/ws521sc52/servant.jvm.options
server_region_libpath=/usr/lpp/java/J1.4/bin:/usr/lpp/java/J1.4/bin/classic:/WebSphereTV/V5R1M0
/BS01/AppServer/lib
server_region_workload_profile=IOBOUND
server_specific_name=ws521sc52
server_specific_short_name=WS521
server_specific_uuid=BBD395FAEFB32B870000100C00000001090C042A
server_type=AppServer
shell_command_proc_name=BBOW5SH
topology_server_BBD395FAEFB32B870000100C00000001090C042A=1
topology_server_servernamefromuuidBBD395FAEFB32B870000100C00000001090C042A=WS521
transaction_defaultTimeout=120
```

```
transaction_maximumTimeout=300
was_env_file=/WebSphereTV/V5R1M0/BS01/AppServer/config/cells/cel521/nodes/nd521sc52/servers/ws5
21sc52/was.env
wlm_dynapplenv_single_server=1
wlm_maximumSRCount=1
wlm_minimumSRCount=1
```

### *control.jvm.options*

Located deep in the HFS, the control.jvm.options file contains parameters that are passed to the Control Region JVM.

*Example: A-6 The control.jvm.options file*

```
/WebSphereTV/V5R1M0/BS01/AppServer/config/cells/cel521/nodes/nd521sc52/servers/ws521sc52/contro
l.jvm.options

-Dws390.nodename=nd521sc52
-Dwbisf.smpe.install.root=/SC52/zWBISFTV
-Djava.security.policy=/WebSphereTV/V5R1M0/BS01/AppServer/properties/server.policy
-Djava.security.auth.login.config=/WebSphereTV/V5R1M0/BS01/AppServer/properties/wsjaas.conf
-Dwas.repository.root=/WebSphereTV/V5R1M0/BS01/AppServer/config
-Dfile.encoding=ISO8859-1
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
-Dws.ext.dirs=/usr/lpp/java/J1.4/lib:/WebSphereTV/V5R1M0/BS01/AppServer/classes:/WebSphereTV/V5
R1M0/BS01/AppServer/lib:/WebSphereTV/V5R1M0/BS01/AppServer/lib/ext:/WebSphereTV/V5R1M0/BS01/App
Server:/WebSphereTV/V5R1M0/BS01/AppServer/web/help:/WebSphereTV/V5R1M0/BS01/AppServer/deploytoo
l/itp/plugins/com.ibm.etools.ejbdeploy/runtime
-DtraceSettingsFile=config/cells/cel521/nodes/nd521sc52/servers/ws521sc52/trace.dat
-Dserver.root=/WebSphereTV/V5R1M0/BS01/AppServer
-Dcom.ibm.itp.location=/WebSphereTV/V5R1M0/BS01/AppServer/bin
-Dwas.install.root=/WebSphereTV/V5R1M0/BS01/AppServer
-Dwas.history.dir=/tmp
-Xms48m
-Dsecurity.use.localos.authorization=true
-Djava.awt.headless=true
-Djava.ext.dirs=/usr/lpp/java/J1.4/lib/ext:/WebSphereTV/V5R1M0/BS01/AppServer/java/jre/lib/ext
-Xmx128m
-Djava.home=/usr/lpp/java/J1.4
-Duser.install.root=/WebSphereTV/V5R1M0/BS01/AppServer
-Dsmpe.install.root=/usr/lpp/zWebSphereTV/V5R1M0
```

### *servant.jvm.options*

This servant.jvm.options file contains the parameters passed to the Servant Region JVM at startup.

*Example: A-7 The servant.jvm.options*

```
/WebSphereTV/V5R1M0/BS01/AppServer/config/cells/cel521/nodes/nd521sc52/servers/ws521sc52/servan
t.jvm.options

