

IBM WebSphere Process Server Best Practices in Error Prevention Strategies and Solution Recovery



Preventive practices



Strategies for recovery



Troubleshooting tips



Kent Below
Jeff Brent
Eric Herness



International Technical Support Organization

**IBM WebSphere Process Server Best Practices in Error
Prevention Strategies and Solution Recovery**

December 2008

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

First Edition (December 2008)

This edition applies to IBM WebSphere Process Server Version 6.1 and IBM WebSphere Enterprise Server Bus Version 6.1.

This document was created or updated on December 29, 2008.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v
Trademarks	vi
Preface	vii
The team that wrote this paper	vii
Become a published author	viii
Comments welcome	viii
Chapter 1. WebSphere Process Server solution recovery	1
Chapter 2. Recovery patterns, anti-patterns, rules, and helpful ideas.	5
Chapter 3. Preventive practices.	7
3.1 Creating an error handling strategy	8
3.2 Module design and connectivity groups	8
3.3 Application design, exception types, and fault processing	9
3.3.1 Service business exceptions	9
3.3.2 Service runtime exceptions	10
3.4 Functional and system testing	10
3.5 Continual and regularly scheduled environment turning	11
3.6 Infrastructure monitoring	11
3.6.1 IBM Tivoli Composite Application Manager	11
3.7 Solution-specific problem determination methodology	13
3.8 Software currency	14
3.9 Additional stability recommendations	14
3.9.1 Automated environment creation	14
3.9.2 Automated application deployment	14
Chapter 4. Where data goes when failures occur	15
4.1 Use case	16
4.2 Business process engine	17
4.3 Failed Event Manager	17
4.3.1 When failed events are created	18
4.4 Service integration bus destinations	19
4.4.1 SCA module destination	19
4.4.2 System integration bus retry	19
4.4.3 System exception destinations	21
4.4.4 Failed Event Manager and service integration bus destinations	22
4.5 Summary	23
Chapter 5. Recovery scenarios	25
5.1 Scenario types	26
5.1.1 Production environment scenario	26
5.1.2 Test environment scenario	27
5.2 Basic procedures	27
5.2.1 Conducting a situational analysis	28
5.3 How to use the provided information	30
5.3.1 Description of the scenario	30
5.3.2 Assumptions	30

5.3.3 Recovery procedure	30
5.4 Solution recovery scenarios in a production environment.	31
5.4.1 Abnormal termination	31
5.4.2 System is not responding	32
5.4.3 System is functional but severely overloaded.	33
5.4.4 System is responsive but fails to initiate new process instances	34
5.5 Reclaiming the test environment.	35
5.5.1 Manually reclaiming the test environment.	35
Appendix A. Troubleshooting tips	39
Gathering the essential documents	40
Restarting the deployment environment	40
Using Business Process Choreographer's predefined views.	41
Tools for evaluating or purging the system	42
Process cleanup service	42
Viewing the service integration bus by using the administrative console	42
Service Integration Bus Explorer.	47
Capturing a Javacore file	48
Starting the server in recovery mode	51
Peer recovery.	51
Quiescing the system.	52
Stopping the processing of asynchronous events.	52
Export bindings	53
Replaying messages from the retention and hold queues	54
Resubmitting failed events	58
Business Process Choreographer maintenance	59
Deleting process templates that are not in use.	59
Deleting human task templates that are not in use.	59
Deleting completed process instances	59
Resolving indoubt transactions	60
Reviewing the diagnostic log for DB2	62
Deleting the completed process instances.	62
Determining the number of finished process instances within a given period of time. . .	65
Business Process Choreographer Explorer tips	68
IBM Support Assistant	74
Recovering the messaging subsystem	76
Appendix B. Additional material	77
Locating the Web material	77
Using the Web material	77
Related publications	79
IBM Redbooks	79
Online resources	79
How to get Redbooks.	79
Help from IBM	80

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:


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

Trademarks

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

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

alphaWorks®
DB2®
developerWorks®

IBM®
Redbooks®
Redbooks (logo) ®

Tivoli®
WebSphere®

The following terms are trademarks of other companies:

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

Excel, Microsoft, 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.

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

Preface

IBM® WebSphere® Process Server and IBM WebSphere Enterprise Service Bus (ESB) are middleware servers that are optimized to enable the execution and management of business process management (BPM) and service-oriented architecture (SOA) solutions. They are built on the foundational capabilities of IBM WebSphere Application Server. Middleware systems execute under various conditions, not all of which are traditionally “good path” conditions. Many of the key features within these products are in place to deal with the uncertainty that can arise through what might appear to be normal operations.

The topic of system analysis and recovery is a broad topic, for which entire books have been written. This IBM Redpaper publication provides basic guidance on how to handle ordinary and extraordinary conditions that might arise. We base this paper on the assumption that you are familiar with the WebSphere Process Server and WebSphere ESB products, the basic architectural principles upon which they build, and the basic kinds of applications that they execute. We also assume that you have a base understanding of integration projects, planning, and implementations.

Unless otherwise specified, the information is relevant to all versions of WebSphere Process Server and WebSphere ESB starting with V6.1.0. Although you will find that most of the information is applicable to the V6.0.2 product line, the internal validation of IBM was conducted on a V6.1.0 WebSphere Process Server cell with the standard *golden topology*. Results and actions might vary slightly on releases prior to V6.1.0.

The team that wrote this paper

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

Kent Below is a Software Engineer in the BPM Bring-up Lab in Rochester, Minnesota. He has over nine years of experience with business process automation and integration. Kent works with BPM development teams to produce consumable products and is frequently involved in resolving customer issues.

Jeff Brent is a Senior Engineer and Technical Lead for the BPM SWAT team. In this role, Jeff is responsible for helping customers resolve difficult technical issues that involve the BPM product suite. Prior to this position, Jeff was a member of IBM Software Services for WebSphere, the IBM Service Component Architecture (SCA) development team. Jeff is a subject matter expert in SCA, integration project best practices, and WebSphere Process Server system recovery. Jeff has over 10 years of integration experience in various roles including product development and service implementation.

Eric Herness is an IBM Distinguished Engineer and the Chief Architect for BPM products in the AIM Division of IBM Software Group. He is a senior member of the WebSphere Foundation Architecture Board and a member of the Software Group Architecture Board Steering Committee. He also directly leads a worldwide team of architects that work on BPM product architecture and direction. Eric has deep experience in distributed systems and object technology. He has had over a decade of key lead architectural roles in WebSphere, including lead roles and technical contributions to Component Broker, J2EE/EJB, WebSphere Enterprise Edition, and WebSphere Business Integration Server Foundation. Eric joined IBM in 1985. He has a bachelor's degree in management information systems from the University

of Wisconsin – Eau Claire and a Master in Business Administration from the Carlson School of Management at the University of Minnesota.

Become a published author

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

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

Find out more about the residency program, browse the residency index, and apply online at:
ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- Send your comments in an e-mail to:

redbooks@us.ibm.com

- Mail your comments to:

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



WebSphere Process Server solution recovery

Solution recovery represents a set of activities that are required in case of an unforeseen circumstance that affects system stability. Recovery activities might be required in the following situations:

- ▶ **Hardware failure**

In the case of a catastrophic hardware failure, the deployed solution might enter an inconsistent state on restart.

- ▶ **Database, network, or infrastructure failure**

In the case of a fundamental infrastructure failure, the solution might require administration to restart or resubmit business transactions after the infrastructure failure is resolved.

- ▶ **Poor tuning or a lack of capacity planning**

Incomplete capacity planning or performance tuning is a common cause for solution instability.

- ▶ **Defects in application module development**

The modules that are part of a custom developed solution can have bugs. These bugs will result in solution instability and failed services. There can be many causes including the following causes among others:

- Business data that was not planned for or unforeseen by the application design
- An incomplete error handling strategy for the application design

- ▶ **WebSphere software defect**

A defect in the WebSphere product causes a backlog of events to process or clear.

WebSphere Process Server is based on WebSphere Application Server, which supports a transactional model for conducting business transactions. WebSphere Process Server builds on this support, providing for loosely-coupled service-oriented architecture (SOA) applications and business process management (BPM) applications. Technically, this means that the following two points are true:

- ▶ WebSphere Process Server relies on databases and messaging systems to achieve these application execution patterns.
- ▶ Transactions are incumbent in messaging systems and database systems. Transactions are compliant with the atomicity, consistency, isolation, durability (ACID) properties.

ACID: To learn more about the ACID properties, see the entry for *ACID property* at the following Web address:

<http://www-01.ibm.com/software/globalization/terminology/ab.jsp#a00>

In addition, with proper tuning and configuration of the available resource managers, no data will be lost in the event of failures of a given part of the system. Transactional integrity, including rollback and recovery mechanisms, are the key components in WebSphere that make this happen.

For these basic WebSphere mechanisms to work properly, the resource managers (database and messaging) must be set up properly. For example, lock timeouts in databases must be set properly, so that when a server recovers, it can complete either a commit or a rollback without encountering lock conditions.

WebSphere Process Server and WebSphere Enterprise Service Bus add additional capabilities that augment those of WebSphere Application Server, to provide a complete solution as it relates to recovering from unexpected failures. Further details about how these features work with the basic WebSphere Application Server features are provided later in this paper. We provide a high level description here.

The core recovery model for WebSphere Process Server is based on units of work. When failures occur during system operations, centered on a single piece of work being accomplished, the system can handle this failure and recover, providing uninterrupted service. This ability occurs through a series of retry mechanisms and error queues.

Application design should be in place to determine if the error is a system error or an application error. System errors are passed back to the infrastructure that supports the calling component, where additional system level recovery can be attempted or a transformation into a more generic business exception can occur. Various retry mechanisms can be configured to run automatically. Additionally, a set of consoles and corresponding APIs, for more human intervention, are provided as appropriate. Many of these capabilities and the failures that they deal with can be leveraged, while the server that contains that work continues processing new requests.

When failures occur that cause one or more servers in a highly available WebSphere cluster to become unavailable, more capabilities within the system are called upon. First, inbound work is routed away from the failing system. This is done by using underlying WebSphere Application Server workload management facilities, which can vary based on protocol, topology, and configuration.

Second, while the system as a whole remains active and available, recovery by an administrator can proceed. This is centered on doing basic triage and then restarting the failing server. This restart replays transactions logs and should clean up most server down situations. The utilization of the error handling mechanisms that are provided by WebSphere

Process Server is sometimes required to administer a complete recovery. Details about some of the actions and effects of recovery are provided later in this paper.

If the entire cluster becomes unavailable or unresponsive, then a more detailed set of actions is necessary. For example, if a shared resource, such as a database, becomes unavailable, then all servers in a cluster will have the same difficulties getting work completed. Procedures for dealing with shared resource failures depend on the specific resource. Various WebSphere techniques to apply to minimize overall downtime and restart stalled work are also explained in this paper.

In some more catastrophic situations, entire machines can become unavailable or certain servers deemed not recoverable. In these cases, advanced features in WebSphere allow recovery of one server's failures to be executed on another server. Through the use of this feature and the prerequisite of having network attached storage or some other mechanism to share logs, this kind of recovery is also possible. See "Peer recovery" on page 51 for details.

The remainder of this paper focuses on situations where servers or entire clusters become unresponsive or unavailable. We focus on various recovery techniques, including how to handle shared resource slowdowns, failures and outages, and how to assess and recover from those kinds of situations.



Recovery patterns, anti-patterns, rules, and helpful ideas

The following actions can prevent a successful recovery:

- ▶ Do not delete WebSphere Application Server transaction logs. Deleting transaction logs can cause a loss of key customer data and extend server recovery time.
- ▶ Do not place the WebSphere Application Server transaction logs local on the cluster members. Instead place them on a shared drive because this is the only way to allow peer recovery, which helps to minimize the downtime during recovery.

Note: Each WebSphere Application Server has a unique set of transaction logs.

- ▶ Do not attempt database operations, as in the following examples, where the result set is large enough to create additional resource contention (OutOfMemory):
 - Performing Business Process Choreographer Explorer operations that return large result sets
 - Executing administrative scripts on process instances without considering the result set size
- ▶ Do not drop or recreate databases in production.
- ▶ Do not uninstall applications as part of your standard recovery procedures. Only uninstall applications with direction provided by the IBM support organization.
- ▶ Do not enable too much trace if the system is overloaded. This action makes things worse, not better. Too much trace causes a slowdown in system throughput and might cause transaction timeouts. It might also add to the problems that need to be addressed, rather than providing the insight needed to tackle the original problems. Obtain immediate assistance from IBM support to define the correct trace specification.
- ▶ Do not experiment with or try new scripts or new commands on production systems.

- Do not put your production servers into “development mode.” Figure 2-1 shows the page that is used to configure an application server that contains this setting.

Application servers > default.AppTarget.cleanup3Node01.0

Use this page to configure an application server. An application server is a server that provides services required to run enterprise applications.

Runtime **Configuration**

General Properties	Container Settings
Name <input type="text" value="default.AppTarget.cleanup3Node01.0"/>	<input type="checkbox"/> Session management
Node Name <input type="text" value="cleanup3Node01"/>	<input type="checkbox"/> SIP Container Settings
<input type="checkbox"/> Run in development mode	<input type="checkbox"/> Web Container Settings
	<input type="checkbox"/> Business Process

Figure 2-1 Configuring the application server

In general, each of the actions listed might be appropriate in a rare place and time. However, never take these steps without a direct recommendation from the WebSphere Process Server support organization.

The following points are always helpful to know or do in recovery situations:

- Take a snap shot of the configuration tree, the project interchange file of the application in question, and the logs that are available. Logs might be overwriting themselves depending on the configuration. Capturing a set early and often is important to a post-mortem analysis. See “IBM Support Assistant” on page 74 for details about IBM Support Assistant, which helps with this type of activity.
- Understand your database settings, especially those that are related to database transaction log file size, connection pools, and lock timeouts.



Preventive practices

As with all IT endeavors, planning against and practicing for extreme situations increases the possibility for a successful recovery. Several required considerations are associated with preparing for system and application recovery, which we describe in this chapter.