-Xmx256m
-Dws390.nodename=nd521sc52
-Dwbisf.smpe.install.root=/SC52/zWBISFTV
-Djava.security.policy=/WebSphereTV/V5R1M0/BS01/AppServer/properties/server.policy
-Djava.security.auth.login.config=/WebSphereTV/V5R1M0/BS01/AppServer/properties/wsjaas.conf
-Dwas.repository.root=/WebSphereTV/V5R1M0/BS01/AppServer/config
-Dfile.encoding=ISO8859-1
-Duser.dir=/WebSphereTV/V5R1M0/BS01/AppServer
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
-Dws.ext.dirs=/usr/lpp/java/J1.4/lib:/WebSphereTV/V5R1M0/BS01/AppServer/classes:/WebSphereTV/V5
R1M0/BS01/AppServer/lib:/WebSphereTV/V5R1M0/BS01/AppServer/lib/ext:/WebSphereTV/V5R1M0/BS01/App
Server:/WebSphereTV/V5R1M0/BS01/AppServer/web/help:/WebSphereTV/V5R1M0/BS01/AppServer/deploytoo
l/itp/plugins/com.ibm.etools.ejbdeploy/runtime
-DtraceSettingsFile=config/cells/cel521/nodes/nd521sc52/servers/ws521sc52/trace.dat
-Dserver.root=/WebSphereTV/V5R1M0/BS01/AppServer
-Dcom.ibm.CORBA.iiop.noLocalCopies=false
-DbrbPropertiesFile=/WebSphereTV/V5R1M0/BS01/AppServer/bin/brbeansDefaultProperties
-Dcom.ibm.itp.location=/WebSphereTV/V5R1M0/BS01/AppServer/bin
-Dwas.install.root=/WebSphereTV/V5R1M0/BS01/AppServer
-Dwas.history.dir=/tmp
-Dsecurity.use.localos.authorization=true
-Djava.awt.headless=true
-Djava.ext.dirs=/usr/lpp/java/J1.4/lib/ext:/WebSphereTV/V5R1M0/BS01/AppServer/java/jre/lib/ext
-Xms256m
-Djava.home=/usr/lpp/java/J1.4
-Duser.install.root=/WebSphereTV/V5R1M0/BS01/AppServer
-Dsmpe.install.root=/usr/lpp/zWebSphereTV/V5R1M0
```

### *resources.xml*

The following resources.xml files are located in the HFS under the /WebSphereTV/V5R1M0/BS01/AppServer directory:

- resources.xml
- config/cells/cel521/resources.xml
- config/cells/cel521/nodes/nd521sc52/resources.xml
- config/cells/cel521/nodes/nd521sc52/servers/ws521sc52/resources.xml
- config/templates/default/nodes/servers/server1/resources.xml
- config/templates/system/nodes/servers/jmsserver/resources.xml

## DB2

We used the DB2 subsystem at version 8. It is a datasharing environment under the group DB2 subsystem ID D8IG. Table A-5 specifies the DB- related settings for the two test systems in a sysplex.

*Table A-5 The DB2 environment (sysplex)*

| | **SC48** | **SC52** |
|---|---|---|
| Subsystem ID | D8I1 | D8I3 |
| JDBC HFS | /usr/lpp/db2/db2810 | /usr/lpp/db2/db2810 |
| Location ID | DB8I | DB8I |
| TCPPort | 38100 | 38100 |
| RESPort | 38101 | 38103 |

## CICS

The CICS region name CICSPAME was created on SC66 only. This region was CICS Transaction Server 2.2. This single CICS region was also used for the monoplex configuration.

## CICS Transaction Gateway

For details about the CICS Transaction Gateway, this configuration for WBISF involved the same details as explained in “CICS Transaction Gateway” on page 282.

## WebSphere MQ

WebSphere MQ ran on SC48 and SC52 with the configuration specified in Table A-6. These WebSphere MQ subsystems are capable of Queue Sharing. However, this feature is not used for the WBISF examples in this book.

*Table A-6 WebSphere MQ configuration*

| | **SC48** | **SC52** |
|---|---|---|
| Subsystem names | MQ8B | MQFI |
| Listener ports | 1414 | 1562 |
| WMQ Java HFS | /usr/lpp/mqm/V5R3M1 | /usr/lpp/mqm/V5R3M1 |

B

# Running graphics applications on z/OS

Within the UNIX System Services environment on z/OS, programs can use the X Window System to exploit graphics capabilities, for example, to provide a graphical user interface (GUI) to an application. This appendix explains the steps to enable z/OS and the user workstation to exploit such graphical features. For this solution, we installed open source software on the Microsoft Windows workstation.

## B.1 X-server and z/OS UNIX System Services

As a UNIX environment, UNIX System Services on z/OS can use the X Window System, which is the de facto standard graphics engine for UNIX systems. X11 is often used to refer to Version 11 of the X Window System.

Applications, such as those on z/OS, can be clients for an X Window System on a workstation. This can either be a hardware device known as an X-terminal, which has the X-server software built into it. Or, this can be an X-server emulation on a workstation platform that supports it. Examples are Linux and Microsoft Windows workstations. Through this client/server technology, UNIX applications can use the graphical interfaces.

**Or is it server/client?** While the workstation is usually a client for the applications on the UNIX server, the UNIX server itself becomes a client for the windowing and GUI capabilities on the workstation.

## B.2 Setting up the workstation as an X-server

Our workstations run Windows 2000 and Windows XP. We installed the X11 emulation software that is available in the Cygwin Linux emulator for Windows.

**Cygwin:** Cygwin is a trademark of Red Hat, Inc. The software is freely available only to those who are willing to accept it under the GPL+exception license. Those who want to include it in a binary only application must obtain a proprietary license from Red Hat.

For a free download of Cygwin, go to the Web site:

http://www.cygwin.com

You can find licensing for this Red Hat software on the Web at:

http://cygwin.com/licensing.html

Cygwin is a Linux emulation package for the Windows platform. It is available both in the binary and source format. For the purpose of our book, we used Cygwin X-server enablement. This enabled our z/OS applications to be clients for the windowing GUI facilities on our workstation.

From the Cygwin Web site indicated above, you can find a link to download the setup.exe file for Cygwin. Running this setup program gives you the option to download and install additional Cygwin features. The following steps explain how to install Cygwin.

1. Figure B-1 shows the initial window of the setup program. Click **Next**.

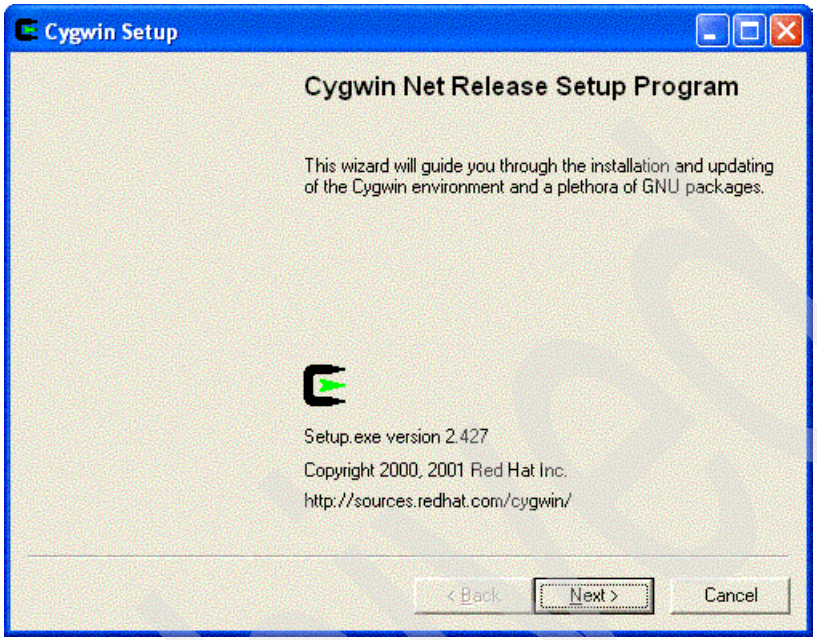



*Figure B-1 Cygwin installation options*

2. From the Choose A Download Source panel (Figure B-2), select the **Download from Internet** option to save the installation packages to your local system. If you have no intention of replicating the installation on another workstation, you may choose to install directly from the Internet. Click **Next**.

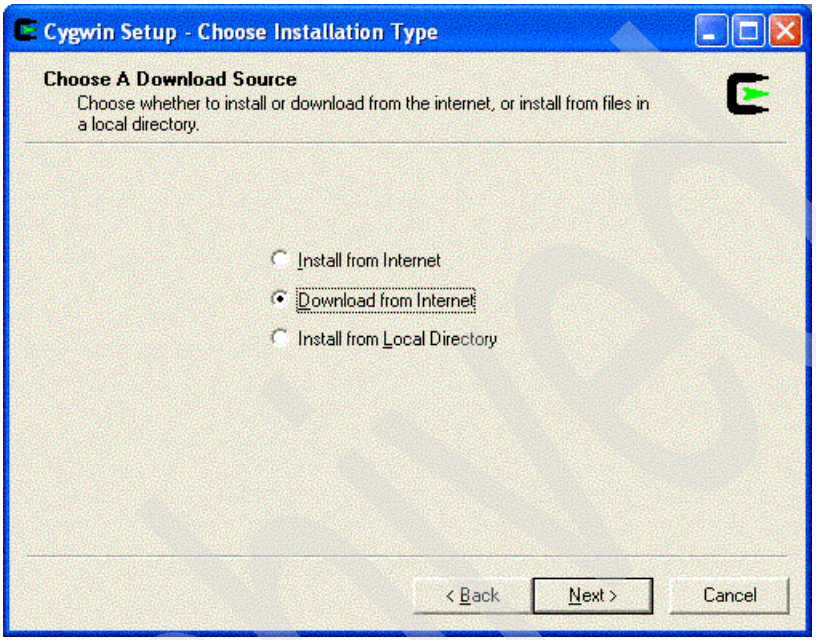



*Figure B-2 Cygwin installation options*

3. On the Select Local Package Directory panel (Figure B-3), specify a directory to save the installation packages, and click **Next**.

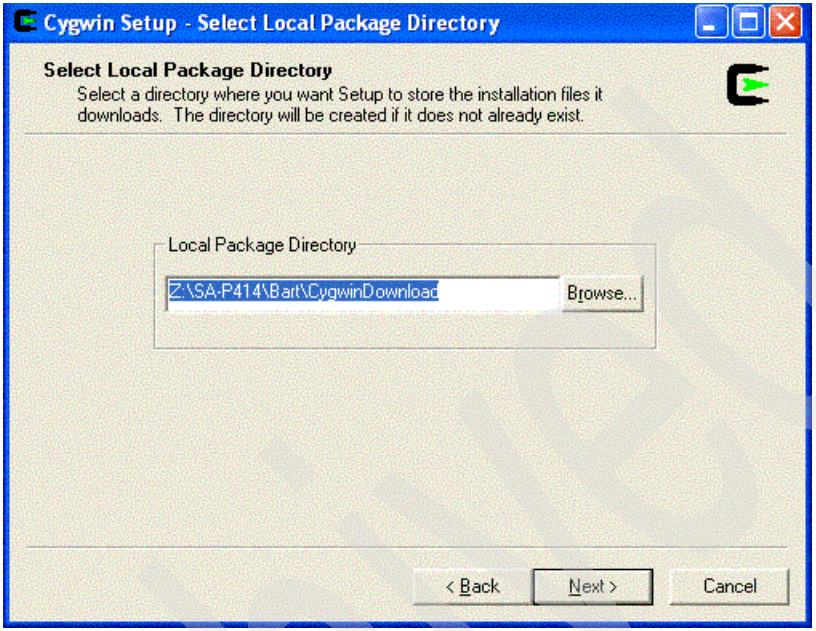



*Figure B-3 Cygwin download directory*

4. In the Select Your Internet Connection window (Figure B-4), choose the network connections that are appropriate for you, or accept the default option, and click **Next**.

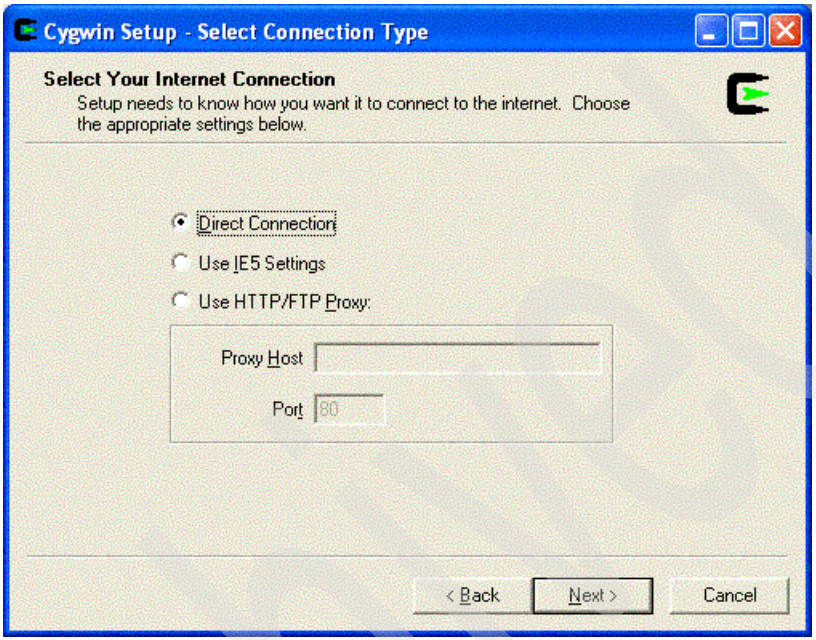



*Figure B-4 Cygwin network connection options*

5. In the Choose A Download Site (Figure B-5), from the available download servers, choose one that is close to your location, and click **Next**.

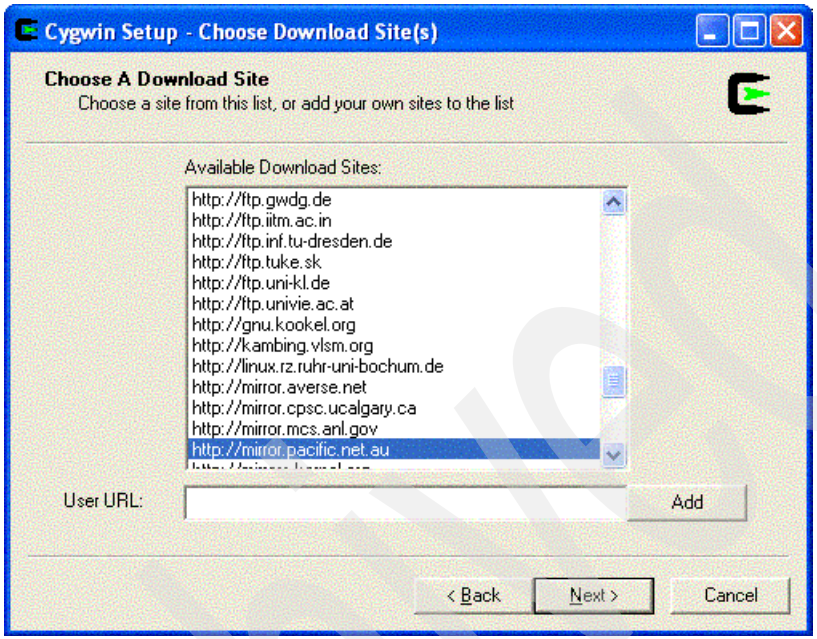



*Figure B-5 Cygwin download servers*

6. After a few seconds, the Progress monitor window for the download (Figure B-6) disappears.

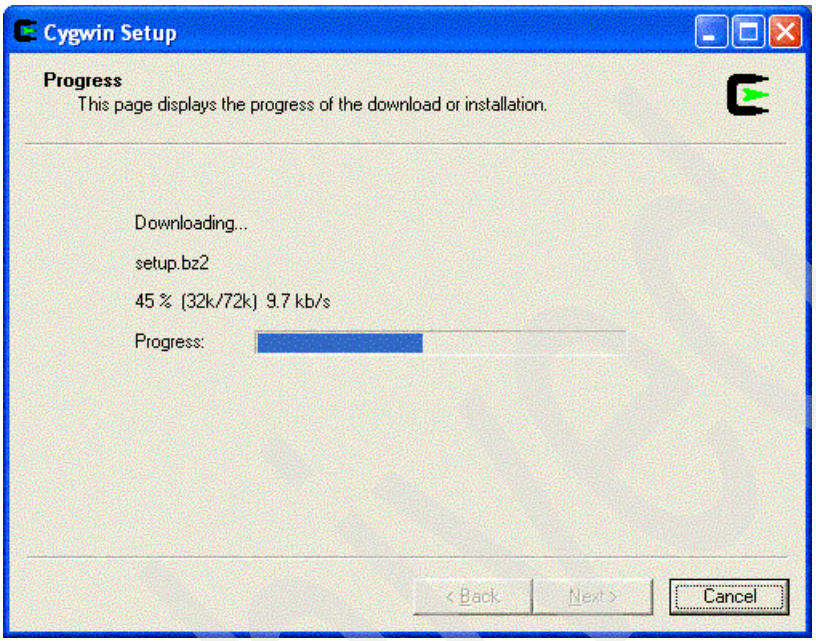



*Figure B-6 Cygwin connecting to download server*

7. In the Select Packages window (Figure B-7), select the packages that you require for your purpose. We have chosen to select all packages. Click the circular clockwise arrow icon in the list of packages to toggle between the selection options for that package. Make sure that the packages you require are set to **Install** as highlighted in Figure B-7. Then click **Next**.

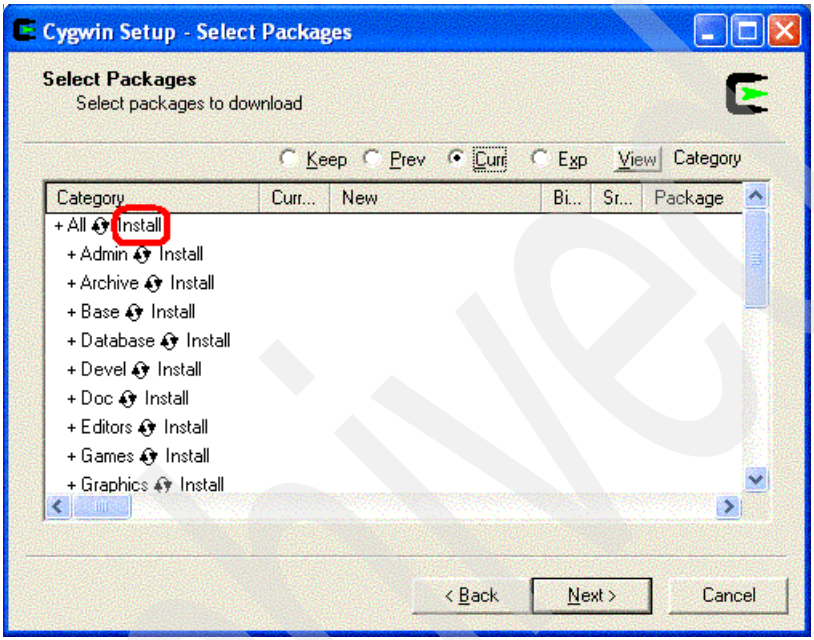



*Figure B-7 Cygwin package selection for download*

8. Notice that the download of all packages transfers almost 400 MB to your workstation. Make sure that you have this space available and that your network connection has sufficient capacity. The download process takes several minutes. Figure B-8 shows an example of the download progress window that you see.

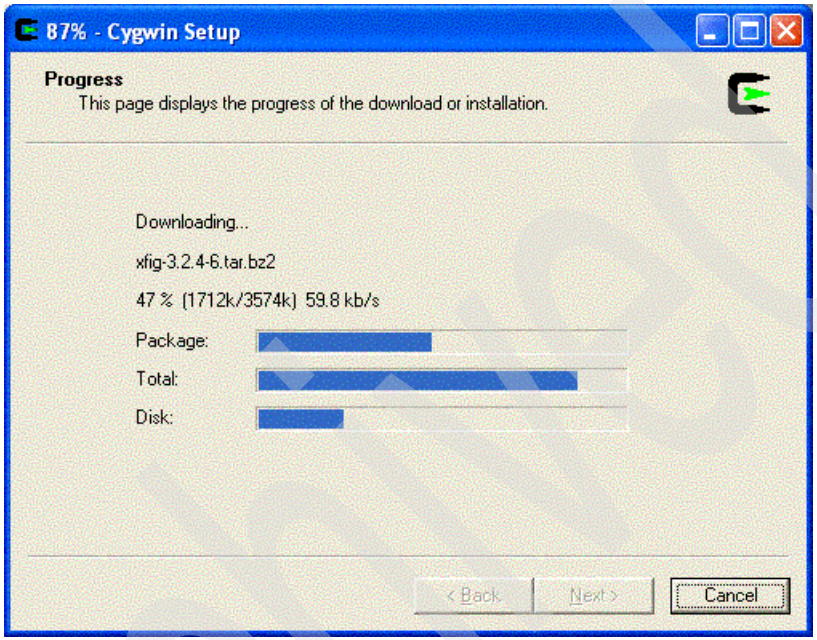



*Figure B-8 Cygwin package download progress*

9. After the download completes, restart the Cygwin setup.exe program.
10. This time, in the Choose A Download Source panel (Figure B-9), choose the **Install from Local Directory** option and click **Next**.

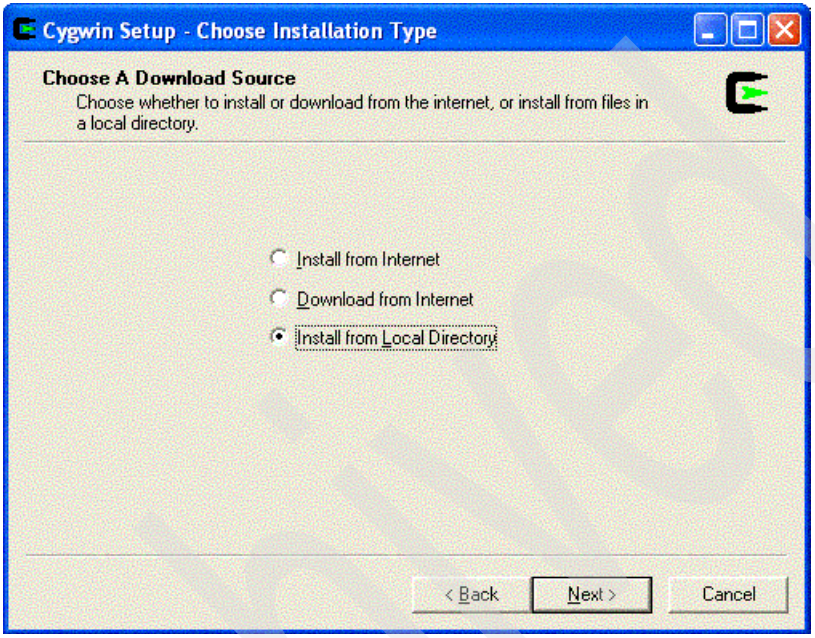



*Figure B-9 Cygwin package installation after download*

11. The previous download steps saved the package installation file to your workstation. These are now expanded in the installation directory that you specify under Root Directory (see Figure B-10). If you have multiple users defined on your workstation, under Install For, you may want to choose the **All Users** option.

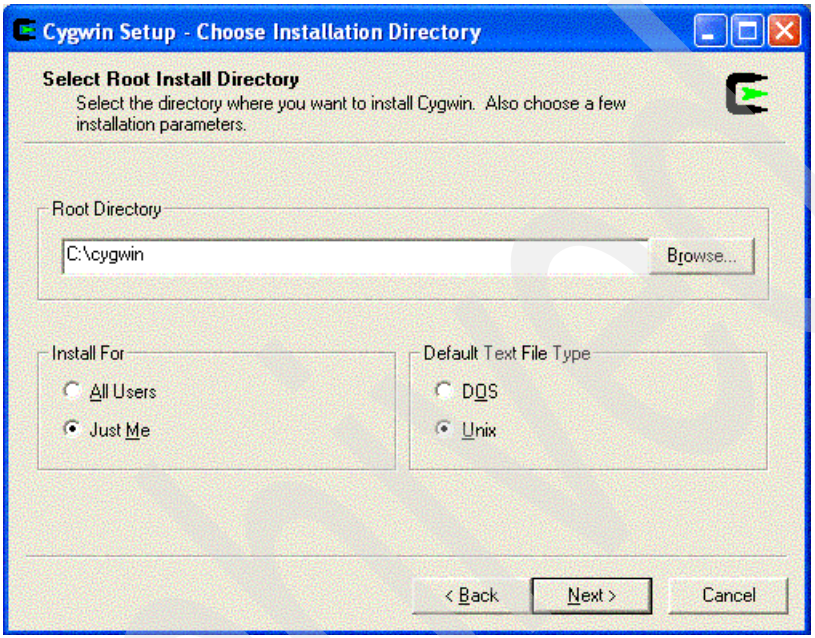



*Figure B-10 Cygwin installation directory path*

12. In the Select a Local Package Directory window (Figure B-11), specify the directory path into which you previously downloaded the Cygwin packages. Click **Next**.

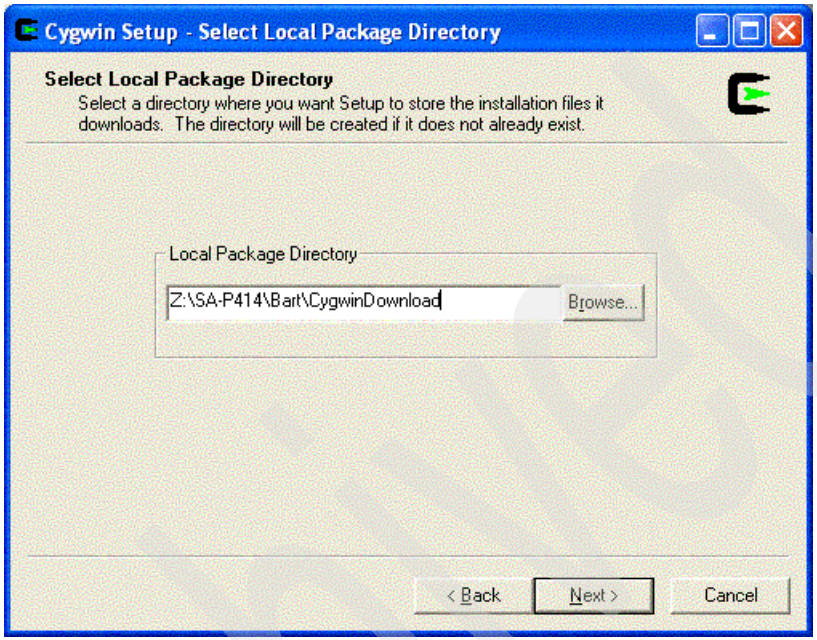



*Figure B-11 Cygwin installation from downloaded packages*

You now see a window that is similar to the one shown in Figure B-7 on page 301. Again, make sure that you set the appropriate package selection option. The next window shows you the progress of the installation, which may take several minutes to complete. After it has completed, you are given the option to create Cygwin icons on your workstation desktop of Start menu. This completes the installation.

## B.3 Starting the X-server on the workstation

After you install Cygwin, you should see the startxwin.bat file in your installation directory. For our installation, the full path is shown in Figure B-12.

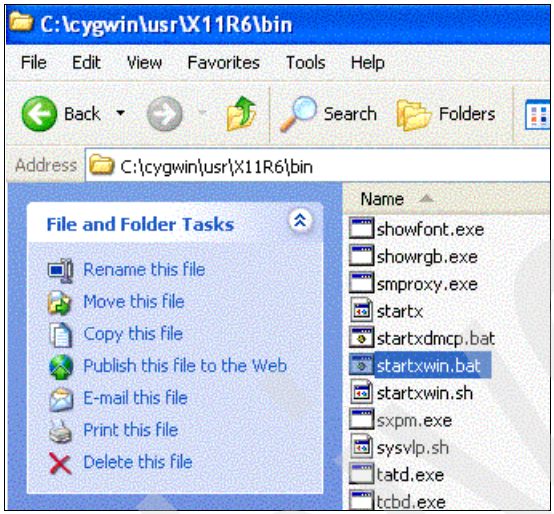



*Figure B-12 Cygwin startup script*

Start the startxwin.bat program on your workstation. It may be a good idea to add a link to it on your Desktop or Taskbar. As a result, you see an X-server icon appear in the Taskbar of your workstation, as shown in Figure B-13. This indicates that the X-server is started.


*Figure B-13 Cygwin X-server icon*

After a while, an X-terminal window opens (see Figure B-14). You have to specify the graphics clients that can connect to your X-server. To do this, you have to use the **xhost** command. As shown in Figure B-14, the **xhost** command has the following syntax:

```
xhost +9.12.4.30
```

Here, 9.12.4.30 is the IP address of the z/OS system on which our GUI application will be started. From now on, these applications can be successfully started, as shown in the following section.

You can exit from the X-terminal window using the **exit** command as shown in Figure B-14.

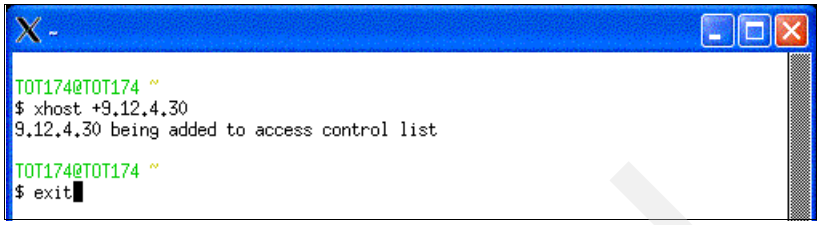



*Figure B-14 Cygwin X-terminal setup*

## B.4 Setting up the z/OS application as an X-client

We use the CloudView application as an example for our demonstration. CloudView is a graphical user interface to the Cloudscape relational database that is freely available from WebSphere Application Server Version 5 and later, and on the z/OS platform.

We set up the UNIX shell script **cview.sh** shown in Figure B-15 to start the CloudView interface.

```
#!/bin/ksh
#
export JAVA_HOME=/usr/lpp/java/J1.4
export CLSCAPE=/usr/lpp/zWebSphere/V5R1M0/cloudscape
#-------------------------------------------------------
java -Djava.ext.dirs=$CLSCAPE/lib -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.cview
```

*Figure B-15 CloudView startup script cview.sh*

However, executing this script returns the error message shown in Figure B-16, which can be found in the SysVisual.LOG application logfile.

```
java.lang.InternalError: Can't connect to X11 window server using ':0.0' as
the value of the DISPLAY variable.
```

*Figure B-16 CloudView X11 failure*

The application is trying to connect to an X-server workstation, but the location of the X-server is not properly set. It requires a DISPLAY variable to be set with the correct host name, with a suffix of “:0.0”, which is the default X-server and X Window System on that workstation host. If you have additional X Window System started on the same workstation, you may have to use the suffix “:0.1”, for example, to direct the graphics of the application to the second window.

We set the DISPLAY variable in the **cview.sh** script as shown in Figure B-17. Notice that its value takes the first parameter specified on the **cview.sh** command as the host name for the X-server workstation. For example, the following command runs the CloudView GUI on the workstation at network location 9.12.10.4:

```
cview.sh 9.12.10.4
```

```
#!/bin/ksh
#-------------------------------------------------------
# Call syntax:
#
#   cview.sh HOST_NAME
#
# where HOST_NAME should be replaced by a workstation's hostname or
# ip-address.  The window-based GUI will be replaced on that workstation
#
# The workstation should run an X-server since the cview Java program
# will use the X11 protocol to communicate the GUI emulation.
#
# We have used the Cygwin package for Windows, which is available from
# http://cygwin.com/licensing.html
#
#-------------------------------------------------------
#
export JAVA_HOME=/usr/lpp/java/J1.4
export CLSCAPE=/usr/lpp/zWebSphere/V5R1M0/cloudscape
export DISPLAY=$1:0.0
#-------------------------------------------------------
java -Djava.ext.dirs=$CLSCAPE/lib -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.cview
```

*Figure B-17 CloudView script with DISPLAY variable set*

As a result, the GUI for CloudView opens on the workstation (see Figure B-18).

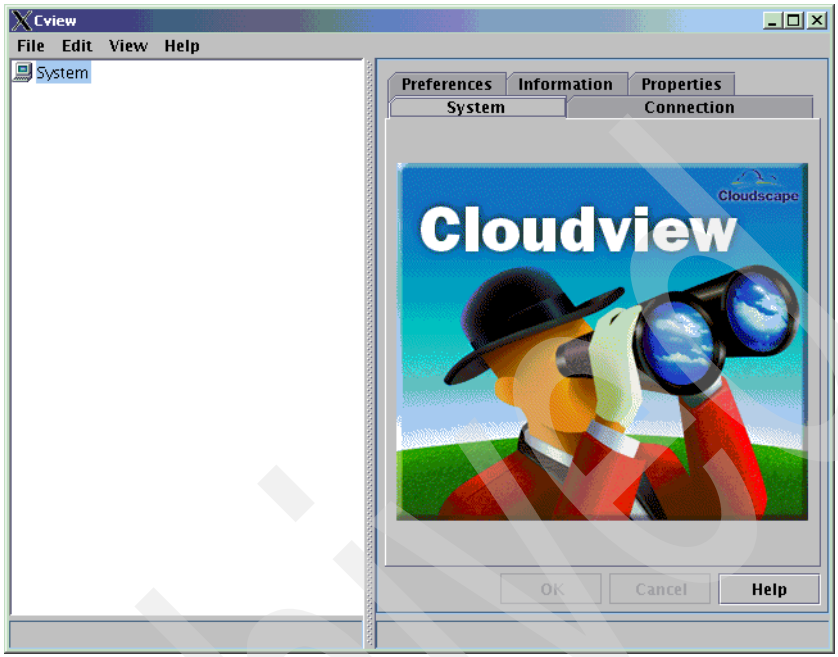



*Figure B-18 CloudView GUI*

## B.5 Stopping the X-server on the workstation

As long as your Microsoft Windows workstation has the X-server icon in the Taskbar, as shown in Figure B-13, the X-server is still active and listening for work. To stop the X-server, double-click the icon.

The Exit window (Figure B-19) opens. If you have no more connected clients, you can safely exit the X-server.

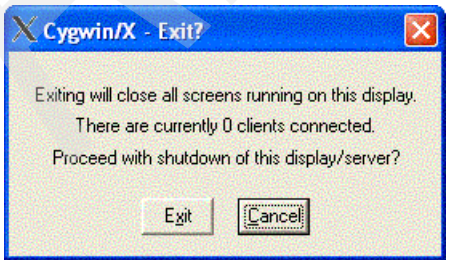



*Figure B-19 Cygwin exit X-server*

C

# Creating Cloudscape databases

Cloudscape is an object-relational, pure-Java database system. It is shipped as a component of WebSphere Application Server. It is supported for the development and testing of WebSphere applications. IBM does not currently support it as a production database system.

In this appendix, we explain how we use Cloudscape on z/OS for some of the test scenarios.

## C.1 Cloudscape background

The Cloudscape database system is implemented in the Java language. It comes with the WebSphere Application Server package as several Java archive (JAR) files (db2j.jar, db2jcc.jar, db2jcc_license_c.jar, db2jtools.jar, db2jcview.jar, and db2jnet.jar).

The Cloudscape license that comes with WebSphere Business Integration Server Foundation (WBISF) is only for development and test, and not for production purposes. There are two variations of the product:

- The embedded Cloudscape, which is installed automatically with WebSphere
- The Cloudscape Network Server that allows client/server JDBC access over Distributed Relational Database Architecture™ (DRDA®) protocol

The embedded Cloudscape only provides access to the database from a single Java Virtual Machine. This effectively limits its scope of operation to a Base Application Server node.

The Network Server enables remote applications, that is JVMs, to connect to the Cloudscape database on another system, which can be useful in a clustered or sysplex environment. However, the Cloudscape Network Server does has not include XA transactional support, which makes it unsuitable for most production applications.

## C.2 Our system setup

For the purpose of this book, we create the folder structure shown in Figure C-1 for our Cloudscape databases and utility programs.

```
 Directory List

Select one or more files with / or action codes.
EUID=3133   /sg246356/Cloudscape/
  Type  Perm  Filename
_ Dir    775  .
_ Dir    755  ..
_ Syml   777  databases
_ Dir    775  DDL
_ Dir    775  jobs
_ Dir    775  logs
_ Dir    775  scripts
```

*Figure C-1 Cloudscape utility file structure*

The scripts folder contains the script files to run the Cloudscape Data Definition Language (DDL), while the DDL files themselves are saved in the DDL folder. We use the logs folder to save the output messages from our scripts. In the jobs directory, we keep the job control language (JCL) files to run the scripts in batch. The databases are created in the databases folder.

Cloudscape provides a utility program, called **ij**, which can be used to execute its DDL commands. We create the **executeDDL.sh** UNIX shell script shown in Figure C-2 to run the DDL scripts more easily.

```
 BROWSE -- /sg246356/Cloudscape/scripts/executeDDL.sh
 Command ===>                                                    Scroll ===> CSR
********************************** Top of Data ************************************
#!/bin/ksh
#
export PATH=/usr/lpp/java/J1.4/bin/:$PATH
export JAVA_HOME=/usr/lpp/java/J1.4
export CLOUD_HOME=/usr/lpp/zWebSphere/V5R1M0/cloudscape
#-------------------------------------------------------
java -Djava.ext.dirs=$CLOUD_HOME/lib -Dij.protocol=jdbc:db2j:com.ibm.db2j.tools.ij $1
```

*Figure C-2 Script to run Cloudscape DDL*

We create an additional DDL for all our Cloudscape changes. These will all use the **executeDDL.sh** script.

## C.3 Creating the BPEDB database in Cloudscape

Because Cloudscape Network Server has no XA support, Process Choreographer can only use the embedded Cloudscape version that cannot be accessed remotely. That is why Cloudscape cannot be used as a database system for Process Choreographer in a clustered environment.

This is documented in the product manual *WebSphere Business Integration Server Foundation: Applications,* SA22-7978.

## C.3.1 Creating the database

Create a Cloudscape database named `BPEDB` by using the following steps:

1. Make sure that you have administrator rights on the machine where your application server is installed.
2. To use a database location other than the default location, perform either of the following actions:
   - Change to the directory where you want the new database created
   - Edit the createDatabaseCloudscape.ddl script located in the ProcessChoreographer subdirectory of the application server installation directory, and add the fully qualified path to the database name.

   For example, locate the following line:

   ```
   connect "BPEDB;create=true" as BPEDB;
   ```

   Change the line to:

   ```
   connect "install_root/ProcessChoreographer/BPEDB;create=true" as BPEDB
   ```
3. If you do not use the default location, *install_root*/ProcessChoreographer/BPEDB, and you plan to use the Business Process Container Install Wizard, remember to also change the custom properties field when selecting the database. You must change the value of the property databaseName to your fully qualified database location.
4. At the command prompt, enter the command to run the database creation script using the Cloudscape command line processor. Type the following command (on a single line):

   ```
   java -Djava.ext.dirs=install_root/cloudscape/lib
      -Dij.protocol=jdbc:db2j:
      com.ibm.db2j.tools.ij
      install_root/ProcessChoreographer/createDatabaseCloudscape.ddl
   ```

   **Note:** Be sure to run this command with a user ID that has authorization to create subfolders and files in the install_root/ProcessChoreographer directory. We used the WSADMIN user ID to run the command.

As a result, the Cloudscape database for Process Choreographer now exists. You can verify this with the CloudView utility as explained in C.3.3, "Verifying the result in the CloudView GUI" on page 318.

## C.3.2 Using scripts and JCL

Figure C-3 shows the script that we use to create the BPEDB database as required for the Business Process Container.

```
 BROWSE -- /sg246356/Cloudscape/scripts/createBPEDB.sh
 Command ===>                                                  Scroll
******************************** Top of Data **********************
#!/bin/ksh
#
# this script creates the database for the Business Process Container
# see the reference DDL file for additional information
#
executeDDL.sh ../DDL/createBPEDB.ddl>../logs/createBPEDB.log
```

*Figure C-3 Script to create the Business Process Engine database*

To run the script, we start a UNIX command window for OMVS, using the TSO OMVS command. Figure C-4 shows how to run the command for this scenario. Notice the **su** command to switch the user context to that of WSADMIN, the user ID that owns the install_root/ProcessChoreographer file system.

```
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2003
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

BARTV @ SC49:/sg246356>cd Cloudscape
BARTV @ SC49:/sg246356/Cloudscape>cd scripts
BARTV @ SC49:/sg246356/Cloudscape/scripts>su WSADMIN
FSUM5019 Enter the password for WSADMIN:
BARTV @ SC49:/sg246356/Cloudscape/scripts>createBPEDB.sh
BARTV @ SC49:/sg246356/Cloudscape/scripts>
```

*Figure C-4 Running the DDL script*

To run this script as a batch job, we create the JCL shown in Figure C-5.

```
//EXECDDL  JOB ACCNT#,USER=WSADMIN,PASSWORD=
//* ------------------------------------------------------------------
//EXECSH   EXEC PGM=BPXBATCH,
//            PARM='SH cd /sg246356/Cloudscape/scripts',
//            REGION=0M
//STDIN    DD PATH='/sg246356/Cloudscape/scripts/createBPEDB.sh',
//            PATHOPTS=(ORDONLY)
//STDOUT   DD PATH='/tmp/execddl.log',
//            PATHOPTS=(OWRONLY,OCREAT),PATHMODE=(SIRWXU,SIRGRP,SIROTH)
```

*Figure C-5 JCL to run DDL for Cloudscape*

Our script directs its output messages to a file in the logs directory. Figure C-6 shows an extract of this logfile.

```
ij version 5.1 (c) 2001,2003 IBM Corp.
connect '/SC49/WebSpherePB/V5R1M0/BS01/AppServer/ProcessChoreographer/BPEDB;create=true' as BPEDB;
ij> -- create the schema
CREATE TABLE SCHEMA_VERSION
(
    SCHEMA_VERSION                      INTEGER                             NOT NULL ,
    PRIMARY KEY ( SCHEMA_VERSION )
);
0 rows inserted/updated/deleted
ij> -- index for primary key to avoid deadlocks when running RR
CREATE INDEX SCHEMA_VERSIONP ON SCHEMA_VERSION
(
    SCHEMA_VERSION
);
0 rows inserted/updated/deleted
ij> CREATE TABLE PROCESS_TEMPLATE_B_T
(
    PTID                                BIT(128)                            NOT NULL ,
    NAME                                VARCHAR(220)                        NOT NULL ,
    DISPLAY_NAME                        VARCHAR(64)                                  ,
    APPLICATION_NAME                    VARCHAR(220)                                 ,

    ...

);
0 rows inserted/updated/deleted
ij> CREATE TABLE SCHED_LMPR
(
   LEASENAME                                VARCHAR(254) NOT NULL,
   NAME                                     VARCHAR(254) NOT NULL,
   VALUE                                    VARCHAR(254) NOT NULL
);
0 rows inserted/updated/deleted
ij> CREATE INDEX SCHED_LMPR_IDX1 ON SCHED_LMPR
(
   LEASENAME, NAME
);
0 rows inserted/updated/deleted
ij> -- end import scheduler DDL: createSchemaCloudscape.ddl


DISCONNECT ALL;
ij>
```

*Figure C-6 Execution log of the createBPED.sh script*

**Note:** The execution of the script in Figure C-3 creates a subfolder BPEDB in the *install_root*/ProcessChoreographer directory, if it does not exist already. Consequently, the Cloudscape database would be created in the same HFS file system as the installed Process Choreographer files. We do not recommend this for a production environment. Instead, we recommend that you use the following steps:

1. Create the BPEDB directory, either on the default location or in a directory outside the install_root path.
2. Create a new HFS file and mount it on that new mount point.

These steps are also required if the Cloudscape database has to be shared between two sysplex members.

### C.3.3 Verifying the result in the CloudView GUI

Cloudscape ships a GUI, called CloudView, which can also be used on the z/OS system, provided your workstation has X-server capabilities. Such X11 server emulations are available on Linux workstations. For our Windows workstation, we use Cygwin, which we obtain from:

http://cygwin.com/licensing.html

Refer to Appendix B, “Running graphics applications on z/OS” on page 293, to set up Cygwin on z/OS.

**Note:** Make sure that you run the **/sg246356/Cloudscape/scripts/cview.sh** script with sufficient authorization to access the database files that you will display and manipulate in the CloudView GUI.

We run the **cview.sh** script using the following command:

```
cview.sh 9.12.10.4
```

Here, 9.12.10.4 is the IP address of our workstation, on which the X-server has been started. As a result, the CloudView interface (Figure C-7) starts on the workstation.

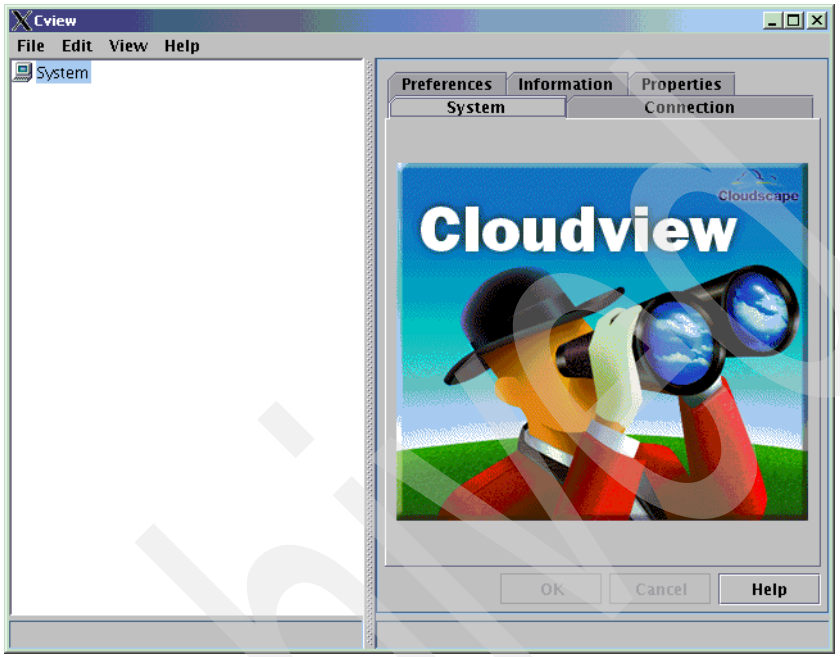



*Figure C-7 CloudView opening window*

Then we select **File** → **Open...** In the Browse for Folder window (Figure C-8), we locate the directory in which the BPEDB database was created and click **Open**.

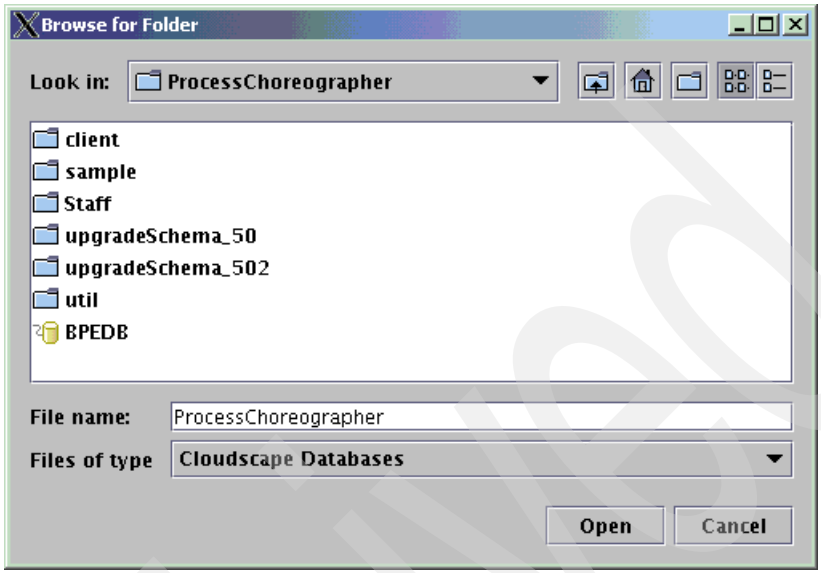



*Figure C-8 Browse for Folder window: Locating the database folder in CloudView*

After we open the BPEDB database in CloudView, we see a result similar to the one in Figure C-9. This indicates that the database has been created.

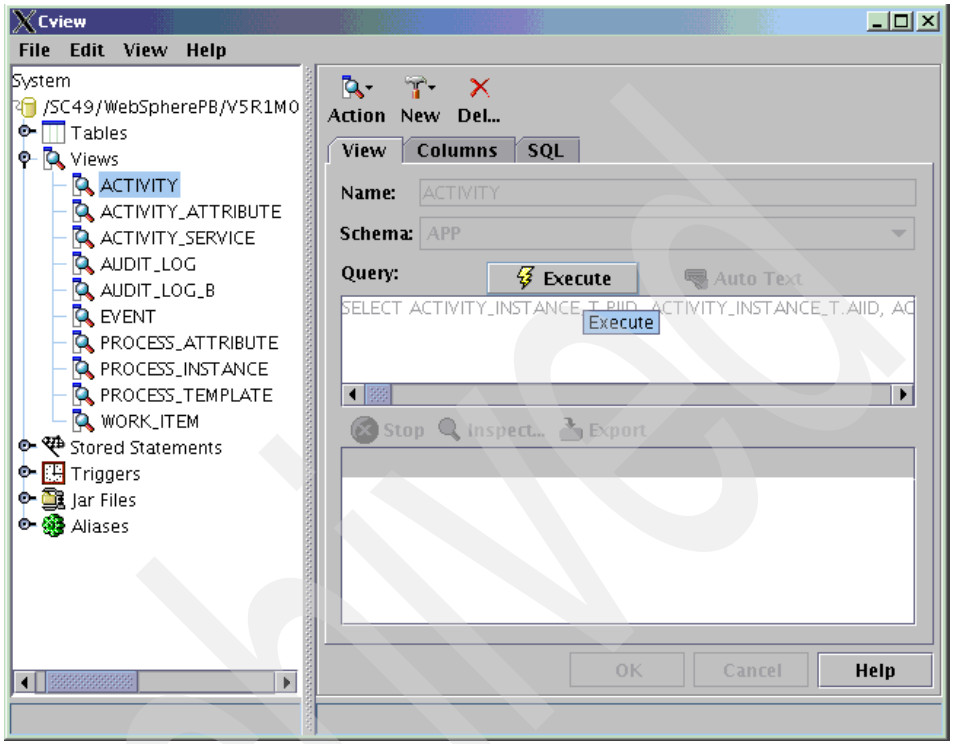



*Figure C-9 BPEDB database in CloudView*

## C.3.4 Cloudscape restriction

WebSphere-embedded Cloudscape does not support simultaneous access from two or more JVMs. For example, running CloudView and an application server instance may lead to the error message shown in Figure C-10.

```
Database access resulted in SQL exception: SQLState , Error code 40000,
Message: Failed to start database
'/SC49/WebSpherePB/V5R1M0/BS01/AppServer/ProcessChoreographer/BPEDB',
see the next exception for details.