The first set of preventative practices focuses on application design and proper usage of the product capabilities. These practices are guidelines and calls-to-action for the teams that use the development tools and create the applications. A strong system of governance, complete with architectural and design guidelines, and appropriate standards that are combined with reviews and checkpoints are essential to building the right kind of application.

The second set of preventative practices focuses more on the governance and development process, which is in place for rolling out projects. Specifically, this set is focused on testing, tuning, measuring, and retesting activities.

In this chapter, we also provide tips for handling the operational environment. Again, this must be part of the overall governance procedure that is involved with taking projects live based on this technology.

3.1 Creating an error handling strategy

The architecture team must understand the tools and capabilities of the product regarding error handling and recovery. This team is responsible for creating the error handling standards that the application development team will follow. The error handling strategy for the project must account for the following concepts:

- ▶ Appropriate usage of units of work (transactions and activity sessions)
- ▶ Declaration and usage of faults and `ServiceBusinessExceptions`
- ▶ Consistent fault processing for all component types, especially Business Process Execution Language (BPEL) and mediation flow components
- ▶ Usage of retry logic and “continue on error” Business Process Choreographer capabilities
- ▶ Appropriate settings for completed process instance deletion
- ▶ Correct usage of synchronous and asynchronous invocation patterns
- ▶ Appropriate usage of Import and Export types
- ▶ Proper usage of the retry capability in mediation flows

In addition to these points, the architecture team must create design patterns where the built-in recovery capabilities (Failed Event Manager and so on) of WebSphere Process Server are used appropriately.

3.2 Module design and connectivity groups

The creation of “connectivity groups” to represent the possible request sources for the system is a best practice. A *connectivity group* is a specific pattern of behavior found in a Service Component Architecture (SCA) module. The connectivity group does not contain stateful component types such as long running business processes and business state machines. These connectivity groups provide encapsulation and isolation of the specific endpoint’s integration requirements. WebSphere Enterprise Service Bus (ESB) mediation modules are commonly used for this purpose because they represent convenient ways to implement *infrastructure* related tasks.

The concept of connectivity groups also provides a convenient way to quiesce the system in case there is a need for recovery. Since a connectivity group module is stateless, then the module can be temporarily stopped, thus cutting off the inbound flow of new events. After the system is recovered and able to process new work, these modules can be restarted.

In Figure 3-1, the highlighted module (in red) is considered part of a connectivity group.

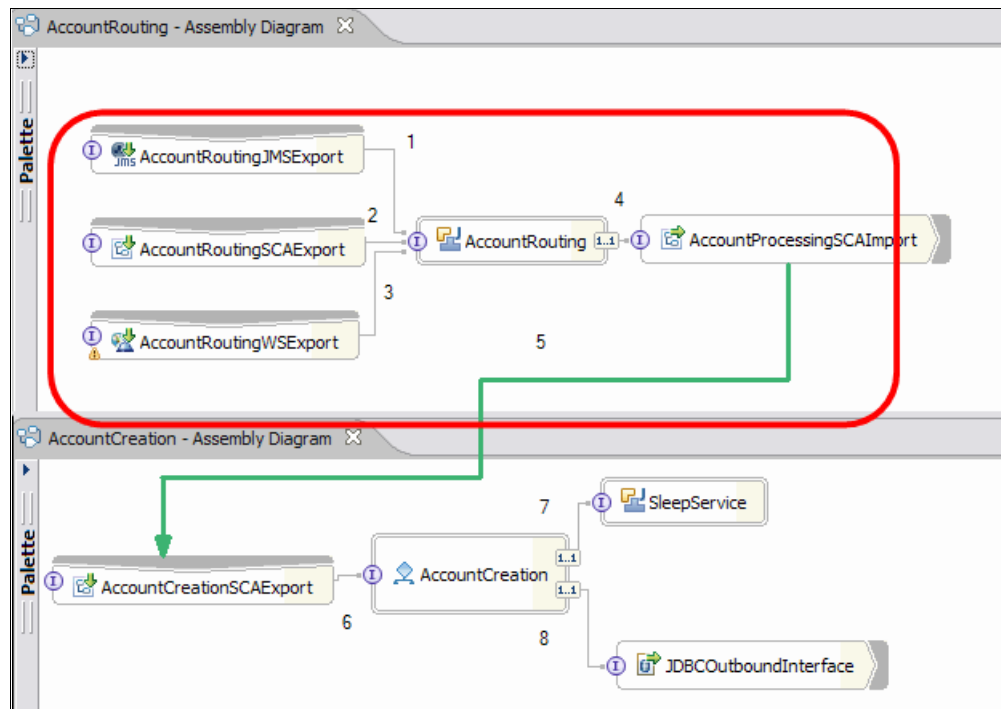


Figure 3-1 Connectivity groups

3.3 Application design, exception types, and fault processing

The proper application design takes advantage of the error handling and fault processing capabilities from WebSphere Process Server and WebSphere ESB. In order to create a comprehensive error handling strategy, solution architects must understand how WebSphere Process Server and WebSphere ESB represent declared and undeclared exceptions.

The SCA programming model provides two types of exceptions:

- ▶ Service business exceptions
- ▶ Service runtime exceptions

3.3.1 Service business exceptions

Service business exceptions are defined on the service interface. They represent known and declared exceptions.

Component developers must take care to declare the possible exceptions that might be thrown from the service so that service consumers can handle them. The WebSphere Process Server and WebSphere ESB business exceptions are returned to the client to catch and handle appropriately.

When dealing with exceptions, service consumers must implement the client so that it performs one of the following actions with a declared business exception:

- ▶ Catch the exception and rewrap it in as a service business exception declared on the client interface.
- ▶ Catch the exception and perform alternate logic.

3.3.2 Service runtime exceptions

Service runtime exceptions are undeclared. These exceptions are used to signal an unexpected condition in the run time.

As a developer of the component, you have three choices for handling these exceptions:

- ▶ You can catch them and perform some alternative logic. For example, if one partner is unable to service a request, another partner might be able to service it.
- ▶ You can catch the exception and rethrow it to your client.
- ▶ You can remap the exception to a business exception. For example, a timeout condition for a partner might result in a business exception that indicates that most of the request was processed. However, one piece of the request was not completed and should be retried later or tried with different parameters.

If an exception is not caught and allowed to pass to the client, the client is responsible for handling the runtime exception. Because runtime exceptions are not declared as part of the interface, component developers should attempt to resolve the exception, and therefore, prevent a runtime exception from inadvertently being propagated to the client if the client is a user interface.

In general, the occurrence of a service runtime exception usually results in a roll back of a transaction for the service. If an asynchronous invocation pattern was used between the client and the service provider, a failed event can be created to represent the failure.

The following are four current subclasses of the `ServiceRuntimeException`:

- ▶ `ServiceExpirationRuntimeException`
This exception is used to indicate that an asynchronous SCA message has expired. Expiration times can be set by using the `RequestExpiration` qualifier on a service reference.
- ▶ `ServiceTimeoutRuntimeException`
This exception is used to indicate that the response to an asynchronous request was not received within the configured period of time. Expiration times can be set by using the `ResponseExpiration` qualifier on a service reference.
- ▶ `ServiceUnavailableException`
This exception is used to indicate that an exception was thrown while invoking an external service through an import.
- ▶ `ServiceUnwiredReferenceRuntimeException`
This exception is used to indicate that the service reference on the component is not wired correctly.

3.4 Functional and system testing

The best tool for problem prevention in production is the execution of a comprehensive functional and system test plan. In general, tests for deployed solutions can be broken into two groups:

- ▶ Functional test
Functional tests confirm the functionality of the implementation compared to the business requirements. These tests are created by business users and application designers.

- **System test**

System tests include cases to verify performance, high availability, and recovery service level agreements (SLAs). It is important to combine test aspects, such as performance and high availability, to evaluate the recovery of a system in extreme production situations.

For both types of tests, the use of automation is strongly recommended. An automated testing strategy provides the organization the ability to prevent regressions from being introduced quickly and effectively.

3.5 Continual and regularly scheduled environment turning

Tuning exercises are a regular part of the system development life cycle. A performance evaluation must be scheduled along with each major application deployment. Each solution released to the production environment should be evaluated and tested in the preproduction environment to measure the impact to existing applications and the current system parameters and resources. If this activity is missed, the solution will suffer a higher than necessary risk of having a recovery problem.

Many publicly available resources describe the process and execution of performance test plans. Review the materials and construct a test plan that is right for your application and topology.

Specific performance tuning information for WebSphere Process Server and WebSphere ESB is available in the product documentation, the performance and tuning Redbooks publications, white papers, and the Performance Report, which accompanies each new release of the BPM and connectivity products from IBM.

3.6 Infrastructure monitoring

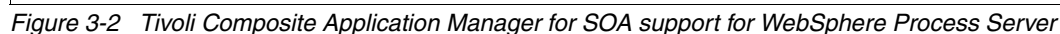
Infrastructure monitoring and the use of infrastructure monitoring tools is a requirement for a production system. The use of these tools provides system administrators the ability to detect critical system behavior and react appropriately to prevent an outage. IBM provides infrastructure monitoring tools such as IBM Tivoli® Composite Application Manager and Tivoli Performance Viewer. Regardless of whether the tool is from the IBM product stack, a required basic level of IT monitoring for the production system is essential to meeting availability SLAs.

3.6.1 IBM Tivoli Composite Application Manager

Tivoli Composite Application Manager can monitor WebSphere Process Server and WebSphere ESB. In addition, it can be used to automate the mediation of problems and manage the configuration and deployment of solutions. It provides the following features, among others, to manage service-oriented architecture (SOA) services:

- Visibility into SOA service interactions
- Visibility into message content and transaction flow patterns
- Ability to identify and isolate performance bottlenecks across technology and platform boundaries
- Lightweight, industry standard Application Response Measurement (ARM)-based performance instrumentation

- Figure 3-2 shows how services, response times, message counts, and message sizes can be monitored.



The screenshot displays the 'Service Operation Response Time - SENTINEL - SYSADMIN' application interface. It features three main views:

- Catalog View:** Located on the left, it shows a tree structure of services under 'Physical' and 'Enterprise' categories. The 'Performance Summary' for 'Patron Service' is selected.
- Threshold View:** Located in the bottom left, it displays a table of service behavior against pre-set thresholds. The table has columns for ServiceName, OperationName, App/Server/Env, Local/Hostname, Local, EventTime, and ElapsedTime. The 'Patron Service' is highlighted in red.
- Dynamic Graphing:** Located in the top right, it displays a bar chart titled 'Average Response Time by Service Operation'. The chart shows response times for various operations: reserve (approx. 220), refIndCopies (approx. 210), getPatronStatus (approx. 200), getPatronId (approx. 190), refIndCopies (approx. 180), find (approx. 170), and checkout (approx. 160).

The bottom status bar indicates the Hub Time is Tue, 04/12/2005 04:14 PM, the Server is Available, and the current view is Service Operation Response Time - SENTINEL - SYSADMIN.

Figure 3-3 Problem detection

In addition, special functions are available that work with WebSphere ESB to dynamically modify mediation flow configuration (Figure 3-4).

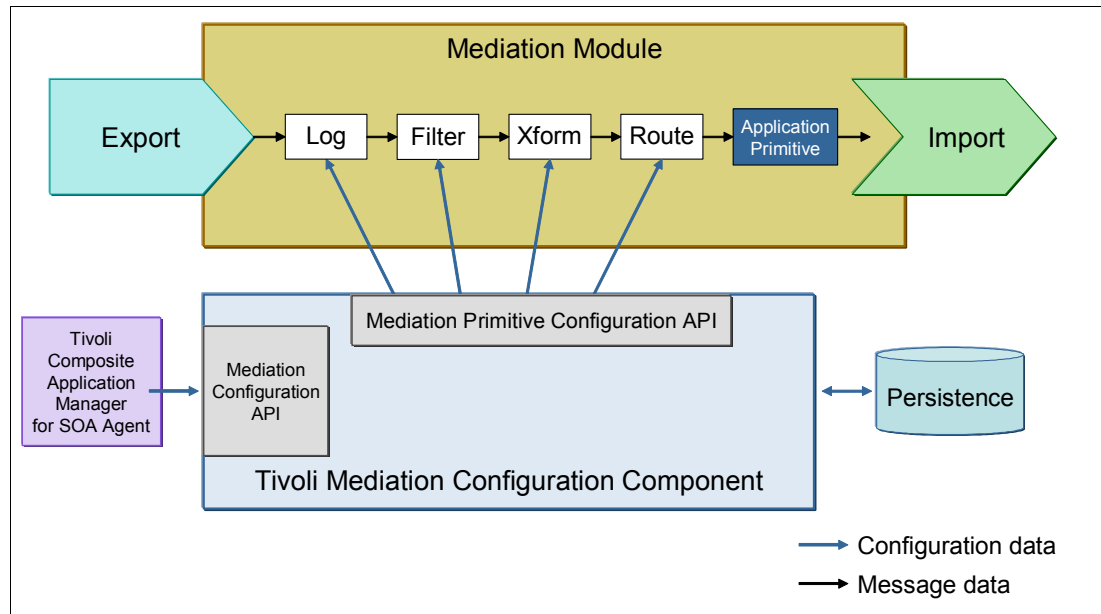


Figure 3-4 Mediation configuration flow

In the following sections, we describe the aspects of the WebSphere Process Server and WebSphere ESB solutions that require additional IT monitoring.

3.7 Solution-specific problem determination methodology

Establish a well articulated and clear problem determination methodology for the solution that is deployed to production. This means that you must document and practice a problem determination methodology on a consistent basis.