SQLState , Error code 45000, Message:
WARNING: Cloudscape (instance [Ljava.lang.Object;@e74b281) is attempting to boot the
database {1} even though Cloudscape (instance {2}) may still be active.  Only one
instance of Cloudscape should boot a database at a time. Severe and non-recoverable
corruption can result if 2 instances of Cloudscape boot on the same database at the
same time.  The db2j.database.forceDatabaseLock=true property has been set, so the
database will not boot until the db.lck is no longer present.
Normally this file is removed when the first instance of Cloudscape to boot on the
database exits, but it may be left behind in some shutdowns.
It will be necessary to remove the file by hand in that case.
It is important to verify that no other VM is accessing the database before deleting
the db.lck file by hand.
```

*Figure C-10 Only a single JVM can access a Cloudscape database*

D

# Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

## D.1 Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246356

Alternatively, you can go to the IBM Redbooks Web site at:

**ibm.com**/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-6356.

## D.2 Using the Web material

The additional Web material that accompanies this redbook includes the following files:

| *File name* | *Description* |
|---|---|
| **SG246356.zip** | Zipped code samples |

The total file size is approximately 55 MB. There are three directories, each of which contain a version of the application processes used in this redbook.

- **Deployable Ears**: This directory contains the application code and descriptors that were deployed to the z/OS systems.
- **WSAD-IE**: This directory contains the BPEL, Java code, and packages needed to successfully import the business processes into the WebSphere Studio Application Developer Integration Edition (WSAD-IE) V5.1 workspace.
- **Sysplex version**: This directory contains the extended samples to load into WSAD-IE V5.1, along with the EAR files that can be deployed to z/OS Parallel Sysplex.

  Use the workspace supplied in the wksp_zOS_before directory to follow a complete a step-by-step guide to build the target solution. If you are interested in deploying only the resulting applications, then use the workspace from the wksp_zOS_after directory, or import the EAR files directly from the Deployable EAR subdirectory.

# Glossary

**address space** A range of virtual storage pages identified by a number (ASID) and a collection of segment and page tables which map the virtual pages to real pages of the computer's memory.

**address space connection** The result of connecting an allied address space to DB2. Each address space containing a task connected to DB2 has exactly one address space connection, even though more than one task control block (TCB) can be present. See *allied address space* and *task control block*.

**Advanced Program-to-Program Communication (APPC)** (1) The general facility characterizing the LU6.2 architecture and its implementation in different Systems Network Architecture (SNA) products. (2) Sometimes used to refer to an LU6.2 product feature in particular, such as an APPC application programming interface (API).

**allied address space** An area of storage, external to DB2, that is connected to DB2 and is capable of requesting DB2 services.

**American National Standards Institute (ANSI)** An organization consisting of producers, consumers, and general interest groups that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

**American Standard Code for Information Interchange (ASCII)** A standard assignment of 7-bit numeric codes to characters. See also *Unicode*. (2) An encoding scheme used to represent strings in many environments, typically on PCs and workstations. Contrast with *EBCDIC*.

**ANSI** See *American National Standards Institute*.

**AOR** See *application-owning region*.

**APAR** See *authorized program analysis report*.

**API** See *application program Interface.*

**APPC** See *Advanced Program-to-Program Communication*.

**applet** See *Java Applet*.

**application** (1) A program or set of programs that perform a task, such as a payroll application. (2) In Java programming, a self-contained, stand-alone Java program that includes a static main method. It does not require an applet viewer. Contrast with *applet*.

**application plan** The control structure produced during the bind process and used by DB2 to process SQL statements encountered during statement execution.

**application program interface (API)** A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or licensed program.

**application requester (AR)** See *requester*.

**application service provider (ASP)** An agent or broker that aggregates, facilitates, and brokers IT services to deliver IT-enabled business solutions across a network via subscription-based pricing.

**application-owning region (AOR)** A CICS region in an MRO environment that "owns" the CICS applications, and invokes them on behalf of remotely attached terminal (or Web) users. See also *TOR* and *listener region.*

**AR** Application requester. See *requester*.

**ASCII** See *American Standard Code for Information Interchange*.

**ASP** See *application service provider*.

**attribute** In XML, a name="value" pair that can be placed in the start tag of an element. The value must be quoted with single or double quotation marks.

**authorization ID** A string that can be verified for connection to DB2 and to which a set of privileges is allowed. It can represent an individual, an organizational group, or a function, but DB2 does not determine this representation.

**authorized program analysis report (APAR)** A report of a problem caused by a suspected defect in a current, unaltered release of a program.

**automatic bind** More correctly worded as automatic rebind. A process by which SQL statements are bound automatically (without a user issuing a BIND command) when an application process begins execution and the bound application plan or package it requires is not valid.

**auxiliary table** A table that stores columns outside the table in which they are defined.

**base table** (1) A table created by the SQL CREATE TABLE statement that is used to hold persistent data. Contrast with *result table* and *temporary table*. (2) A table containing an LOB column definition. The actual LOB column data is not stored along with the base table. The base table contains a row identifier for each row and an indicator column for each of its LOB columns. Contrast with *auxiliary table*.

**basic mode** A S/390 central processing mode that does not use logical partitioning. Contrast with *logically partitioned (LPAR) mode*.

**bean** A definition or instance of a JavaBeans component. See *JavaBeans*.

**bind** The process by which the output from the DB2 precompiler is converted to a usable control structure called a *package* or an *application plan*. During the process, access paths to the data are selected, and some authorization checking is performed.

**browser** (1) In VisualAge® for Java, a window that provides information about program elements. There are browsers for projects, packages, classes, methods, and interfaces. (2) An Internet-based tool that lets users browse Web sites.

**bytecode** Machine-independent code generated by the Java compiler and executed by the Java interpreter.

**call-level interface (CLI)** A callable API for database access, which is an alternative to using embedded SQL. In contrast to embedded SQL, DB2 CLI does not require the user to precompile or bind applications, but instead provides a standard set of functions to process SQL statements and related services at run time.

**cascading stylesheet (CSS)** Defines a stylesheet language for HTML 4.0. Allows a Web page designer to separately specify style elements of a Web page, such as colors, fonts, and font styles.

**case-sensitive** Indicates whether an application, processor, or operating system distinguishes between uppercase and lowercase. If it does, it is case-sensitive. XML tags are case-sensitive, but HTML tags are not.

**casting** Explicitly converting an object or primitive data type.

**catalog** In DB2, a collection of tables that contains descriptions of objects such as tables, views, and indexes.

**catalog table** Any table in the DB2 catalog.

**cell** A logical collection of one or more nodes in a WebSphere Application Server administrative domain. It is defined by an administrators to meet their administration needs. In a z/OS environment, a cell can consist of nodes on multiple logical partitions (LPARs) in a sysplex.

**CGI** See *Common Gateway Interface*.

**channel-attached** (1) Pertaining to the attachment of devices directly by data channels (I/O channels) to a computer. (2) Pertaining to devices attached to a controlling unit by cables rather than by telecommunication lines.

**character large object (CLOB)** A sequence of bytes representing single-byte characters or a mixture of single and double-byte characters where the size can be up to 2 GB - 1. Although the size of character large object values can be anywhere up to 2 GB - 1, in general, it is used whenever a character string might exceed the limits of the VARCHAR type.

**class** An encapsulated collection of data and methods to operate on the data. A class may be instantiated to produce an object that is an instance of the class.

**class hierarchy** The relationships between classes that share a single inheritance. All Java classes inherit from the object class.

**class method** Methods that apply to the class as a whole rather than its instances (also called a *static method*).

**class variable** Variables that apply to the class as a whole rather than its instances (also called a *static field*).

**CLASSPATH** In a deployment environment, the environment variable keyword that specifies the directories in which to look for class and resource files.

**CLI** See *call level interface*.

**client** A networked computer in which the integrated development environment (IDE) is connected to a repository on a team server. See *requester*.

**CLOB** See *character large object (CLOB)*.

**cluster** A set of servers within a cell that are managed together and participate in workload management. A “vertical cluster” is a grouping of multiple servers running on the same system. A “horizontal cluster” includes servers on different systems that span a sysplex.

**codebase** An attribute of the <APPLET> tag that provides the relative path name for the classes. Use this attribute when your class files reside in a different directory than your HTML files.

**column function** An SQL operation that derives its result from a collection of values across one or more rows. Contrast with *scalar function*.

**commit** The operation that ends a unit of work by releasing locks so that the database changes made by that unit of work can be perceived by other processes.

**Common Connector Framework** In the Enterprise Access Builder, interface and class definitions that provide a consistent means of interacting with enterprise resources (for example, CICS and Encina® transactions) from any Java execution environment.

**Common Gateway Interface (CGI)** A means of allowing a Web server to run a program that you provide, rather than to retrieve a file. A number of popular Web servers support the CGI. For some applications (for example, displaying information from a database), you must do more than simply retrieve an HTML document from a disk and send it to the Web browser. For such applications, the Web server has to call a program to generate the HTML to be displayed. The CGI is not the only such interface, however.

**connection** In the VisualAge for Java Visual Composition Editor, a visual link between two components that represents the relationship between the components. Each connection has a source, a target, and other properties.

**connection handle** The data object that contains information associated with a connection managed by DB2 CLI. This includes general status information, transaction status, and diagnostic information.

**constant** A user-defined data type that shares a common representation with built-in data types. Contrast with *variable*.

**content model** In XML, the expression that specifies which elements and data are allowed within an element.

**controller** The same concept as a control region of WebSphere for z/OS v4. A controller is a z/OS address space in which authorized programs run; it manages communication to servants. Each server can only have one controller.

**cookie** (1) A small file stored on an individual's computer; this file allows a site to tag the browser with a unique identification. When a person visits a site, the site's server requests a unique ID from the person's browser. If this browser does not have an ID, the server delivers one. On the Wintel platform, the cookie is delivered to a file called *cookies.txt*, and on a Macintosh platform, it is delivered to *MagicCookie*. Just as someone can track the origin of a phone call with Caller ID, companies can use cookies to track information about behavior. (2) Persistent data stored by the client in the Servlet Builder.

**CRM** See *customer relationship management*.

**cursor** A named control structure used by an application program to point to a row of interest within some set of rows, and to retrieve rows from the set, possibly making updates or deletions.

**customer relationship management (CRM)** Includes the systems and infrastructure required to analyze, capture and share all parts of the customer's relationship with the enterprise. From a strategy perspective, it represents a process to measure and allocate organizational resources to those activities that have the greatest return and impact on profitable customer relationships.

**DASD** See *database management system*.

**Data Access Bean** In the VisualAge for Java Visual Composition Editor, a bean that accesses and manipulates the content of JDBC/ODBC-compliant relational databases.

**Data Access Builder** A VisualAge for Java Enterprise tool that generates beans to access and manipulate the content of JDBC/ODBC-compliant relational databases.

**data source** A local or remote relational or non-relational data manager that is capable of supporting data access via an ODBC driver which supports the ODBC APIs. In the case of DB2 for OS/390, the data sources are always relational database managers.

**database management system (DBMS)** A software system that controls the creation, organization, and modification of a database and access to the data stored within it.

**DB2 thread** The DB2 structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to DB2 resources and services.

**DBCLOB** See *double-byte character large object*.

**DBMS** See *database management system*.

**direct access storage device (DASD)** A mass storage medium on which a computer stores data.

**distinct type** A user-defined data type that shares a common representation with built-in data types.

**distributed relational database architecture (DRDA)** A connection protocol for distributed relational database processing that is used by IBM's relational database products. Includes protocols for communication between an application and a remote relational DBMS, and for communication between relational DBMS.

**DLL** See *dynamic link library*.

**Document Object Model** (**DOM)** Allows the representation and manipulation of an XML document in memory as a programming object. DOM is defined by the WorldWide Web Consortium.

**Document Type Definition (DTD)** Defines which elements and attributes are acceptable in a specific XML file. Defines a subset of XML which may be used for a particular application.

**DOM** See *Document Object Model*.

**DOM Tree** An in-memory representation of an XML Document.

**double precision** A floating-point number that contains 64 bits. See also *single precision*.

**double-byte character large object (DBCLOB)** A sequence of bytes representing double-byte characters where the size can be up to 2 GB. Although the size of double-byte character large object values can be anywhere up to 2 GB, in general, they are used whenever a double-byte character string might exceed the limits of the VARGRAPHIC type.

**DRDA** See *distributed relational database architecture*.

**DTD** See *Document Type Definition*.

**duplex** Pertaining to communication in which data or control information can be sent and received at the same time. Contrast with *half duplex.*

**dynamic bind** A process by which SQL statements are bound as they are entered.

**Dynamic I/O Reconfiguration** An S/390 function that allows I/O configuration changes to be made non-disruptively to the current operating I/O configuration.

**dynamic link library (DLL)** A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a dynamic link library can be shared by several applications simultaneously. The DLL's Enterprise Access Builders also generate platform-specific DLLs for the workstation and OS/390 platforms.

**dynamic SQL** SQL statements that are prepared and executed within an application program while the program is executing. In dynamic SQL, the SQL source is contained in host language variables rather than being coded into the application program. The SQL statement can change several times during the application program's execution.

**EBCDIC** See *Extended Binary Coded Decimal Interchange Code*.

**EBNF** See *Extended Backus-Naur Form*.

**EDI** See *electronic data interchange*.

**EJB** See *Enterprise JavaBean*.

**electronic data interchange (EDI)** The automatic machine-to-machine transfer of trading documents (for example, invoices and purchase orders) using electronic networks such as the Internet. Originally conducted only through value-added networks, EDI is gradually moving to the Internet.

**element** In XML, a start tag and its end tag, plus the content between the tags. An empty tag is also an element.

**embedded SQL** SQL statements coded within an application program. See *static SQL*.

**EmbeddedJava** An API and application environment for high-volume embedded devices, such as mobile phones, pagers, process control, instrumentation, office peripherals, network routers, and network switches. Such applications run on real-time operating systems and are optimized for the constraints of small-memory footprints and diverse visual displays.

**empty declaration** In XML, the DTD declaration for an empty tag. For example, if `<foo/>` is an empty tag, the empty declaration looks like: `<!ELEMENT foo EMPTY>`.

**empty tag** In XML, a start and end tag combined in one tag. The tag has a trailing slash, so an XML parser can immediately recognize it as an empty tag and not bother looking for a matching end tag. For example, if `foo` is an empty tag, it looks like `<foo/>`.

**Enterprise Java** Includes Enterprise JavaBeans and open API specifications for database connectivity, naming and directory services, CORBA/IIOP interoperability, pure Java distributed computing, messaging services, managing system and network resources, and transaction services.

**Enterprise JavaBeans (EJB)** A cross-platform component architecture for the development and deployment of multi-tier, distributed, scalable, object-oriented Java applications. The EJB specification defines a way of building transactionally-aware business objects in Java.

**Enterprise Systems Architecture/390® (ESA/390)** An IBM architecture for mainframe computers and peripherals. Processors that follow this architecture include the S/390 Server family of processors.