We highly recommend that you define the problem determination methodology in an operations manual and include the following information in doing so:

- ▶ An established format for consistently recording your observations during problem determination

Microsoft® Excel® spreadsheets are common observational reporting tools. A sample observational report is included in the download material that is provided with this paper. For more information, see Appendix B, “Additional material” on page 77.
- ▶ The traces to enable, the servers on which to enable them, and the conditions for enabling them

Make sure the trace specification selection does not cause further problems. It is not appropriate to enable everything. Use care to enable the right trace specification for the observed condition. Get help from IBM support and use intelligent situational analysis to collect the correct diagnostic information.
- ▶ How to enable verbose GC
- ▶ How to take heap dumps
- ▶ How to create Javacore files

- ▶ Where and which logs must be collected for opening problem management records (PMRs)
Define the proper usage of IBM MustGather scripts.
- ▶ How to gather information using versionInfo so that all the maintenance package information is included
- ▶ Database specific procedures for gathering logs and information that is recorded by the database as various problems arise

This operations manual should be a living document that is maintained often as new observational practices are learned from functional and system test.

Customers must become familiar with and use IBM Support Assistant (see “IBM Support Assistant” on page 74) and other tools for problem determination and problem reporting. Collection of the information listed in this section should be a prerequisite for opening any new PMR because the inclusion of this data will significantly reduce PMR cycle times.

3.8 Software currency

It is important to maintain the software for the deployed solution. IBM creates regular fix packs to aid in the application of authorized program analysis reports (APARs) in the product base. For more information, see the published list of APAR fixes at the following address:

<http://www-01.ibm.com/support/docview.wss?rs=2307&uid=swg27006649>

3.9 Additional stability recommendations

In addition to the information provided in the previous sections, an infrastructure team can use other measures, as presented in the following sections, to reduce the number of manual processes that can affect solution stability and system recovery.

3.9.1 Automated environment creation

Consistent environment creation is the key to solution stability. You must develop a scripted framework for environment creation. All actions that can be accomplished through the administrative console of WebSphere Process Server or WebSphere ESB can also be done through scripting.

3.9.2 Automated application deployment

To complement an automated environment strategy, write automated scripts to assist in the deployment of application or solution groups to the proper environment. These scripts reduce manual intervention with the environments, in turn reducing the chances of human error on redeployment or recovery. Build on similar procedures that are being used in your WebSphere Application Server production environments.



Where data goes when failures occur

For all production and test recovery activities, there is a finite number of locations in the solution where events can accumulate. If you follow the best practices mentioned previously in this paper, then all business events will reliably accumulate in one of these locations. If sound architectural and application development practices are not followed, then a percentage of in-flight events might find themselves in an inconsistent state that is not recoverable.

When these circumstances are found (presumably during testing cycles), post recovery investigation and cleanup is necessary to correct the issue so that future recovery activities are completely successful. To accurately describe these recovery scenarios, it is helpful to put the information in the context of a use case.

4.1 Use case

Our business has an application that receives a request to create a new account. The solution is comprised of multiple modules as recommended through the module best practices. As shown in Figure 4-1, the first module mediates the request and delegates work to an Account Creation process. We implement the solution as separate modules where the request is passed between the mediation module (AccountRouting) and the processing module (AccountCreation) through an SCA import/export.

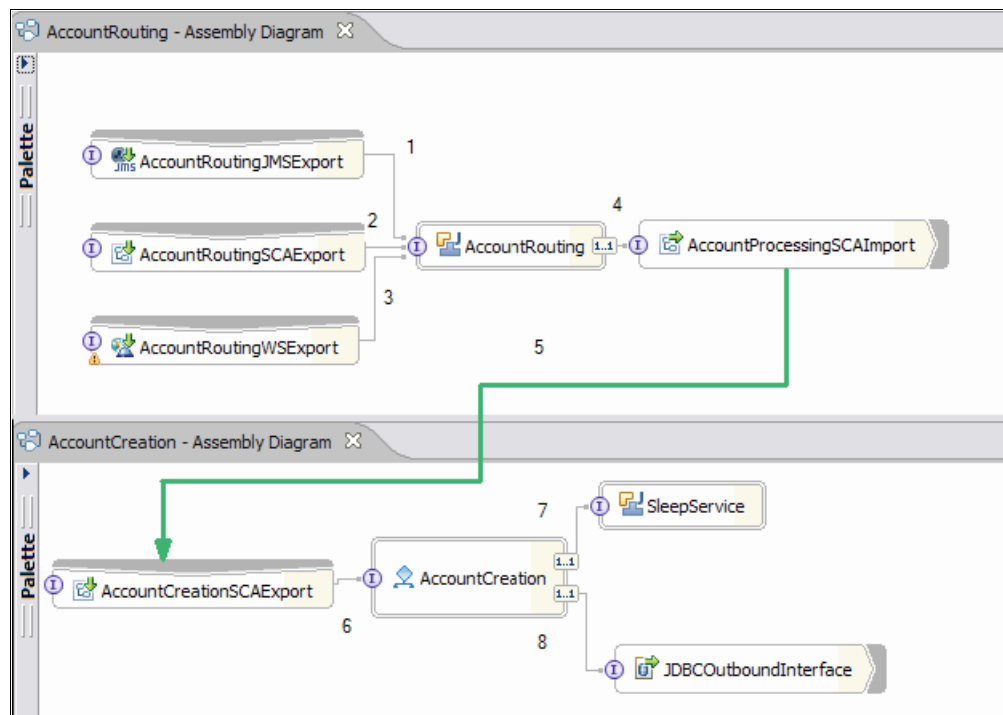


Figure 4-1 Viewing assembly diagrams

Based on the assembly diagram in Figure 4-1, we begin to see the various places where failures can occur. Any of the invocation points can propagate or involve a transaction. There are a few places where data will collect as a result of application or system failures.

In general, transaction boundaries are created and managed by the interaction (synchronous and asynchronous) between components and import/export bindings and their associated qualifiers. Business data will accumulate in specific recovery locations most often due to transaction failure, deadlock, or rollback.

Transaction capabilities within WebSphere Application Server help WebSphere Process Server enlist transactions with service providers. These interactions are particularly important to understand with respect to import and export bindings. Understanding how import/exports are used within your specific business cases is important in determining where events will accumulate in need of recovery.

The project's error handling strategy should clearly define the patterns of interactions, transactions used, and import and export usage before application development. Additionally, the application uses connectivity groups and module development best practices. By using this pattern, we now have the ability to stop the inbound flow of new events by stopping the AccountRouting module.

In the following sections, we address the location of business data in the case of failure and recovery.

4.2 Business process engine

In our business case, we leverage a Business Process Execution Language (BPEL) process for the Account Creation process (Figure 4-1 on page 16).

The following considerations must be made with respect to BPEL and human task management and recovery:

- ▶ What type of process is it? Short or long running? Business state machine? Human task?
- ▶ Is the process developed properly and using fault handling to promote data integrity?
- ▶ How are the invocation patterns and unit of work properties configured to predict and control transaction boundaries?

These considerations will impact your recovery strategy for invocations 7 and 8 that are shown in Figure 4-1 on page 16.

Stateful components, such as long running BPEL processes and business state machines, involve many database transactions where process activity transitions and state changes are committed to the database. Work progresses by updating the database and placing a message on an internal queue that describes what is to be done next. For more information about macro flow transactions, see the “Transactional behavior of long-running processes” topic in the WebSphere Process Server Information Center at the following address:

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

If there are problems processing messages that are internal to the business process engine, these messages are moved to a *retention queue*. The system attempts to continue to process messages. If a subsequent message is successfully processed, the messages on the retention queue are submitted for reprocessing. If the same message is placed on the retention queue five times, it is then placed in the *hold queue*. For information such as which internal queues are used or the retry algorithms for these queues, see the “Recovering from infrastructure failures” topic in the WebSphere Process Server Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/c5replay.html>

See “Replaying messages from the retention and hold queues” on page 54 for additional information about viewing the number of messages and replaying messages.

4.3 Failed Event Manager

The Failed Event Manager is available in WebSphere Process Server 6.1.x profiles and earlier. In version 6.2, the Failed Event Manager will be available in both WebSphere Process Server and Enterprise Server profiles. The Failed Event Manager is used to replay events or service invocation requests that are made asynchronously between most component types. See the assembly diagram in Figure 4-1 on page 16.

Failed events are created if the AccountRouting component makes an asynchronous call to the SCA Import binding AccountCreationSCAImport and a ServiceRuntimeException is returned.

It is important to note that failed events are not generated in most cases where BPEL is the client in the service interaction. This means that the invocation for 7 and 8 in the use case will not typically result in a failed event. BPEL provides fault handlers and other ways to model for failure. For this reason, if there is a ServiceRuntimeException failure calling JDBCOutboundInterface, the service runtime exception is returned to the BPEL for processing. The error handling strategy for the project should define how runtime exceptions are consistently handled in BPEL.

Important: Failed events are created for asynchronous response messages for the BPEL client if these messages cannot be delivered to the process instance due to an infrastructure failure.

4.3.1 When failed events are created

As stated, failed events are neither created for synchronous invocations nor for two-way business process interactions (Table 4-1). Failed events are generally created when clients use an asynchronous invocation pattern and a ServiceRuntimeException is thrown by the service provider.

Table 4-1 Invocation patterns

Invocation pattern used	Exception type thrown	Failed event created	Comments
Synchronous	ServiceBusinessException	No	Failed events are not created for service runtime exceptions or when using a synchronous pattern.
Synchronous	ServiceRuntimeException	No	Failed events are not created for service runtime exceptions or when using a synchronous pattern.
Asynchronous-one way	ServiceBusinessException	No	By definition, one-way invocations cannot declare faults, meaning that it is impossible to throw a ServiceBusinessException.
Asynchronous-one way	ServiceRuntimeException	Yes	Not applicable
Asynchronous-deferred response	ServiceBusinessException	No	Failed events are not created for service runtime exceptions.
Asynchronous-deferred response	ServiceRuntimeException	Yes	Not applicable
Asynchronous-callback	ServiceBusinessException	No	Failed events are not created for service runtime exceptions.
Asynchronous-callback	ServiceRuntimeException	Yes	Not applicable
BPEL-two way	ServiceRuntimeException	No	Failed events are not created when the source component is a business process.
BPEL-one way	ServiceRuntimeException	Yes	Not applicable

For additional information, see “Managing failed events” in the WebSphere Process Server Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.wps.610.doc/doc/recovery/cadm_failedoverview.html#failedoverview

For additional information about viewing and resubmitting failed events, see “Resubmitting failed events” on page 58.

4.4 Service integration bus destinations

Messages that are waiting to be processed can accumulate in a few service integration bus destinations. For the most part, these destinations are “system” destinations. Messages within these destinations typically are a mixture of three types:

- ▶ Asynchronous requests for processing
- ▶ Asynchronous replies to requests
- ▶ Asynchronous messages that failed deserialization or a function selector resolution

Asynchronous replies: Asynchronous replies can be valid business objects or faults returned as a result of a request.

4.4.1 SCA module destination

Again, we refer to our business case in Figure 4-1 on page 16.

In the situation of the use case presented in this chapter, two SCA module destinations are in the solution:

- ▶ sca/AccountRouting
- ▶ sca/AccountCreation

These destinations are created when the module is deployed to an application server or a cluster. There are rare opportunities for messages to accumulate in these destinations. The accumulation of messages in these locations is a strong indication that there might be a performance problem or an application defect. In either case, be sure to investigate immediately. It is important to monitor the depth of the module destinations (with your chosen IT monitoring solution) because a backup of messages can lead to a system outage or a prolonged recycle time.

We call these *SCA module* destinations because the generated name is the same as the module name with the additional *sca/*. These destinations are pivotal in the functioning of SCA asynchronous invocations (brokering requests and responses). A varying number of additional destinations are generated during application installation on the SCA.SYSTEM bus. However, for the purpose of this discussion, we address the importance of the SCA module destination.

4.4.2 System integration bus retry

As we learned previously, the Failed Event Manager has a built-in retry mechanism with the SCA message-driven bean (MDB). This retry behavior can be controlled by modifying the “Maximum Failed Deliveries” attribute on the module destination.

Note: Typically, there is no reason to adjust this retry capability. The information provided here is for completeness.

Referring again to our business case, several service integration bus destinations are created by SCA to support asynchronous communication. As mentioned previously, one of these destinations is called *sca/AccountRouting*. You can adjust the number of retries (as shown in Figure 4-2) that happen during a *ServiceRuntimeException* of an asynchronous service invocation. You do this by changing the value of the Maximum Failed Deliveries property through the administrative console. However, you cannot set the value less than 2 in modules with a BPEL process. The second delivery is required to return *ServiceRuntimeExceptions* back to the BPEL for processing.

The screenshot displays the configuration page for the *sca/AccountRouting* destination in the IBM WebSphere Administrative Console. The breadcrumb trail at the top reads: *Buses > SCA.SYSTEM.cleanup1Cell01.Bus > Destinations > sca/AccountRouting*. Below the breadcrumb, a description states: "A queue for point-to-point messaging." The "Configuration" tab is selected. The "General Properties" section includes fields for "Identifier" (sca/AccountRouting), "UUID" (7A39E0EE77787F61C25E9056), "Type" (Queue), and "Description" (active:sca/AccountRouting). The "Mediation" section is empty. The "Quality of Service" section has a checked box for "Enable producers to override default reliability", with "Default reliability" and "Maximum reliability" both set to "Assured persistent". The "Default priority" is set to 0. The "Exception destination" section has the "Specify" radio button selected, with the value *WBI.FailedEvent.default.AppTarget* entered. The "Maximum failed deliveries" property is highlighted with a red box and set to 5.

Figure 4-2 Adjusting the *sca/AccountRouting* destination

4.4.3 System exception destinations

The Failed Event Manager is one place where we can look to administer failures. When dealing with imports and exports that are Java™ Message Service (JMS) or enterprise information system (EIS) based, we must consider another important location.

Destinations on the SCA application bus are configured to route failed messages to the system exception destination for that bus. Therefore, if a JMS export picks up a message from the SCA application bus and runs into a rollback situation, the failed message is routed to the system exception destination instead of the WebSphere Business Integration recovery exception destination. This scenario differs from the failed event discussion in that a failure to deserialize a message on the SCA application bus will not result in a failed event.

There is a system exception destination on every bus within the solution. These destinations must be monitored and administered much like the *dead letter queue* that is common to WebSphere MQ infrastructures.

Consider the scenario shown in Figure 4-3. An external JMS client places a message on an inbound queue that is exposed through a JMS export

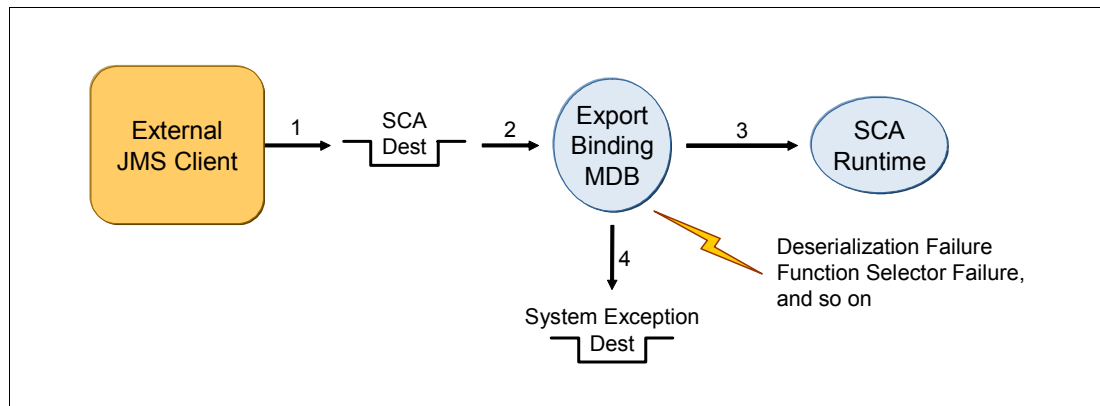


Figure 4-3 External JMS client scenario

The JMS export binding MDB picks up the message for processing. From here, either of the following actions occurs:

- ▶ The JMS export successfully parses the message and determines which operation on the interface to invoke, at which point the message is sent to the SCA run time for processing.
- ▶ The JMS export fails to recognize the message body as a valid business object, or the JMS export binding deserializes the message body but is unable to determine the appropriate operation on the interface to invoke. At this point, the message is placed on the system exception destination for the bus.

Refer again to the use case shown in Figure 4-1 on page 16. We have this type of failure when we try to receive requests from the AccountRoutingJMSExport (1 in Figure 4-1 on page 16). This export is a JMS export, and there is a possibility that events can accumulate on the system exception destination on the SCA.Application.Bus. Use the chosen IT monitoring solution to observe the depth of this destination.

4.4.4 Failed Event Manager and service integration bus destinations

For WebSphere Process Server, the exception destination is set to the WebSphere Process Server recovery exception destination queue. This queue uses the following naming convention:

- ▶ Node name: WPSNode
- ▶ Server name: server1

The recovery exception destination is WBI.FailedEvent.WPSNode.server1.

In general, all the destinations created on the SCA system bus are configured to route failed messages to the recovery exception destination (Figure 4-4).

Buses > SCA.SYSTEM.cleanup1Cell01.Bus > Destinations > sca/AccountRouting

A queue for point-to-point messaging.

Configuration

General Properties

Identifier: sca/AccountRouting

UUID: 7A39E0EE77787F61C25E9056

Type: Queue

Description: active:sca/AccountRouting

Mediation

Quality of Service

☒ Enable producers to override default reliability

Default reliability: Assured persistent

Maximum reliability: Assured persistent

Default priority: 0

Exception destination

☐ None

☐ System

☒ Specify: WBI.FailedEvent.default.AppTarget

Maximum failed deliveries: 5

Figure 4-4 SCA system bus destinations

When a system failure occurs, in addition to capturing the failed message in this exception destination, the WebSphere Process Server recovery feature generates a failed event that represents the system error and stores it in the recovery database as described in 4.3, “Failed Event Manager” on page 17.

4.5 Summary

In summary, WebSphere Process Server and WebSphere ESB provide administrative capabilities are above and beyond the underlying WebSphere Application Server platform (shown in Table 4-2). Ensure that proper measures are made to understand and use these capabilities along with the advice from Chapter 3, “Preventive practices” on page 7.

Table 4-2 Administrative capabilities

Administrative capability	Bundled with WebSphere Process Server?	Summary
Business Process Explorer	Yes	Read, write, edit, and delete access. This is the central place to administer business processes and human tasks.
Failed Event Manager	Yes	Read, edit, and delete access. This is the central place to administer service runtime exceptions and other forms of infrastructure failures. It is only available with WebSphere Process Server profiles.
SIBus Explorer	No	Read and delete access. This is an IBM alphaWorks® tool that can browse service integration destinations. It performs similar functions as WebSphere MQ Explorer.

Note: The number of events or records that can be simultaneously administered by these tools is specific to external factors such as memory allocation, result sets and database tuning, and connection timeouts. You must run tests and set the appropriate thresholds to avoid exceptions (Out Of Memory, TransactionTimeOut).



Recovery scenarios

In this chapter, we provide basic recovery procedures and illustrate those procedures by using several scenarios.

5.1 Scenario types

For the sake of establishing the information provided in this paper, we describe two types of scenarios.

5.1.1 Production environment scenario

The recovery scenarios in 5.4, “Solution recovery scenarios in a production environment” on page 31, provide steps that you must conduct on a production system. The goal and the number of assumptions for a production environment are different than that of a typical test environment. In the production environment, the goal is to process all the requests that have entered the system in a methodical and consistent manner. Data preservation is required for this environment, and all measures must be taken to minimize system unavailability and data loss.

There are important dimensions with a production environment to consider. First, you must consider the type of production topology that is used. When defining the correct topology for your specific project needs, see the article “WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern” on IBM developerWorks® at the following address:

http://www.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html

In addition, insight into the “down” condition is required. For example, if a cluster has multiple cluster members, it is possible that only a single-cluster member must be recovered and that the workload management machinery has already redirected work to running servers. If this is the case, restarting servers should force recovery and that server should join back into the cluster again.

In some high availability situations, there is even the ability to recover failed transactions from one server on another. See “Peer recovery” on page 51 for more information.

Regardless of the situation, recovery of production data requires success on two levels: the system level and the application level.

System recovery

System recovery involves correcting conditions that affect the infrastructure of the solution. WebSphere Process Server solutions rely on fundamental infrastructure requirements. If any of the following types of failures or outages occurs, then these infrastructure services must be corrected and restored prior to WebSphere Process Server system recovery:

- ▶ Power outage
- ▶ Loss of network
- ▶ Database failure
- ▶ Hardware failure

After the prerequisite is re-established, WebSphere Process Server relies on the inherited capabilities from WebSphere to begin application recovery.

Application recovery

Application recovery involves the recovery and resolution of in-flight business transactions. At any given point of a system outage, many transactions are active in the system. We provide

details about how these transactions are used and how they can be resolved in the previous sections.

To have a completely successful application recovery, the applications themselves must observe the preventive practices that are provided. If the applications are not developed with best practices, recovery, and transaction scopes in mind, then application recovery will likely not be completely successful.

A poorly designed or untuned system or application will inevitably leave a percentage of in-flight transactions or processes that remain unresolved after the rest of the application starts processing new events. This statement is true for only WebSphere Process Server and for all J2EE™ applications and application servers.

Untuned system: The term “untuned system” refers to a solution that uses default settings for all components without regard to performance considerations or error handling practices.

Unresolved events can come in different forms such as processes that stay in a running state or failed events that cannot be resubmitted. A post recovery analysis of these events is required to determine the changes that are necessary within the application for a full recovery. These changes should be found during the execution of the comprehensive functional and system test plan.

5.1.2 Test environment scenario

The recovery scenario in 5.5, “Reclaiming the test environment” on page 35, explains the steps to conduct on a test system. The goal and the number of assumptions for a test environment are different than for a production environment. In the test environment, the goal is to recover the system so that new tests can be conducted as soon as possible. Data preservation is not required, and it is assumed that all the requests in the system can be discarded.

Note: Recovering a test environment is not the same as a *recovery* test. Recovery tests use the recommendations that are provided for the production scenarios and should be conducted during the system test phase of the project.

5.2 Basic procedures

For all unplanned system events, a set of basic procedures can be used at the point of identification. For each production scenario, the symptoms that initiate a recovery action can vary. It is important to follow the guidelines for situational analysis and take the corrective action that is unique to the symptoms. Realize that the situational analysis is a read-only activity. While it provides vital information from which to determine the proper recovery actions, it should not change the state of the system under review. It is impossible to predict and provide prescriptive actions for all possible causes of a system outage.

For example, consider the decision tree shown in Figure 5-1 on page 28. There are many broad categories to investigate in the event of an unplanned outage. These broad categories will have subcategories and so on. Defining prescriptive actions for each node and the subsequent node depends on the results of each investigation.

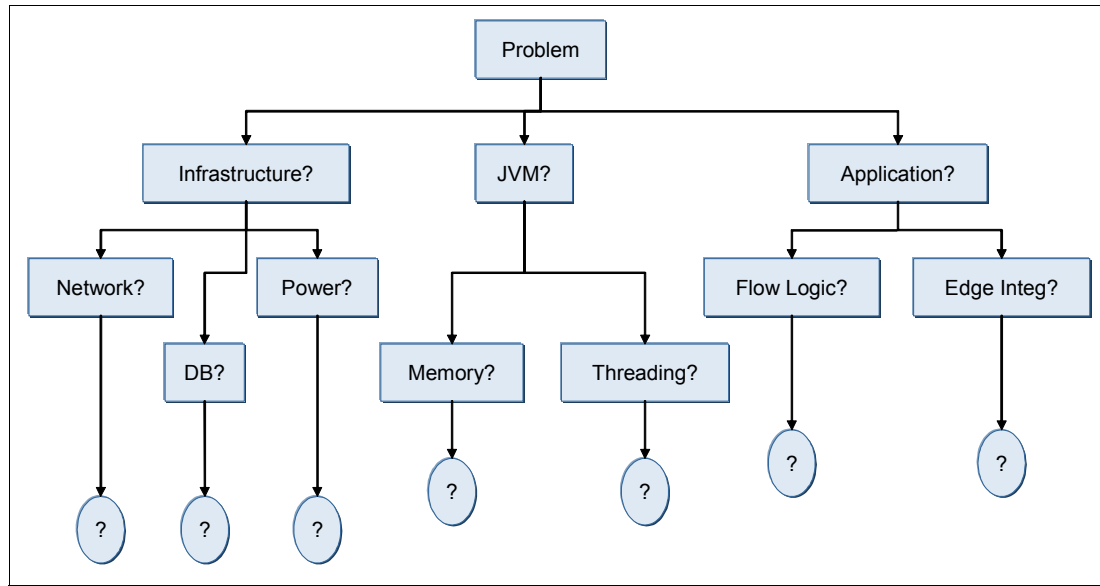


Figure 5-1 Decision tree

5.2.1 Conducting a situational analysis

First, we describe the various situations that initiate a recovery procedure:

- ▶ An abnormal termination or system down has occurred.
A power outage or catastrophic hardware failure has caused the system (all if not most Java virtual machines (JVMs)) to stop.
- ▶ The system is not responsive.
New requests continue to flow into the system, but on the surface, it seems that all processing has stopped.
- ▶ The system is functional, but is severely overloaded.
Transaction timeouts are reported, and there is evidence of an overflow of the planned capacity.
- ▶ The system is unable to initiate a new process instance.
The system is responsive, and the database seems to work correctly. Unfortunately, new process instance creation is failing.

Situational analysis is the cyclical execution of the scientific method. This method entails defining a problem, gathering information, stating a hypothesis, conducting experiments and collecting data, analyzing and interpreting the data, and drawing conclusions. We can apply this methodical approach to our practice of situational analysis.

Gathering information and making observations

When facing an abnormal condition, you must first take the “pulse” of the overall system. Try to understand how much or how little of the system is operational. Also try to understand how much of the system is rendered “out of service” by whatever the external stimuli was that caused this condition.

Run through a predefined set of questions, such as the following questions, to assess the extent of the outage:

► Is this system still performing work?

Consider that the system is still operational, but because of overload, inappropriate tuning, or both, it is not getting work done quickly or is attempting to do work that is failing. The litmus test for these considerations is specific to the nature of the deployed integration solution.

► What special error handling support is built-in to the application?

If the application contains automated retry and various support logic, then the application itself might shield some errors from bubbling up to the IT operator. These conditions must be known and documented for reference by the recovery team.

To help assess the state of the system:

1. Check to see if the server on which WebSphere Process Server runs is at least running. Do you see the process ID (PID) or get a positive feedback from the deployment manager through the administrative console?
2. Check to see if there are locks in the databases or any unusual database traffic. Most databases have facilities to look at locks. Depending on the deployment topology, there might be multiple databases such as the following types:
 - Messaging engine database
 - Business Process Container database
 - WebSphere Process Server common database (failed events and relationship data)
3. Check to see the status of the messaging system. Check for events or messages in the following locations:
 - Business Process Choreographer hold and retention destinations
 - Number of failed events
 - Number of messages on the solutions module destinations.
4. Check to see if the database is working. Verify whether you can perform a simple SELECT operation (on unlocked data) in a reasonable amount of time.
5. Check to see if there are errors in the database log.

If the *database* is not working properly, then recovering the database, so that it can at least release locks and perform simple selects, is vital to system recovery.

If the *messaging system* is not working properly, then recovering the messaging subsystem (see “Recovering the messaging subsystem” on page 76), so that it can at least be viewed and managed, is vital to system recovery.

Note: A bottoms-up approach is not always conclusive. However, chances of successful recovery vary based on these basic activities.

From these basic procedures and health check types of activities, we start to look at some specific situations. Patterns are described, specifics are given, and insights as to what is going on under the covers are provided.