**entity** In XML, an entity declaration provides the ability to have constants or replacement strings, which are expanded by a pre-processor. An entity declaration maps some token to a replacement string. Later the token can be prefixed with the ampersand (&) character and the replacement string is put in its place.

**environment handle** In DB2 ODBC, the data object that contains global information regarding the state of the application. An environment handle must be allocated before a connection handle can be allocated. Only one environment handle can be allocated per application.

**ESA/390** See *Enterprise Systems Architecture/390*.

**exception** An object that has caused a new condition, such as an error. In Java, *throwing* an exception means passing that object to an interested party; a signal indicates the kind of condition that has taken place. *Catching* an exception refers to receiving the sent object. *Handling* an exception means managing the problem after receiving the object, although it might mean doing nothing (which is undesirable programming practice).

**executable content** Code that runs from within an HTML file (such as an applet).

**Extended Backus-Naur Form (EBNF)** A formal set of production rules that comprise a grammar that defines another language, such as XML.

**Extended Binary Coded Decimal Interchange Code (EBCDIC)** An encoding scheme used to represent character data in the MVS, VM, VSE, and OS/400® environments. Contrast with *ASCII*.

**extends** A subclass or interface extends a class or interface if it adds fields or methods, or overrides its methods.

**Extensible Markup Language (XML)** An important new standard emerging for structured documents on the Web. XML extends HTML beyond a limited tag set and adapts SGML, making it easy for developers to write programs that process this markup and providing for a rich, more complex encoding of information.

**Extensible Stylesheet Language Transformations (XSLT)** Defines the part of the XSL specification which allows the stylesheet to reformat and reorganize the XML data. It is most often used to transform XML into XSL.

**external function** A function for which the body is written in a programming language that takes scalar argument values and produces a scalar result for each invocation. Contrast with *sourced function*.

**extranet** In some cases, intranets have connections to other independent intranets, for example one company connecting its intranet to the intranet of one of its suppliers. Such a connection of intranets is called an *extranet*. Depending on the implementation, they may or may not be fully or partially visible to the outside.

**factory** A bean that dynamically creates instances of beans.

**FastCGI** A way of combining the advantages of CGI programming with some of the performance benefits you get by using the GWAPI. FastCGI, written by Open Market, Inc., is an extension to normal Web server processing. It requires server-specific API support, which is available for AIX®, Sun™ Solaris™, HP-UX, and OS/390. With FastCGI, you can start applications in independent address spaces and pass requests for these applications from the Web server. The communication is through either the TCP/IP sockets interface or UNIX Domain socket bind path in the hierarchical file system (HFS).

**Fibre Channel connection (FICON)** (1) An ESA/390 computer peripheral interface. The I/O interface uses ESA/390 logical protocols over a FICON serial interface that configures attached units to a FICON communication fabric. (2) An FC4 proposed standard that defines an effective mechanism for the export of the SBCON command protocol via Fibre Channels.

**Fibre Channel standard** An ANSI standard for a computer peripheral interface. The I/O interface defines a protocol for communication over a serial interface that configures attached units to a communication fabric. The protocol has four layers. The lower of the four layers defines the physical media and interface. The upper of the four layers defines one or more logical protocols (for example, FCP for SCSI command protocols and FC-SB-2 for FICON® for ESA/390). Refer to ANSI X3.230.1999x.

**FICON** See *Fibre Channel connection*.

**field** A data object in a class, for example a variable.

**File Transfer Protocol (FTP)** In the Internet suite of protocols, an application layer protocol that uses TCP and Telnet services to transfer bulk-data files between machines or hosts.

**first tier** The client; the hardware and software with which the end user interacts.

**foreign key** A key that is specified in the definition of a referential constraint. Because of the foreign key, the table is a dependent table. The key must have the same number of columns, with the same descriptions, as the primary key of the parent table.

**form data** A generated class representing the HTML form elements in a visual servlet.

**FTP** See *File Transfer Protocol*.

**function** A specific purpose of an entity or its characteristic action, such as a column function or scalar function. See *column function* and *scalar function*. Furthermore, functions can be user-defined, built-in, or generated by DB2. See *user-defined function*, *external function*, and *sourced function*.

**garbage collection** Java's ability to clean up inaccessible unused memory areas ("garbage") dynamically. Garbage collection slows performance, but keeps the machine from running out of memory.

**Go Web Server Application Programming Interface (GWAPI)** Because CGI has some architectural limitations, most Web servers provide an equivalent mechanism that is optimized for their native environment. Domino Go Web Server, IBM's strategic Web server, offers GWAPI, optimized for a given environment, such as OS/390. It enables you to create dynamic content similar to the CGI, but in a more specialized way than the generalized CGI. The GWAPI process is similar to OS/390 exit processing. There is an exit point for various server functions that can be exploited.

**GWAPI** See *Go Web Server Application Programming Interface*.

**half duplex** In data communication, pertaining to transmission in only one direction at a time. Contrast with *duplex.*

**handle** In DB2 CLI, a variable that refers to a data structure and associated resources. See *connection handle* and *environment handle*.

**hard disk drive** (1) A storage media within a storage server used to maintain information that the storage server requires. (2) A mass storage medium for computers that is typically available as a fixed disk or a removable cartridge.

**hierarchy** The order of inheritance in object-oriented languages. Each class in the hierarchy inherits attributes and behavior from its superclass, except for the top-level Object class.

**HTML** See *Hypertext Markup Language*.

**HTTP** See *Hypertext Transfer Protocol*.

**HTTPS** HTTPS is a de facto standard developed by Netscape for making HTTP flows secure. Technically, it is the use of HTTP over SSL.

**Hypertext Markup Language (HTML)** A file format, based on SGML, for hypertext documents on the Internet. Allows for the embedding of images, sounds, video streams, form fields, and simple text formatting. References to other objects are embedded using URLs, enabling readers to jump directly to the referenced document.

**Hypertext Transfer Protocol (HTTP)** The Internet protocol, based on TCP/IP, used to fetch hypertext objects from remote hosts.

**IDE** See *integrated development environment*.

**identifier** A unique name or address that identifies such items as programs, devices, or systems.

**initial program load (IPL)** (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction. (3) The process of loading system programs and preparing a system to run jobs.

**integrated development environment (IDE**) In VisualAge for Java, the set of windows that provides the user with access to development tools. The primary windows are the Workbench, Log, Console, Debugger, and Repository Explorer.

**Internet** The vast collection of interconnected networks that use TCP/IP and evolved from the ARPANET of the late 1960s and early 1970s. The number of independent networks connected into this vast global net is growing daily.

**Internet Protocol (IP)** In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical network. However, this protocol does not provide error recovery and flow control, and does not guarantee the reliability of the physical network.

**interpreter** A tool that translates and executes code line by line.

**intranet** A private network inside a company or organization that uses the same kinds of software that you would find on the Internet, but that are only for internal use. As the Internet has become more popular, many of the tools used on the Internet are being used in private networks. For example, many companies have Web servers that are available only to employees.

**IP** See *Internet Protocol*.

**IPL** See *initial program load*.

**JAR file format** JAR (Java archive) is a platform-independent file format that aggregates many files into one. Multiple Java applets and their requisite components (.class files, images, sounds and other resource files) can be bundled in a JAR file and subsequently downloaded to a browser in a single HTTP transaction.

**Java** An object-oriented programming language for portable, interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems™, Incorporated. The Java environment consists of the JavaOS™, the virtual machines for various platforms, the object-oriented Java programming language, and several class libraries.

**Java applet** A small Java program designed to run within a Web browser. It is downloadable and executable by a browser or network computer.

**JavaBeans** Java component architecture, developed by Sun, IBM, and others. The components can be parts of Java programs, or they can exist as self-contained applications. They can be assembled to create complex applications, and they can run within other component architectures, such as ActiveX® and OpenDoc.

**Java Database Connectivity (JDBC)** In the JDK, the specification that defines an API that enables programs to access databases that comply with this standard.

**Java Development Kit (JDK)** The set of Java technologies made available to licensed developers by Sun Microsystems. Each release of the JDK contains the Java Compiler, Java Virtual Machine, Java Class Libraries, Java Applet Viewer, Java Debugger, and other tools.

**Java Naming and Directory Interface (JNDI)** An extension to the Java platform that provides a standard interface for heterogeneous naming and directory services.

**Java Native Interface (JNI)** A native programming interface that allows Java code running inside a Java Virtual Machine (JVM) to interoperate with applications and libraries written in other programming languages, such as C and C++.

**Java Platform** A collective term for the Java language for writing programs; a set of APIs, class libraries, and other programs used in developing, compiling, and error-checking programs; and a JVM which loads and executes the class files. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Java Remote Method Invocation (RMI)** A method of invocation between peers, or between client and server, when applications at both ends of the invocation are written in Java. Included in JDK 1.1.

**Java Runtime Environment (JRE)** A subset of the JDK for end users and developers who want to redistribute the JRE. The JRE consists of the JVM, the Java core classes, and supporting files. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Java Virtual Machine (JVM)** A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

**JavaDoc** The Sun tool for generating HTML documentation on classes by extracting comments from the Java source code files.

**JavaScript** A scripting language used within an HTML page. Superficially similar to Java but JavaScript scripts appear as text within the HTML page. Java applets are programs written in the Java language and are called from within HTML pages or run as stand-alone applications.

**JavaServer Page (JSP)** Web pages that include dynamic tags which are executed on the server. JSPs are the presentation layer for Web-based applications built in Java.

**JDBC** See *Java Database Connectivity.*

**JDK** See *Java Development Kit*.

**JIT** See *Just-In-Time compiler.*

**JNDI** See *Java Naming and Directory Interface.*

**JNI** See *Java Native Interface.*

**JRE** See *Java Runtime Environment*.

**JSP** See *JavaServer Page*.

**Just-In-Time compiler (JIT)** A platform-specific software compiler often contained within JVMs. JITs compile Java bytecodes on-the-fly into native machine instructions, reducing the need for interpretation.

**JVM** See *Java Virtual Machine*.

**LAN** See *local area network.*

**large object (LOB)** A sequence of bytes representing bit data, single-byte characters, double-byte characters, or a mixture of single- and double-byte characters. A LOB can be up to 2 GB -1 bytes in length. See also *CLOB* and *DBCLOB*.

**LIC** See *Licensed Internal Code*.

**Licensed Internal Code (LIC)** Microcode that IBM does not sell as part of a machine, but instead, licenses to the customer. LIC is implemented in a part of storage that is not addressable by user programs. Some IBM products use it to implement functions as an alternate to hard-wire circuitry.

**linker** A computer program for creating load modules from one or more object modules or load modules by resolving cross references among the modules and, if necessary, adjusting addresses. In Java, the linker creates an executable from compiled classes.

**load module** A program unit that is suitable for loading into main storage for execution. The output of a linkage editor.

**LOB** See *large object*.

**local area network (LAN)** A computer network located in a user's premises within a limited geographic area.

**logical partition (LPAR)** A set of functions that create a programming environment that is defined by the ESA/390 architecture. ESA/390 architecture uses this term when more than one LPAR is established on a processor. It is conceptually similar to a virtual machine environment except that the LPAR is a function of the processor. Also, it does not depend on an operating system to create the virtual machine environment.

**logical switch number (LSN)** A two-digit number used by the I/O Configuration Program (IOCP) to identify a specific ESCON® Director.

**logically partitioned (LPAR) mode** A central processor mode, available on the configuration frame when using the PR/SM™ facility. It allows an operator to allocate processor hardware resources among logical partitions. Contrast with *basic mode.*

**LPAR** See *logical partition*.

**LSN** See *logical switch number*.

**megabyte (MB)** (1) For processor storage, real and virtual storage, and channel volume, $2^{20}$ or 1 048 576 bytes. (2) For disk storage capacity and communications volumes, 1 000 000 bytes.

**method** A fragment of Java code within a class that can be invoked and passed a set of parameters to perform a specific task.

**middle tier** The hardware and software that resides between the client and the enterprise server resources and data. The software includes a Web server that receives requests from the client and invokes Java servlets to process these requests. The client communicates with the Web server via industry standard protocols such as HTTP and IIOP.

**middleware** A layer of software that sits between a database client and a database server, making it easier for clients to connect to heterogeneous databases.

**multithreading** Multiple TCBs executing one copy of code concurrently (sharing a processor) or in parallel (on separate central processors).

**National Committee for Information Technology Standards (NCITS)** Develops national standards, and its technical experts participate on behalf of the United States in the international standards activities of ISO/IEC JTC 1, information technology.

**native class** Machine-dependent C code that can be invoked from Java. For multi-platform work, the native routines for each platform need to be implemented.

**NCITS** See *National Committee for Information Technology Standards.*

**node** A node is a logical grouping of managed servers. It usually corresponds to a physical system with a distinct IP host address.

**NUL terminator** In C, the value that indicates the end of a string. For character strings, the NUL terminator is X'00'.

**NUL-terminated host variable** A varying-length host variable in which the end of the data is indicated by the presence of a NUL terminator.

**null** A special value that indicates the absence of information.

**object** The principal building block of object-oriented programs. Objects are software programming modules. Each object is a programming unit consisting of related data and methods.

**ODBC** See *Open Database Connectivity.*

**ODBC driver** A DLL that implements ODBC function calls and interacts with a data source.

**OEMI** See *original equipment manufacturers information.*

**Open Database Connectivity (ODBC)** A Microsoft database API for C that allows access to DBMS by using callable SQL. ODBC does not require the use of an SQL preprocessor. It provides an architecture that lets users add modules, called *database drivers*, that link the application to their choice of DBMS at run time. Therefore applications no longer need to be directly linked to the modules of all the DBMS that are supported.

**open system** A system whose characteristics comply with standards made available throughout the industry and that therefore can be connected to other systems complying with the same standards.

**original equipment manufacturers information (OEMI)** A reference to an IBM guideline for a computer peripheral interface. More specifically, refer to IBM S/360™ and S/370™ Channel to Control Unit Original Equipment Manufacturer's Information. The interface uses ESA/390 logical protocols over an I/O interface that configures attached units in a multi-drop bus environment.

**package** A program element that contains classes and interfaces.

**persistence** In object models, a condition that allows instances of classes to be stored externally, for example in a relational database.

**Persistence Builder** In VisualAge for Java, a persistence framework for object models, which enables the mapping of objects to information stored in relational databases and provides linkages to legacy data on other systems.

**plan** See *application plan*.

**plan name** The name of an application plan.

**precompilation** Processing of application programs containing SQL statements that takes place before compilation. SQL statements are replaced with statements that are recognized by the host language compiler. Output from this precompilation includes source code that can be submitted to the compiler and the database request module (DBRM) that is input to the bind process.

**prepare** The first phase of a two-phase commit process in which all participants are requested to prepare for commit.

**prepared SQL statement** A named object that is the executable form of an SQL statement that has been processed by the PREPARE statement.

**primary key** A unique, non-null key that is part of the definition of a table. A table cannot be defined as a parent unless it has a unique key or primary key.

**process** A program executing in its own address space, containing one or more threads.