5.3 How to use the provided information

Each of the scenarios consists of three parts:

- ▶ Description of the scenario
- ▶ Assumptions
- ▶ Recovery procedure

5.3.1 Description of the scenario

The “Description of the scenario” section describes the condition of the server for WebSphere Process Server. The following conditions are possible:

- ▶ Abnormal termination
- ▶ System is not responding
- ▶ System is functional but severely overloaded
- ▶ System is responsive but fails to initiate new process instances

5.3.2 Assumptions

In the “Assumptions” section, we discuss what must be preserved and what can be discarded. For example, consider the following questions:

- ▶ Is this a test system where all data will be discarded?
- ▶ Must existing long running processes be saved and allowed to complete?
- ▶ Can the number of new inbound requests be limited?
- ▶ Is it acceptable that new inbound requests receive an error message back indicating that new requests are not being accepted?
- ▶ Are other applications also running that must not be affected?

5.3.3 Recovery procedure

In the “Recovery procedure” section, we include the following information:

- ▶ An overview of the recovery steps that apply
- ▶ The reasons for each step
- ▶ Possible results from each step
- ▶ The flow or what to do next based on the results or previous steps
- ▶ Reference to the appendix for detailed instructions for each step

Many of the same recovery steps are used in more than one scenario. Some of the steps have lengthy descriptions with multiple window captures. Instead of repeating all of the details for each recovery step, we refer to a section in Appendix A, “Troubleshooting tips” on page 39, for details. Therefore, we recommend that you review the recovery steps and become familiar with them, their desired outcomes, or possible questions to answer after each step. Then refer to the appendix for the details.

5.4 Solution recovery scenarios in a production environment

In this section, we discuss the following scenarios for solution recovery in a production environment:

- ▶ Abnormal termination
- ▶ System is not responding
- ▶ System is functional but severely overloaded
- ▶ System is responsive but fails to initiate new process instances

5.4.1 Abnormal termination

In this scenario, a server or set of servers has terminated abnormally. *Abnormal termination* means that the server came down by means other than a normal and successful “stop server.”

At the most fundamental level, the areas of concern and interest when doing recovery are the in-flight work or in-flight transactions. In many cases, restarting the server enables in-flight transactions to be committed or rolled back, all fundamentally based on the WebSphere transaction manager.

Assumptions

In this scenario, we assume that the following conditions are true:

- ▶ Maintaining application and data integrity is required.
- ▶ A single cell clustered deployment environment is used, and only one of the servers in the cluster has abnormally terminated.

Recovery procedure

After an abnormal termination of WebSphere Process Server, unfinished processes might be in a state where they need to be recovered before they can continue. To restart the server in the correct way, a two-step approach must be performed. The recovery procedure entails the following steps:

1. Preserve the data associated to the outage (see “Gathering the essential documents” on page 40):
 - Back up log files that will contain data associated with the root cause.
 - Take a snapshot of the configuration.
2. Start the server in recovery mode (see “Starting the server in recovery mode” on page 51) and wait until server finishes the recovery actions.
3. Restart the server in normal mode (see “Restarting the deployment environment” on page 40). In the rare case that indoubt transactions are not solved automatically and need manual intervention, see “Resolving indoubt transactions” on page 60.
4. Check for failed events (see “Resubmitting failed events” on page 58).
5. Check for inconsistent process instances (see “Business Process Choreographer Explorer tips” on page 68).
6. Check the system integration bus for stuck messages (see “Service Integration Bus Explorer” on page 47).

Tranlog: Do not delete the tranlog, unless advised by the Transaction Manager Level 3 support team. Deleting the tranlog can lead to inconsistency in your production database, which is an unrecoverable situation.

5.4.2 System is not responding

In some cases, the system might seem to stop. It might have some of the following symptoms:

- ▶ New process instances are not initializing.
- ▶ Messages are building up in the system integration bus.
- ▶ If work is completing on the system, it is happening at a degraded level of throughput. In fact, frequent transaction timeouts might be observable in the server logs.

Assumptions

In this scenario, we assume that maintaining application and data integrity is required.

Recovery procedure

Gather the logs, vital system statistics (see “Gathering the essential documents” on page 40), and key configuration data for later evaluation.

Determine the extent of the outage by using the observational practices that were previously described. Do the following checks:

1. Make sure that the database is responsive (see “Reviewing the diagnostic log for DB2” on page 62).
2. Determine if the condition is affecting all members of the cluster.
3. See if the deployment manager, Business Process Choreographer Explorer, or both are responding.
4. Check for information in the Business Process Choreographer hold and retention queues (see “Replaying messages from the retention and hold queues” on page 54).
5. Check for indoubt transactions (see “Resolving indoubt transactions” on page 60).

After you have completed this assessment, choose a course of action. While the recovery procedure here is not entirely prescriptive and ordered, the following heuristics should be useful when recovering these kinds of systems:

- ▶ If the database is not responsive, then address the database.

“Not responsive” might mean that simple queries that return only a few rows do not work. It might mean that table level locks are being held on the database and that determining the source of those locks and addressing the source so as to free the locks is a course of action.

There might be database resources that are exhausted and, therefore, are preventing the database from responding. These can usually be addressed by database administrative actions. Restarting the database (not the database manager) is one possible action and the last resort in many cases. Look closely at the database logs and work with your database administrator to determine the right course of action to get the database back into a responsive condition.

- ▶ If the database appears to be functioning as expected, then perhaps just a single member of a cluster is not responding.

You can determine this by looking at the green/red indicators of the deployment manager for members of a cluster. If only a single member of a cluster needs addressing, then work should be completed successfully by using other members of a cluster. Capturing a Java core dump (see “Capturing a Javacore file” on page 48) and restarting a single member of a cluster is a possible last resort action, if some other resource constraint cannot be freed on a given member of a cluster.

- If the root cause of the stop is not determined based on the recovery analysis and procedures described previously, then more detailed analysis and action might be needed to ensure that this condition does not occur again.

One of the behaviors that might occur when the server is in this condition is transaction timeouts. Recovery from transaction timeouts occurs automatically in many cases. For long-running BPEL processes that call SCA components, transactions that timeout are retried several times. For general SCA-to-SCA calls involving asynchronous interactions that time out, automatic retry through the failed events capabilities occurs. Synchronous clients must have their own handling in place to deal with retry.

In both cases, the automated retry starts after either the transaction timeout period or the thread that is waiting becomes available, whichever takes longer. If the reason for the timeout has not been resolved during this window, then the automated retry facilities will not be effective. See “Resubmitting failed events” on page 58 and “Business Process Choreographer Explorer tips” on page 68 for details about these manual actions.

In addition, keep in mind that, when the “retry” primitive available in mediation modules is used, another automated retry is possible that allows one more option to do an automated retry. In fact, this retry is often done before the calling components (regular SCA or BPEL SCA) get to try an automated retry. Application design here is again key to getting the most from the system functionality that is available.

5.4.3 System is functional but severely overloaded

In this case, we find that the production system is under-resourced for the number of business events that are being processed. There can be several reasons for this situation, but the most likely cause is poor capacity planning or a lack of continual system tuning and monitoring.

Assumptions

In this scenario, we assume that maintaining application and data integrity is required.

Recovery procedure

When we find a system in this state, the following immediate steps are necessary to avoid a full system crash:

1. Gather up the logs for later analysis (see “Gathering the essential documents” on page 40), and attempt to quiesce the system (see “Quiescing the system” on page 52).
2. Evaluate the message resources (see “Tools for evaluating or purging the system” on page 42) to determine if events are being processed.
3. Check for data in the Business Process Choreographer hold and retention queue (see “Replaying messages from the retention and hold queues” on page 54), and replay as necessary.
4. Check for failed events by using the Failed Event Manager (see “Resubmitting failed events” on page 58). Resubmit failed events as necessary.
5. If the server is up and processing messages, then monitor the server until it has completed or caught up. After that, turn on the inbound channel again.

The same transaction timeout discussion from the previous section also applies here. In fact, it is worth noting that a stopped system and a severely overloaded system might be in largely the same condition. A truly stopped server is likely to require a restart.

After the system is recovered, determine exactly what transpired by analyzing the logs. If the root cause of the system overload or failure is not known, then there is a strong possibility that the situation will happen again.

In general, any failure such as this must be well understood. It is possible that additional tuning, additional capacity, more error handling in the application design, or some other measure is necessary to prevent recurring failures.

5.4.4 System is responsive but fails to initiate new process instances

This scenario is similar to the previous scenario. However, the difference is that most of the system is functional as in the following examples:

- ▶ Exports are receiving business events.
- ▶ Non-BPEL components and modules are functioning correctly.

Assumptions

In this scenario, we assume that maintaining application and data integrity is required.

Recovery procedure

The following steps might be necessary to get the process engine to start handling events again:

1. Gather log files, key system vital statistics, and configuration information (see “Gathering the essential documents” on page 40).
2. Evaluate the message resources (see “Gathering information and making observations” on page 28) to determine if events are being processed.
3. Check and make sure that the process template has started, which you can do in the administrative console (Figure 5-2).

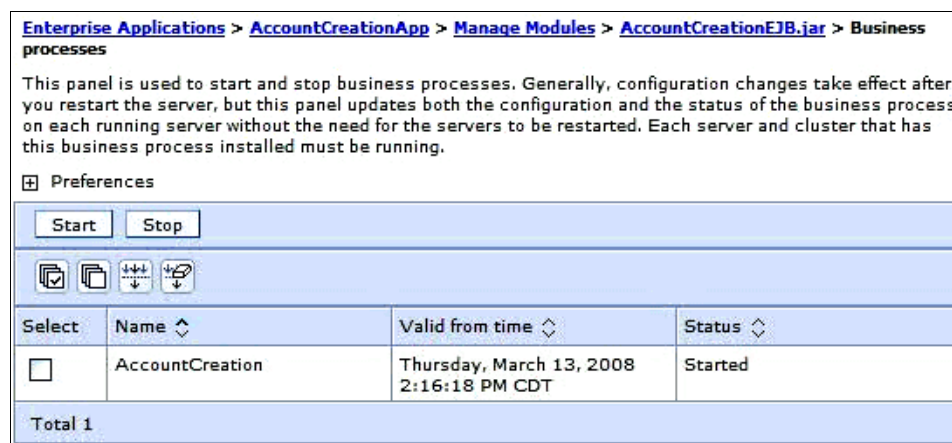


Figure 5-2 Starting or stopping business processes

4. Enable the Business Process Choreographer trace.
5. Log into the Business Process Choreographer Explorer and try to manually create a process instance.
6. If you cannot manually create a process instance, then restart the Business Process Choreographer cluster.
7. After the restart, if you are still unable to process instances, then immediately contact IBM support and initiate your corporate *production down* procedure.

5.5 Reclaiming the test environment

Several approaches are possible to reclaim a test system. When data preservation is not required, the door opens to various techniques including the following methods:

- ▶ Virtual images

The usage of virtual images has grown in popularity with the penetration of VMWare into the marketplace. If there is a need to *reset* the test system, virtual machines can be redeployed from the testing archives.

- ▶ System snapshots

Another common technique is to take cold, stable directory, and database backups. These backups can then be used to return a test environment to a pre-test state.

Important: Both of these approaches are used in test only. The use of these techniques in production results in the loss of stateful process and system information.

5.5.1 Manually reclaiming the test environment

The goal is to reset the system to a pre-test state by stopping and deleting in-flight work to minimize the delay between test cycles.

Assumptions

The following conditions are assumed in this scenario:

- ▶ Applications must be maintained.
- ▶ Server start and restarts are permitted.

Procedure

To reclaim the test environment:

1. Stop the test clients from sending in more events.

If the test client is an isolated application without processes or artifacts needed by other application modules, then stop the test clients from sending in more events by stopping the application. You can accomplish this by using the administrative console or scripting.

2. Stop the Business Process template. Process templates can be stopped by using the administrative console or the BPCTemplates.jacl script (Figure 5-3).

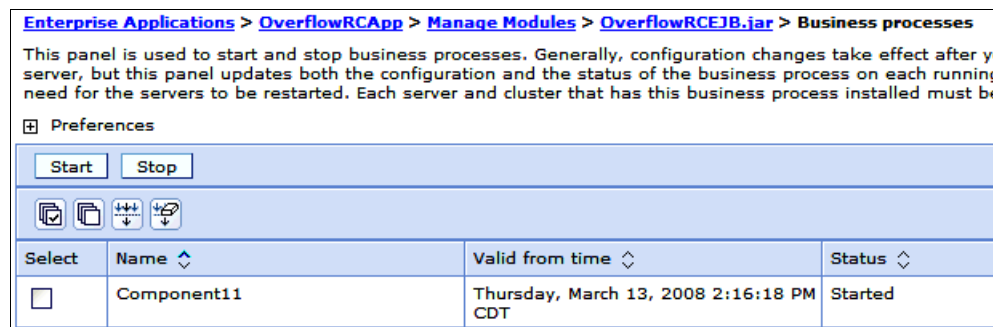


Figure 5-3 Stopping the business process template

3. Stop processing queued events (see “Stopping the processing of asynchronous events” on page 52).

4. Take action to correct database problems (see “Reviewing the diagnostic log for DB2” on page 62) if any are being encountered.
5. Increase the number of secondary log files if transaction logs are full.
6. Terminate running business process instances.

You can use either the Business Process Choreographer Explorer or Business Flow Manager API to do this. It might be necessary to control the number of instances that are being terminated at one time to avoid transaction timeouts or OutOfMemory exceptions. A good rule of thumb is to delete only tens of instances at a time, not hundreds or thousands, at least as a conservative starting point. Deleting instances means that all instances requested are deleted in a single transaction. Therefore, deleting too many can cause a system already under stress to time out.

The Business Process Choreographer Explorer provides capabilities to control the number of instances that are shown on a *page*.

7. Delete the finished and terminated business process instances.

You can use the Business Process Choreographer Explorer or the `deleteCompletedProcessInstances.py` (see “Deleting the completed process instances” on page 62) to do this. It might be necessary to control the number of instances that are being terminated at one time to avoid transaction timeouts. See “Deleting the completed process instances” on page 62 for additional information.

8. Clean out the Failed Event Manager.

Delete the events or use “Clear all on server” (Figure 5-4).

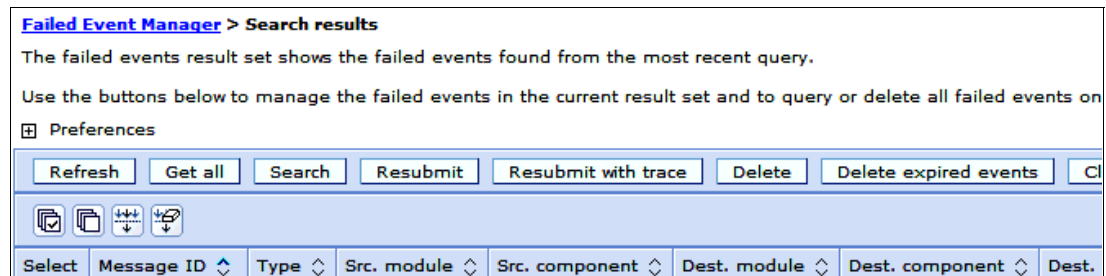


Figure 5-4 Deleting events

9. Clean out the SCA queues for the application.

Delete the messages in the SCA module destinations (Figure 5-5).



Figure 5-5 Deleting messages on the SCA module destinations

10. Clean out the Business Process Choreographer queues.

Delete messages from BPEHldQueue, BPEIntQueue, and BPERetQueue (Figure 5-6).

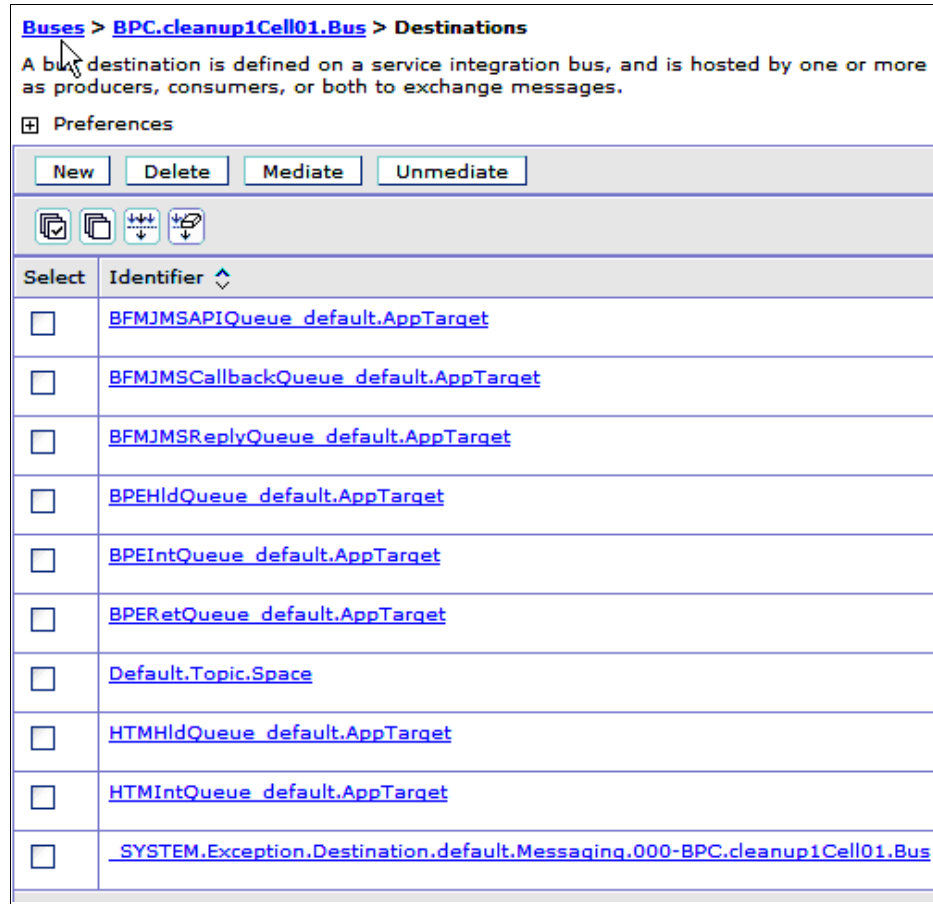


Figure 5-6 Clearing out the Business Process Choreographer queues

11. Stop and then start the messaging and application servers (Figure 5-7).

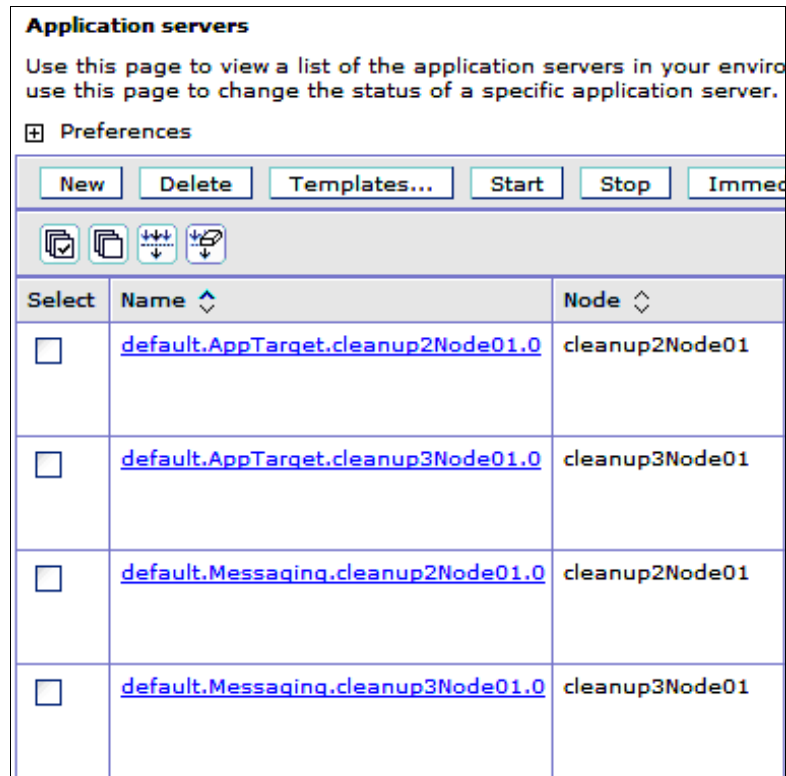


Figure 5-7 Stopping and starting the application servers

12. Start the Business Process template.

At this point, you can test the system by executing a single event through the application.



A

Troubleshooting tips

In this appendix, we provide more information about troubleshooting.

Gathering the essential documents

For suggested MustGather information regarding a variety of situations, see “MustGather: Read first for WebSphere Process Server for Version 6” at the following Web address:

http://www-1.ibm.com/support/docview.wss?rs=2307&context=SSQH9M&dc=DB520&dc=D600&dc=DB530&dc=D700&dc=DB500&dc=DB540&dc=DB510&dc=DB550&q1=Mustgather&uid=swg21239895&loc=en_US&cs=utf-8&lang=en

The MustGather information includes the configuration tree and relevant logs for further analysis and inspection of what has happened on the system. IBM Support Assistant can help you do this. See “IBM Support Assistant” on page 74.

Restarting the deployment environment

The restart procedure of the deployment environment depends on the type of topology that is selected for the business requirements. In a given deployment environment, a number of servers are running as Java virtual machine (JVM™) processes, including the following general types of servers:

- ▶ Messaging servers, which are responsible for providing the service integration bus messaging infrastructure
- ▶ WebSphere Enterprise Service Bus servers, whose profiles are capable of only hosting and running mediation modules
- ▶ WebSphere Process Server servers, whose profiles are capable of hosting and running all module types

This profile hosts the Business Process Choreographer.

- ▶ Support servers, which are responsible for providing support and monitoring services such as Common Event Infrastructure (CEI)

For more advanced and highly available environments, you will find these servers in clusters and distributed across physical resources. See the article “WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern,” at the following address, for descriptions about topology and the associated decision making process:

http://www.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html

A general model for starting servers is to start the messaging servers first, then the support servers, and then the application servers (WebSphere Enterprise Service Bus (ESB) and WebSphere Process Server). Each application architecture might have specific dependencies between application components that must be taken into consideration.

Shutdown happens inverse to the startup procedure, starting with the application server clusters and ending with shutting down the messaging infrastructure after it has had time to quiesce and process any in-flight work.

Using Business Process Choreographer's predefined views

The Business Process Choreographer database, which is the business process engine database (BPEDB), is used to store the following information:

- ▶ Process and task templates
- ▶ Process and task instances

Several predefined views on the BPEDB are described in “Database views for Business Process Choreographer” in the WebSphere Process Server Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.612.doc/doc/bpc/r6bpc_dbviews.html

In the absence of any monitoring products, you can use these views to obtain a glimpse of the work that is being performed by Business Process Choreographer.

The `PROCESS_INSTANCE` view can be used to determine the state of all known long running process instances. Repeatedly executing queries over time shows the rate at which process instance work is being performed.

A query, such as the following example, provides a breakdown of all process instances by state as shown in Figure A-1:

```
SELECT COUNT(*), STATE, TEMPLATE_NAME FROM <schema name>.PROCESS_INSTANCE GROUP BY  
TEMPLATE_NAME, STATE;
```

Commands	Query Results	Access Plan
Edits to these results are performed as searched UPDATES are editing.		
1	STATE	TEMPLATE_NAME
	13	2 AccountCreation
	8938	3 AccountCreation

Process state codes:
STATE_READY (1)
STATE_RUNNING (2)
STATE_FINISHED (3)
STATE_COMPENSATING (4)
STATE_INDOUBT (10)
STATE_FAILED (5)
STATE_TERMINATED (6)
STATE_COMPENSATED (7)
STATE_COMPENSATION_FAILED (12)
STATE_TERMINATING (8)
STATE_FAILING (9)
STATE_SUSPENDED (11)

Figure A-1 Query results

If the process is defined to “automatically delete after completion,” you will not see instances in the Finished state. Setting a process to “save on completion” solely to have a count of finished instances is not considered a best practice.

Figure A-2 shows the process deletion settings in WebSphere Integration Developer.

Details	<input checked="" type="checkbox"/> Process is long-running
Join Behavior	Automatically delete the process after completion: No
Imports	<input type="checkbox"/> Ignore missing data
Server	Compensation sphere: <input type="radio"/> Supports <input type="radio"/> Required
Administration	Autonomy: <input checked="" type="radio"/> Peer <input type="radio"/> Child
Java Imports	On successful completion: No

Figure A-2 Process deletion settings in WebSphere Integration Developer

Tools for evaluating or purging the system

In this section, we discuss the tools that are available for evaluating or purging the system.

Process cleanup service

A publicly available tool uses the WebSphere Scheduler to regularly clean up completed process instances. You can find information about this tool at the following Web address:

<http://www.ibm.com/support/docview.wss?uid=swg27007816>

One important limitation is that this tool does not work if security is enabled. A more robust version of this tool will be provided with the WebSphere Process Server product in a future release.

See “Deleting the completed process instances” on page 62 for information about an alternative tool for deleting process instances.

Viewing the service integration bus by using the administrative console

More information: Service Component Architecture (SCA) module destinations are described in 4.4.1, “SCA module destination” on page 19. You might want to review that section before continuing.

You can use the administrative console to view messages on the service integration bus. This can be particularly useful to determine if messages are accumulating on the SCA module destinations. To view the service integration bus:

1. From the administrative console, expand **Service integration**, and then select **Buses** (Figure A-3).

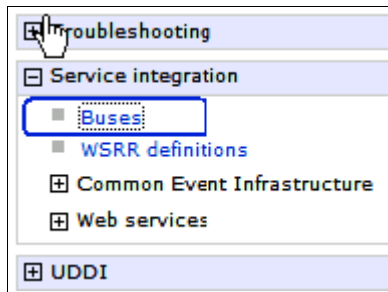


Figure A-3 Service integration buses

2. Select **SCA.System.<cell name>.Bus** (Figure A-4).

Buses

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

⊕ Preferences

New Delete

⊞ ⊞ ⊞ ⊞

Select	Name ↕	Description ↕	Security ↕
<input type="checkbox"/>	BPC.cleanup1Cell01.Bus	Messaging bus for Process Choreographer	Enabled
<input type="checkbox"/>	CommonEventInfrastructure.Bus	CommonEventInfrastructure Bus	Enabled
<input type="checkbox"/>	SCA.APPLICATION.cleanup1Cell01.Bus	Messaging bus for Service	Enabled
<input type="checkbox"/>	SCA.SYSTEM.cleanup1Cell01.Bus	Messaging bus for Service	Enabled

Total 4

Figure A-4 Selecting destinations

3. Select **Destinations** (Figure A-4).

Buses > SCA.SYSTEM.cleanup1Cell01.Bus

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Configuration Local Topology

General Properties	Topology
Name <input type="text" value="SCA.SYSTEM.cleanup1Cell01.Bus"/>	■ Bus members
UUID <input type="text" value="88D12C35D81C8E5C"/>	■ Messaging engines
Description <input type="text" value="Messaging bus for Service"/>	■ Foreign buses
	Destination resources
	■ Destinations
	■ Mediations

Figure A-5 Selecting the SCA.System.<cell name>.Bus

4. Look at the destinations named `sca/<Module Name>` first.

Referring to 4.1, “Use case” on page 16, there were two SCA module destinations, which are shown in Figure A-6:

- `sca/AccountRouting`
- `sca/AccountCreation`

Click the **sca/AccountCreation** link to view the general properties page.