**program temporary fix (PTF)** A temporary solution or bypass of a problem diagnosed by IBM in a current unaltered release of a program.

**property** An initial setting or characteristic of a bean, for example, a name, font, text, or positional characteristic.

**PTF** See *program temporary fix*.

**RDBMS** See *relational database management system*.

**re-entrant** Executable code that can reside in storage as one shared copy for all threads. Reentrant code is not self-modifying and provides separate storage areas for each thread. Re-entrancy is a compiler and operating system concept. Alone, it is not enough to guarantee logically consistent results when multithreading.

**reference** An object's address. In Java, objects are passed by reference rather than by value or by pointers.

**relational database management system (RDBMS)** A relational database manager that operates consistently across supported IBM systems.

**remote** Refers to any object maintained by a remote DB2 subsystem, that is by a DB2 subsystem other than the local one. A remote view, for instance, is a view maintained by a remote DB2 subsystem.

**Remote Method Invocation (RMI)** A specific instance of the more general term RPC. RMI allows objects to be distributed over the network. That is a Java program running on one computer can call the methods of an object running on another computer. RMI and java.net are the only 100% pure Java APIs for controlling Java objects in remote systems.

**Remote Object Instance Manager** In RMI, a program that creates and manages instances of server beans through their associated server-side server proxies.

**Remote Procedure Calls (RPC)** A generic term that refers to any of a series of protocols used to execute procedure calls or method calls across a network. RPC allows a program running on one computer to call the services of a program running on another computer.

**requester** Also application requester (AR). The source of a request to a remote RDBMS, the system that requests the data.

**result table** The set of rows produced by the evaluation of a SELECT statement. Contrast with *temporary table*.

**RMI** See *Remote Method Invocation.*

**rollback** The process of restoring data changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with *commit*.

**RPC** See *Remote Procedure Calls*.

**run-time system** The software environment where compiled programs run. Each Java runtime system includes an implementation of the JVM.

**sandbox** A restricted environment, provided by the Web browser, in which Java applets run. The sandbox offers them services and prevents them from performing tasks such as file I/O or talking to servers other than the one from which the applet was loaded. The analogy of applets to children led to calling the environment in which they run the "sandbox".

**scalar function** An SQL operation that produces a single value from another value and is expressed as a function name followed by a list of arguments enclosed in parentheses. See also *column function*.

**SCSI** See *Small Computer System Interface*.

**Secure Sockets Layer (SSL)** A security protocol that allows communications between a browser and a server to be encrypted and secure. It prevents eavesdropping, tampering, or message forgery on your Internet or intranet network.

**security** Features in Java that prevent applets downloaded off the Web from deliberately or inadvertently doing damage. One such feature is the digital signature, which ensures that an applet came unmodified from a reputable source.

**serialization** Turning an object into a stream and back again.

**servant** Same as the server region in WebSphere for z/OS version 4. A servant is an z/OS address space in which JVM runs. A controller and one or more servants make up a server.

**server** The computer that hosts the Web page that contains an applet. The .class files that make up the applet, and the HTML files that reference the applet reside on the server. When someone on the Internet connects to a Web page that contains an applet, the server delivers the .class files over the Internet to the client that made the request. The server is also known as the *originating host*.

**server bean** The bean that is distributed using RMI services and is deployed on a server.

**servlet** A Java program that runs on a Web server and extends the server's functionality by generating dynamic content in response to Web client requests. Servlets are commonly used to connect databases to the Web.

**SGML** See *Standardized Generalized Markup Language.*

**shell** The user interface of UNIX system software. In z/OS, an xpg4.2-compliant shell is used. Often OMVS is used as an interface for z/OS shells.

**single precision** A floating-point number that contains 32 bits. See also double precision.

**Small Computer System Interface (SCSI)** (1) An ANSI standard for a logical interface to computer peripherals and for a computer peripheral interface. The interface uses a SCSI logical protocol over an I/O interface that configures attached targets and initiators in a multi-drop bus topology. (2) A standard hardware interface that enables a variety of peripheral devices to communicate with one another.

**SmartGuide** In IBM software products, an active form of help that guides you through common tasks.

**source type** An existing type that is used to internally represent a distinct type.

**sourced function** A function that is implemented by another built-in or user-defined function already known to the database manager. This function can be a scalar function or a column (aggregating) function; it returns a single value from a set of values (for example, MAX or AVG). Contrast with *external function*.

**SPUFI** In DB2 UDB for OS/390, SQL Processor Using File Input.

**SQL** See *Structured Query Language*.

**SSL** See *Secure Sockets Layer*.

**Standardized Generalized Markup Language (SGML)** An ISO/ANSI/ECMA standard that specifies a way to annotate text documents with information about types of sections of a document.

**static bind** A process by which SQL statements are bound after they have been precompiled. All static SQL statements are prepared for execution at the same time. Contrast with *dynamic bind*.

**static SQL** SQL statements, embedded within a program, that are prepared during the program preparation process (before the program is executed). After being prepared, the SQL statement does not change (although values of host variables specified by the statement might change).

**stored procedure** A user-written application program that can be invoked through the use of the SQL CALL statement.

**Structured Query Language (SQL)** A standardized language for defining and manipulating data in a relational database.

**Sysout** The regular output for a program on z/OS is SYSOUT. It is the functional equivalent of stdout on UNIX. In batch, there can be multiple SYSOUTs.

**system** A single instance of the z/OS or OS/390 operating system in a sysplex.

**task control block (TCB)** A control block used to communicate information about tasks within an address space that are connected to DB2. An address space can support many task connections (as many as one per task), but only one address space connection. It manages dispatchable tasks. Each UNIX thread is assigned to a TCB. See *address space connection*.

**TCB** See *Task Control Block*.

**Telnet** Provides a virtual terminal facility that allows users of one computer to act like they are using a terminal connected to another computer. The Telnet client program communicates with the Telnet daemon on the target system to provide the connection and session.

**temporary table** A table created by the SQL CREATE GLOBAL TEMPORARY TABLE statement that is used to hold temporary data. Contrast with *result table*.

**thin client** Usually refers to a system that runs on a resource-constrained machine or that runs a small operating system. Thin clients do not require local system administration, and they execute Java applications delivered over the network.

**third tier** The third tier, or back end, is the hardware and software that provide database and transactional services. These back-end services are accessed through connectors between the middle-tier Web server and the third-tier server. Though this conceptual model depicts the second and third tier as two separate machines, the NCF model supports a logical three-tier implementation in which the software on the middle and third tier is on the same box.

**thread** A separate flow of control within a program.

**time stamp** A seven-part value that consists of a date and time expressed in years, months, days, hours, minutes, seconds, and microseconds.

**trace** A facility that provides the ability to monitor and collect monitoring, auditing, performance, accounting, statistics, and serviceability data.

**trading communities** These bring together buyers and sellers in a central online location to trade, using various online mechanisms including auctions and exchanges, in addition to industry content and application services. Trading communities are owned and operated by both large industry players in closed trading networks, and by neutral parties in more fragmented open communities.

**transaction** (1) In a CICS program, an event that queries or modifies a database that resides on a CICS server. (2) In the Persistence Builder, a representation of a path of code execution. (3) The code activity necessary to manipulate a persistent object. For example, a bank application might have a transaction that updates a company account.

**UDF** See *user-defined function.*

**UDT** See *user-defined data type*.

**Unicode** A 16-bit international character set defined by ISO 10646. See also *ASCII*.

**Uniform Resource Locator (URL)** The unique address that tells a browser how to find a specific Web page or file.

**URI/URL** A Uniform Resource Identifier (URI) and Uniform Resource Locator (URL) uniquely define a location on the Web. URLs are familiar to anyone who browses the Web (for example, http://www.ibm.com), and the term URI is a more general term which also incorporates other schemes for identifying resources.

**URL** See *Uniform Resource Locator.*

**user-defined data type (UDT)** See *distinct type*.

**user-defined function (UDF)** A function defined to DB2 using the CREATE FUNCTION statement that can be referenced thereafter in SQL statements. A UDF can be either an external function or a sourced function.

**valid** An XML document is valid if its content conforms to the rules in its DTD.

**variable** (1) An identifier that represents a data item whose value can be changed while the program is running. The values of a variable are restricted to a certain data type. (2) A data element that specifies a value that can be changed. A COBOL elementary data item is an example of a variable. Contrast with *constant*.

**vi** A popular UNIX editor. It can only be used from an ASCII Telnet connection.

**virtual machine** A software or hardware implementation of a CPU that manages the resources of a machine and can run compiled code. See *Java Virtual Machine*.

**visual bean** In the Visual Composition Editor, a bean that is visible to the end user in the graphical user interface (GUI).

**WAP** See *Wireless Application Protocol*.

**Web** See *World Wide Web.*

**Web application** A WebSphere Web application is a collection of static pages, JSPs, and servlets that share a common URL prefix, and together make a complete application.

**Web browser** The Web uses a client/server processing model. The Web browser is the client component. Examples of Web browsers include Mosaic, Netscape Navigator, and Microsoft Internet Explorer. The Web browser is responsible for formatting and displaying information, interacting with the user, and invoking external functions, such as Telnet, or external viewers for data types that it does not directly support. Web browsers are fast becoming the universal client for the GUI workstation environment, in much the same way that the ability to emulate popular terminals such as the DEC VT100 or IBM 3270 allows connectivity and access to character-based applications on a wide variety of computers. Web browsers are available for all popular GUI workstation platforms and are inexpensive (often included with operating systems or related products for no additional charge).

**Web server** Are responsible for servicing requests for information from Web browsers. The information can be a file retrieved from the server's local disk or generated by a program called by the server to perform a specific application function. Web servers are sometimes referred to as httpd servers or daemons. A number of Web servers are available for most platforms including most UNIX variants, OS/2® Warp, OS/390, and Windows NT®.

**well-formed** An XML document is well-formed if there is one root element, and all its child elements are properly nested within each other. Start tags must have end tags, and each empty tag must be designated as such with a trailing slash. Also, all attributes must be quoted, and all entities must be declared.

**white space** In XML, characters that are not visible, but used in formatting documents or programs. These characters include the SPACE, TAB, NEWLINE, and CARRIAGE-RETURN characters.

**Wireless Application Protocol (WAP)** An open industry standard for mobile Internet access that allows mobile users with wireless devices to easily and instantly access and interact with information and services.

**World Wide Web (WWW)** A network of servers that contain programs and files. Many of the files contain hypertext links to other documents available through the network.

**WWW** See *World Wide Web*.

**XML** See *Extensible Markup Language*.

**XSL Stylesheet** The Extensible Stylesheet Language defines stylesheets for XML Documents. It is composed of two parts: the formatting objects, and XSLT (see *XSLT*). XSL is defined by the WorldWide Web Consortium.

**XSLT** See *Extensible Stylesheet Language Transformations*.

# Related publications

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

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 343. Note that some of the documents referenced here may be available in softcopy only.

- *Patterns: Broker Interactions for Intra- and Inter-enterprise*, SG24-6075
- *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303
- *Patterns: Serial Process Flows for Intra- and Inter-enterprise*, SG24-6305
- *Patterns: Serial and Parallel Processes for Process Choreography and Workflow*, SG24-6306
- *WebSphere Business Integration Server Foundation V5.1 Handbook*, SG24-6318
- *WebSphere Business Integration Server Foundation V5.1 for z/OS*, SG24-6382
- *Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager*, SG24-6513
- *EJB 2.0 Development with WebSphere Studio Application Developer*, SG24-6819
- *Architecting High Availability e-business on IBM @server* zSeries, SG24-6850
- *Self-Service Applications using IBM WebSphere V5.0 and WebSphere MQ Integrator V2.1 Patterns for e-business Series*, SG24-6875
- *WebSphere Version 5.1 Application Developer 5.1.1 Web Services Handbook*, SG24-6891
- *WebSphere Application Server Enterprise V5 and Programming Model Extensions WebSphere Handbook Series*, SG24-6932

- *Patterns: Direct Connections for Intra- and Inter-enterprise*, SG24-6933
- *Patterns: Information Aggregation and Data Integration with DB2 Information Integrator*, SG24-7101

## Other publications

These publications are also relevant as further information sources:


- *z/OS MVS Planning: Workload Management*, SA22-7602
- *OS/390 V2R10.0 UNIX System Services Planning*, SC28-1890
- Adams, Jonathan; Galambos, George; Koushik, Srinivas; and Vasudeva, Guru. *Patterns for e-business: A Strategy for Reuse*. MC Press, October 2001. ISBN 1-931182-02-7
- Hammer, M. and Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. Collins, revised and updated edition, January 2004. ISBN 0-06055-953-5.
- Davenport, T.: *Process Innovation: Reengineering Work Through Information Technology*, Havard Business School Press, October 1992. ISBN 0-87584-366-2.
- Senge, Peter M: *The Fifth Discipline: The Art and Practice of the Learning Organization*. Doubleday Publishing, New York, first edition, September 1994. ISBN 0-38526-095-4.
- Olle, T.W., Hagelstein, J., McDonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M., Verrijn-Stuard, A.A.: *Information Systems Methodologies: A Framework for Understanding*. Addison Wesley, 2nd edition, 1991. ISBN 0-20154-443-1.
- Simison, G.C. “A methodology for Business Process Reengineering, IFIP Transactions”.

## Online resources

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

- *WebSphere Business Integration Server Foundation V5.1, Building and Deploying Service-Oriented Applications That Extend and Integrate Existing IT Assets*, published by A D.H. Brown Associates, Inc.

  http://www-306.ibm.com/software/integration/wbisf/library/

- Mohan, C. “Workflow Management in the Internet Age”, Advances in Databases and Information Systems, W. Litwin, T. Morzy, G. Vossen (Eds.), Lecture Notes in Computer Science Volume 1475, pp26-34, Springer Verlag. Proc. 2nd East-European Symposium on Advances in Databases and Information Systems (ADBIS'98), Poznan, Poland, September 1998, at:

  http://www.almaden.ibm.com/cs/exotica/exotica_papers.html

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

**ibm.com**/redbooks

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# Index

## D



## E



## F



## G



## H

## Q



## R



## S

## T



## U



## W



## X



## Z

Redbooks

Flexible Self-Service Application Patterns Using WebSphere and Process

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

# Flexible Self-Service Application Patterns Using WebSphere and Process Choreography on z/OS

**Build solutions using WebSphere BI Server for z/OS V5.1**

**Explore self-service applications and the Decomposition pattern**

**Follow the sample scenarios**

IT architects are responsible for working with lines of business staff and capturing business needs. This person must translate those requirements into an IT solution that satisfies the current business goals and exhibits several characteristics.

To improve this process, the IT architect needs to identify and leverage existing assets. In this IBM Redbook, you learn how to leverage a key set of assets, the IBM Patterns for e-business. You learn how to use these patterns with key architectural elements to deliver on demand and autonomic architectures for the best qualities of service. This book is designed as a resource for IT architects who specialize in enterprise or mainframe systems. It presents business scenarios that you might come across and best practice design guidelines to help you implement solutions.

Part 1 provides an overview of the IBM Patterns for e-business, with a focus on Self-Service business patterns. Part 2 explains concepts, design considerations, technologies, and architectural elements that are key when designing an architecture based on the IBM Eserver zSeries platform. In Part 3 we explain how to prepare and set up the WebSphere Business Integration Server Foundation (WBISF) run-time environment on z/OS in order to deploy and test the sample business process application scenarios. Part 4 describes the implementation of these Self-Service business pattern sample scenarios on the zSeries platform.

SG24-6356-00 ISBN 0738494283

**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