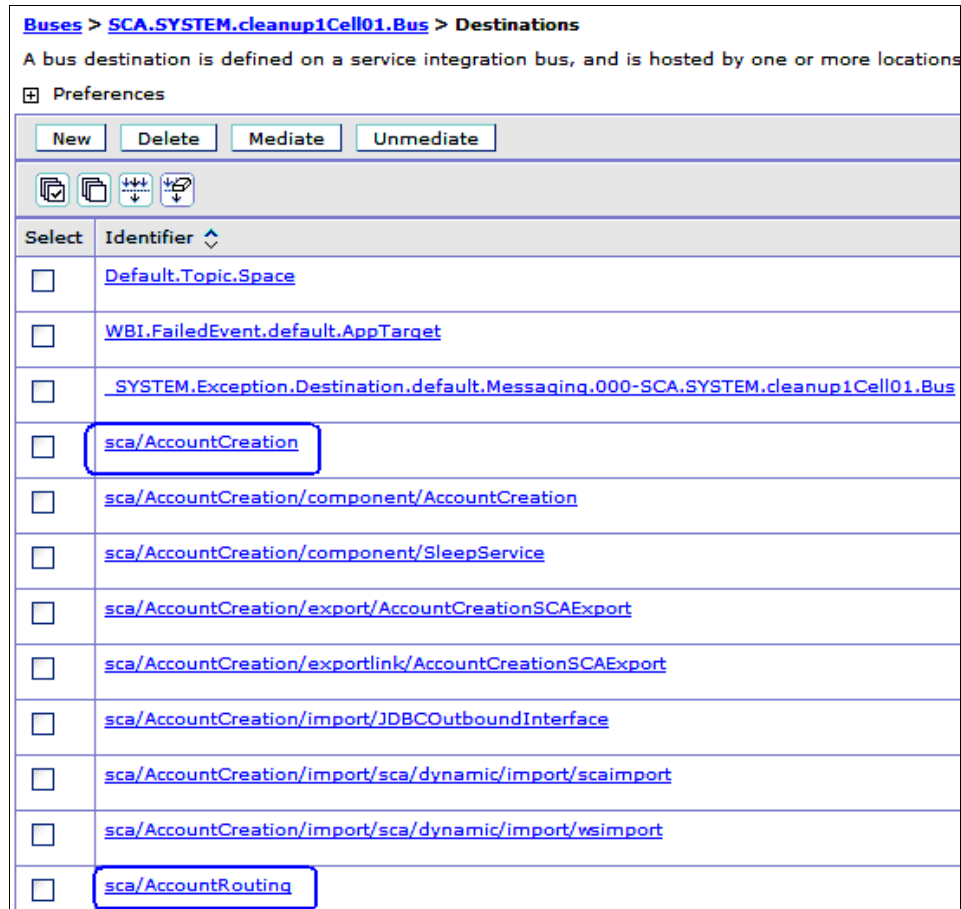


Figure A-6 Looking at the two SCA module destinations

5. On the General Properties page (Figure A-7), select the **Queue Points** link from this page.

Buses > [SCA.SYSTEM.cleanup1Cell01.Bus](#) > [Destinations](#) > [sca/AccountCreation](#)

A queue for point-to-point messaging.

Configuration

General Properties

Identifier

UUID

Type

Description

Message points

- ☒ [Queue points](#)
- ☐ [Mediation points](#)

Additional Properties

- ☐ [Context properties](#)
- ☐ [Mediation execution points](#)

Figure A-7 Viewing the sca/AccountCreation properties

6. Select the link for the message point (Figure A-8).

Buses > [SCA.SYSTEM.cleanup1Cell01.Bus](#) > [Destinations](#) > [sca/AccountCreation](#) > [Queue points](#)

The message point for a queue, for point-to-point messaging.

☒ Preferences

☐ Queue points

☐ Mediation points

Identifier

Total 1

Figure A-8 Working with the sca/AccountCreation message point

7. Click the **Runtime** tab (Figure A-9).

The screenshot shows the IBM WebSphere console interface. At the top, a breadcrumb trail reads: [Buses](#) > [SCA.SYSTEM.cleanup1Cell01.Bus](#) > [Destinations](#) > [sca/AccountCreation](#) > [Queue points](#) > [sca/AccountCreation@default.Messaging.000-SCA.SYSTEM.cleanup1Cell01.Bus](#). Below the trail, a text label states: "The message point for a queue, for point-to-point messaging." There are two tabs: "Configuration" and "Runtime", with "Runtime" being the active tab. The "General Properties" section contains several fields: "Identifier" with value "sca/AccountCreation@default.Messaging.000-SCA.SYSTEM.cleanup1Cell01.Bus", "UUID" with value "EC5684EA089E54D3", "Destination type" with value "QUEUE", "High message threshold" with value "50000" and unit "messages", a checked "Send allowed" checkbox, and "Target UUID" with value "5D2AB86F4EDEC81E01F34714".

Figure A-9 The Runtime tab

8. On the Runtime page, which shows the current *message depth* and the threshold, click the **Messages** link (Figure A-10) to view the message contents.

This screenshot shows the same console page as Figure A-9, but with additional elements. A "Refresh" button is located at the top of the Runtime section. The "General Properties" section now includes a "Run-time ID" field with the value "5D2AB86F4EDEC81E01F34714_QUEUE_28000008" and a "Current message depth" field with the value "0". The "Additional Properties" section on the right contains two links: "Messages" and "Known remote queue points". The "Messages" link is highlighted with a red box. The "High message threshold" field is also highlighted with a red box. An "OK" button is located at the bottom left of the Runtime section.

Figure A-10 Viewing the message depth and threshold and clicking the Messages link

Note: It is a good idea to periodically view the messages (Figure A-11) and determine if any messages have become locked for an extended duration of time, which might indicate that there are indoubt transactions. Ideally, use an appropriate IT monitoring tool and set alert thresholds for these destinations. The threshold value is established during the performance test phase for the application.

Buses > SCA.SYSTEM.cleanup1Cell01.Bus > default.Messaging.000-SCA.SYSTEM.cleanup1Cell01.Bus > Queue points > sca/AccountCreation@default.Messaging.000-SCA.SYSTEM.cleanup1Cell01.Bus > Messages

The messages on the message point.

☐ Preferences

Maximum rows
500

☐ Retain filter criteria.

Apply Reset

Delete Delete all Refresh

Select Position Identifier State Transaction ID

None

Total 0

Figure A-11 Viewing messages to determine if any are locked

Attention: Never delete messages on a production system unless explicitly directed to do so by the SCA L3 team.

For more information about indoubt transactions, see “Resolving indoubt transactions” on page 60.

You can find an additional procedure to resolve indoubt transactions in the article “Resolving indoubt transactions” in the WebSphere Application Server Network Deployment Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/tasks/tjm0165_.html

Service Integration Bus Explorer

The Service Integration Bus Explorer is a stand-alone tool that provides navigation and monitoring of the messaging components of a service integration bus (SIBus). It is an alternative to the service integration bus tooling in the administrative console.

You can find additional information and a download on IBM alphaWorks at the following address:

<http://www.alphaworks.ibm.com/tech/sibexplorer>

Figure A-12 shows the following SCA module destinations, from the previous task, and their current message counts and thresholds:

- ▶ sca/AccountRouting
- ▶ sca/AccountCreation

The administrative console does not provide all this information on a single page. The Service Integration Bus Explorer can also show the contents of a queue point and the contents of a message.

Attention: Never delete messages on a production system unless you are explicitly directed to do so by the SCA L3 team.

Queue Point Name	Current Depth	State	High Message Threshold	Send Allowed
sca/AccountCreation	0	ACTIVE	50000	true
sca/AccountCreation/component/...	0	ACTIVE	50000	true
sca/AccountCreation/export/Acc...	0	ACTIVE	50000	true
sca/AccountCreation/exportlink/...	0	ACTIVE	50000	true
sca/AccountCreation/import/JDB...	0	ACTIVE	50000	true
sca/AccountCreation/import/sca/...	0	ACTIVE	50000	true
sca/AccountCreation/import/sca/...	0	ACTIVE	50000	true
sca/AccountRouting	0	ACTIVE	50000	true
sca/AccountRouting/component/...	0	ACTIVE	50000	true
sca/AccountRouting/export/Acco...	0	ACTIVE	50000	true
sca/AccountRouting/exportlink/A...	0	ACTIVE	50000	true
sca/AccountRouting/import/Acco...	0	ACTIVE	50000	true
sca/AccountRouting/import/sca/d...	0	ACTIVE	50000	true
sca/AccountRouting/import/sca/d...	0	ACTIVE	50000	true
sca/AccountRouting/importlink/Ac...	0	ACTIVE	50000	true
sca/BEMIE_default_AppTarget	0	ACTIVE	50000	true

Figure A-12 View of the current message counts for the SCA modules

Capturing a Javacore file

A *Javacore file* is a snapshot of a running Java process. Several methods exist that can be used to capture a Javacore file from an IBM Java Development Kit (JDK™) and thread dumps for non-IBM JDKs. We present two of these methods as follows. The first method for Windows® systems is the simplest approach for Windows. Use this method if possible. The second method for UNIX® is the simplest approach for UNIX.

1. Use **wsadmin** to produce a Javacore file in the <PROFILE_DIR> directory:

- For Windows, type the following command:

```
<PROFILE_DIR>\bin\wsadmin.bat [-host host_name] [-port port_number] [-user
userid -password password] -c "$AdminControl invoke [$AdminControl
queryNames WebSphere:name=JVM,process=server1,*] dumpThreads"
```

- For UNIX (IBM JDKs), type the following command:

```
<PROFILE_DIR>/bin/wsadmin.sh[-host host_name] [-port port_number] [-user
userid -password password] -c "\$AdminControl invoke [\$AdminControl
queryNames WebSphere:name=JVM,process=server1,*] dumpThreads"
```

Note: The braces ([]) around the **AdminControl queryNames** command are part of the actual command. They are not used to indicate optional parameters, which is the case for the braces around host, port, and user. You might need to change the process name, **server1**, to fit your configuration.

2. Send a signal to the server process:

– Windows

You must use a launch script to start the server process so that the signal can be passed to the process. This requires a special setup prior to starting the server, which entails entering the following commands in the order shown:

- i. Generate a launch script with the **startServer** command, instead of launching the server process directly:

```
<PROFILE_DIR>\bin\startServer.bat server1 -script SERVER1.bat
```

The launch script name is an optional argument. If you do not supply the launch script name, the default script file name is **start_server** based on the server name passed as the first argument to the **startServer** command. The server process starts in a command window.

- ii. Check the logs to verify that the server has successfully started since the intermediate JVM process which usually starts the server process is not used.

Start the server by using the **SERVER1.bat** file in order for the <CTRL><BREAK> in the next step to correctly generate the Javacore:

SERVER1.bat

- iii. Enter the following command in the command window where the server process is running:

<CTRL><BREAK>

A Javacore file is produced.

– UNIX (all JDKs)

For UNIX, enter the following command:

```
kill -3 <pid>
```

Here, <pid> is the process ID of the WebSphere Process Server. For IBM JDKs, a Javacore file is produced in the <PROFILE_DIR> directory. For non-IBM JDKs, a thread dump is written to the **native_stdout.log**.

An alternative method for dumping a Windows core file is to use the **jvmdump** utility. This utility does not require special setup prior to starting the server. However, it requires a special executable from the JVM team. The advantage of this method is that additional information can be obtained about native code that is being executed within JVM. The format of the dump differs from the IBM Javacore files.

You can request the **jvmdump.exe** program by sending an e-mail message to jvmcookbook@uk.ibm.com.

The **jvmdump** program takes a single parameter that is the PID of a process as shown in the following example:

```
jvmdump.exe <PID>
```

When **jvmdump** is run, the program generates a minidump that you can analyze by using the dump formatter, which consists of two utilities: **jextract** and **jdmview**.

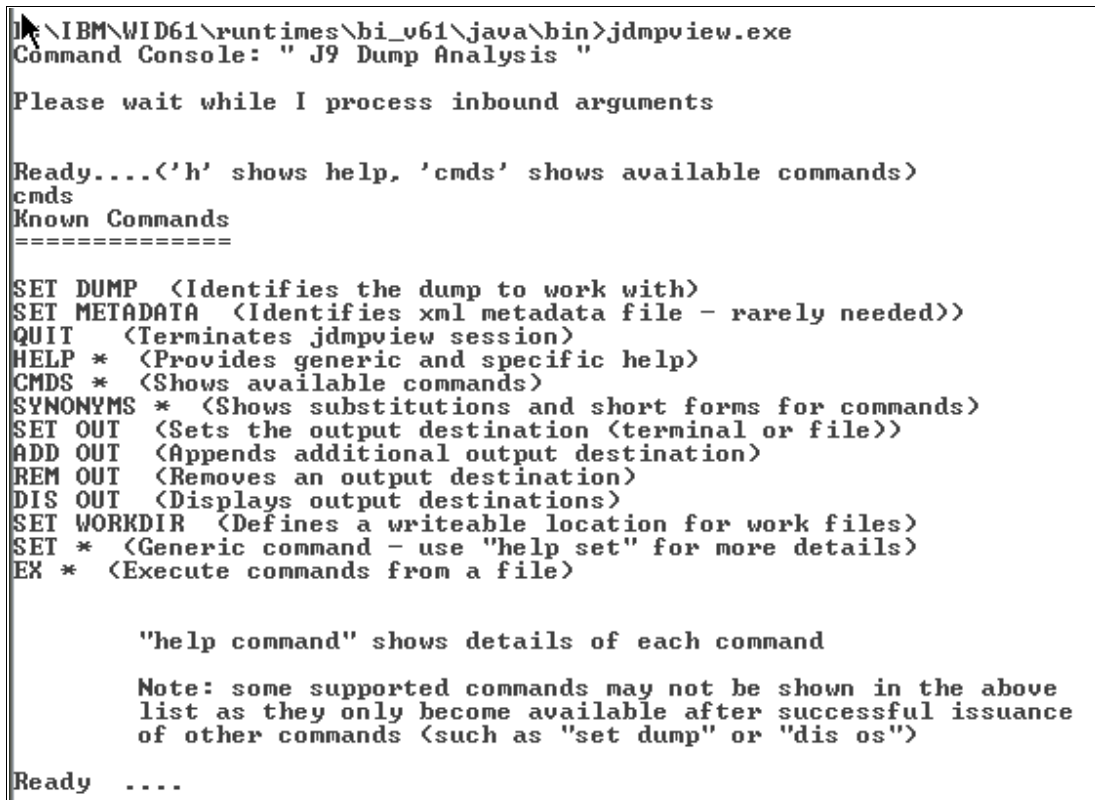
The **jextract** utility produces an XML file with details about useful JVM internal information that you can use together with the original dump to diagnose problems. To invoke **jextract**, at a command prompt, type the following command:

```
<WAS_HOME>\java\jre\bin\jextract.exe <core.name.dmp>
```

jdumpview is a launcher for the main method of the Java class J9JVMConsole that is in the sdk/jre/lib/ext/jdumpview.jar file. To invoke **jdumpview**, from a command prompt, type the following command:

```
<WAS_HOME>\java\bin\jdumpview.exe
```

When **jdumpview** starts, it displays the message Ready..., which means that you can start invoking commands on **jdumpview**. Figure 5-8 shows the options for **jdumpview**.



```
IBM\WID61\runtimes\bi_u61\java\bin>jdumpview.exe
Command Console: " J9 Dump Analysis "

Please wait while I process inbound arguments

Ready....('h' shows help, 'cmds' shows available commands)
cmds
Known Commands
=====
SET DUMP <Identifies the dump to work with>
SET METADATA <Identifies xml metadata file - rarely needed>
QUIT <Terminates jdumpview session>
HELP * <Provides generic and specific help>
CMDS * <Shows available commands>
SYNONYMS * <Shows substitutions and short forms for commands>
SET OUT <Sets the output destination (terminal or file)>
ADD OUT <Appends additional output destination>
REM OUT <Removes an output destination>
DIS OUT <Displays output destinations>
SET WORKDIR <Defines a writeable location for work files>
SET * <Generic command - use "help set" for more details>
EX * <Execute commands from a file>

      "help command" shows details of each command

      Note: some supported commands may not be shown in the above
      list as they only become available after successful issuance
      of other commands (such as "set dump" or "dis os")

Ready ....
```

Figure 5-8 The **jdumpview.exe** options

The following **jdumpview** options are commonly used to view the dump:

- set dump <core.name.dmp>.zip
- display thread

Displays the current executing thread at the time of the dump.

- display thread *

Displays all of the threads from the dump.

For more details about the **jdumpview** utility, see “Using the dump viewer, **jdumpview**” in the *IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 5.0 Diagnostics Guide* at the following address:

<http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/diagnosis/diag50.pdf>

Starting the server in recovery mode

When an application server instance with active transactions in progress restarts after a failure, the transaction service uses recovery logs to complete the recovery process. These logs, which each transactional resource maintains, are used to rerun any indoubt transactions and return the overall system to a self-consistent state.

This recovery process begins as soon as all of the necessary subsystems within the application server are available during a server startup. If the application server is not restarted in recovery mode, the application server can start accepting new work as soon as the server is ready, which might occur before the recovery work has completed. This might be acceptable in many cases, but we provide a more conservative option here. To be clear, recovery runs on a server restart even if the server is started in “normal” start model.

For more information, see “Restarting an application server in recovery mode” in the WebSphere Process Server Information Center at the following address:

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

Peer recovery

Peer recovery is recovery as performed by another member of the same cluster and can be initiated manually or automatically. The article “IBM WebSphere Developer Technical Journal: Transactional high availability and deployment considerations in WebSphere Application Server V6” discusses the requirements, setup, and management of both automated and manual peer recovery available. You can find this article in developerWorks at the following address:

http://www.ibm.com/developerworks/websphere/techjournal/0504_beaven/0504_beaven.html

In addition, see the following documents for further information:

- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
<http://www.redbooks.ibm.com/redpieces/abstracts/sg246392.html>
- ▶ “Configuring transaction properties for peer recovery” in the WebSphere Application Server Network Deployment Information Center
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tjta_cfgpeer.html
- ▶ “Managing manual peer recovery of the transaction service” in the WebSphere Application Server Network Deployment Information Center
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tjta_managing.html

Quiescing the system

For most of our scenarios, our goal is to stop the inbound requests and allow the work that is currently in the system to “settle.” The settling allows the solution to recover from the instability characteristics such as the following examples:

- ▶ Transaction timeouts
- ▶ Stopped thread notification

Many of these error conditions can be attributed to resource contention.

Quiescing the system is recommended in cases where servers are still up, but might be overwhelmed. Given the loosely coupled nature of many of the solutions, work can be queuing up internally, in a number of places. The sooner this is halted and contained, the more quickly the rest of recovery can begin and the shorter the duration is of the recovery period.

To quickly reduce the amount of processing in the overloaded system, one option is to stop the processing of asynchronous events.

Stopping the processing of asynchronous events

You can use the **stopMDB.py** script (Example A-1) to stop the processing of message-driven beans (MDBs). This script is also included in the download materials for this paper. See Appendix B, “Additional material” on page 77, for more information.

Specifically, the script can be used to stop the processing of the SCA MDBs that are responsible for routing asynchronous requests and responses. The script is based on the J2CMessagingEndpoint MBean that was introduced with APAR PK49507 and is documented at the following a Web address:

<http://www-01.ibm.com/support/docview.wss?uid=swg1PK49507>

Example: A-1 stopMDB.py script

```
### ***YOU MUST TAYLOR THIS SCRIPT TO YOUR ENVIRONMENT***

### run as:
### wsadmin.sh -username <username> -password <password> -f stopMDB.py

### For example:
### wsadmin .sh -username admin -password admin -f stopMDB.py

### cell name - customize for your environment
cn='cleanup1Cell01'
### nodes in cell - customize for your environment
nn=['cleanup2Node01', 'cleanup3Node01']
### servers on nodes - customize for your environment
sn=['default.AppTarget.cleanup2Node01.0', 'default.AppTarget.cleanup3Node01.0']

### customize range for your environment
for i in range(0,2):
    jme = AdminControl.queryNames('cell=' + cn + ',node=' + nn[i] +
                                  ',type=J2CMessageEndpoint,process=' + sn[i] + ',*')
    print "check status of J2CMessageEndpoints.  1 = activated, 2 = deactivated, 3 = stopped"
    print
```

```

jme1 = jme.splitlines()
for endpoint in jme1:
    ### customize name to match your SCA module name
    if endpoint.count('AccountCreation') > 0:
        print AdminControl.invoke(endpoint, 'getActivationProperties')
        print "MDB Status before change = " + AdminControl.invoke(endpoint, 'getStatus')
        ### Change the operation to 'pause' to stop the MDB
        ### Change the operation to 'resume' to restart the MDB
        print AdminControl.invoke(endpoint, 'pause')
        print "MDB Status after change = " + AdminControl.invoke(endpoint, 'getStatus')
    print

```

Important: Stopping asynchronous events on a per module basis is not the correct solution for every recovery situation. Be sure to raise a PMR and allow IBM support to provide guidance as needed.

Export bindings

To completely quiesce a system, we must consider the different types of request invocations that are supported by the available Export bindings. Figure A-13 represents the type of SCA invocation pattern that is used for the different Export binding.

Export binding	Operation type	Performance attributes > Interaction style	Invocation style
EIS	One-way	Asynchronous	Asynchronous (default)
		Synchronous	Synchronous
	Request-response	Any value	Synchronous
Export binding	Operation type	Invocation style	
HTTP	One-way or request-response	Synchronous	
MQ or MQ JMS	One-way	Asynchronous	
	Request-response	Asynchronous with callback	
SCA JMS	One-way	Asynchronous	
	Request-response	Asynchronous with callback	
Web services (SOAP/HTTP) or (SOAP/JMS)	One-way or request-response	Synchronous	

Figure A-13 SCA invocation table

Depending on the application and the topology used, a variety of techniques can be used to quiesce synchronous communication.

Recommendation: You must create a quiescing strategy your WebSphere Process Server implementation based on the unique characteristics of the export that is used and the topology.

Replaying messages from the retention and hold queues

The administration of existing messages on the retention queue, the hold queue, or both and potentially replaying these messages can be part of a recovery procedure. Checking and replaying messages can be done by using scripting or the administrative console.

Refer to “Business process engine” on page 17 for additional information about the business process engine queues (internal, hold, and retention).

For information about `queryNumberOfFailedMessages` and the `replayFailedMessages` scripts, see “Querying and replaying failed messages, using administrative scripts” in the WebSphere Process Server Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t4adreplay.html>

In the procedure that follows, we explain the following tasks:

- ▶ Finding the business process engine’s retention and hold queues
- ▶ Determining the current number of messages
- ▶ Replaying the process messages

1. From the WebSphere Application Server administrative console, in the left pane, select **Servers** → **Application servers**. In the right pane, select **your server name**. Then under Business Integration, expand **Business Process Choreographer** and click **Business Flow Manager** (Figure A-14).

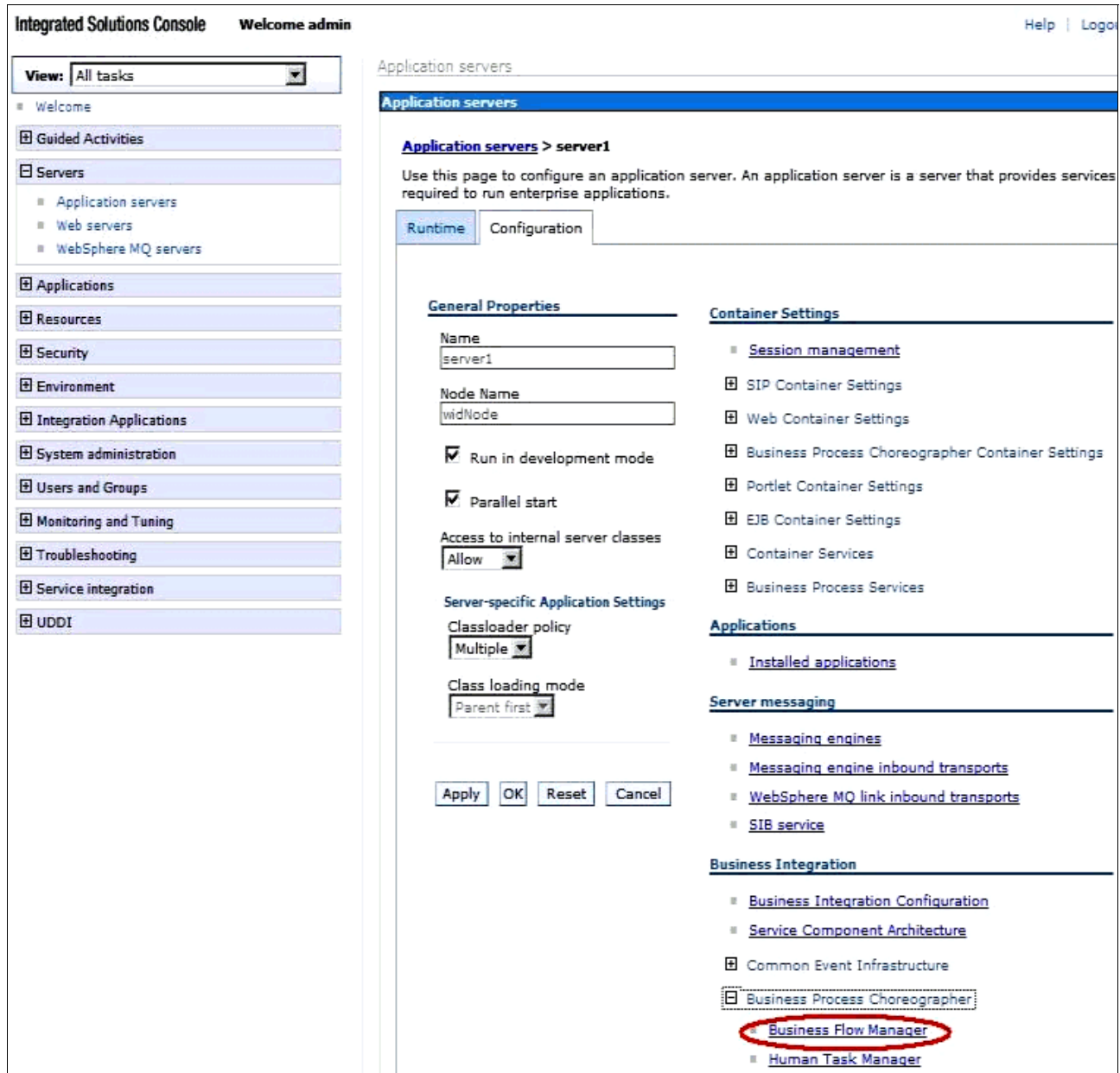


Figure A-14 Selecting Business Flow Manager

2. On the Configuration page (Figure A-15), review the retention queue name, the hold queue name, the retry count, and the queue limits. Click the **Runtime** tab to work with items in the retention and hold queues.

[Application servers](#) > [server1](#) > Business Flow Manager

The business flow manager provides services to run and manage business processes within an application server. After configuring the business flow manager for the first time, you can only change the editable properties.

Configuration **Runtime**

General Properties

Retry limit
5

Retention queue message limit
20

Retention queue
jms/BPERetQueue

Hold queue
jms/BPEHldQueue

State Observers

☐ Enable Common Event Infrastructure logging

☐ Enable audit logging

Apply OK Reset Cancel

Containers

- [Business Process Choreographer Containers](#)
- [Deferred Configuration](#)

Related Items

- [Human Task Manager](#)
- [Business Process Choreographer Explorer](#)
- [Business Process Choreographer Event Collector](#)
- [Business Process Choreographer Observer](#)

Additional Properties

- [Custom Properties](#)

Figure A-15 Viewing the retention queue name, the hold queue name, the retry count, and the queue limits

3. On the Runtime page (Figure A-16), see the number of messages in each queue. Click the respective buttons along the top to refresh the message count, replay the hold queue, or replay the retention queue.

Application servers

Application servers

[Application servers](#) > [server1](#) > [Business Flow Manager](#) > **Runtime Configuration**

Replay failed messages that are in the hold and retention queues.

Configuration | Runtime

Refresh Message Count | Replay Hold Queue | Replay Retention Queue

General Properties

Hold queue messages
0

Retention queue messages
0

Message exceptions

State Observers

☐ Enable Common Event Infrastructure logging

☐ Enable audit logging

☐ Save runtime changes to configuration as well

Apply | OK | Reset | Cancel

Figure A-16 Runtime page - Working with messages

Resubmitting failed events

The Failed Event Manager user interface (Figure A-17) is in the WebSphere administrative console. It shows the number of failed events and provides a number of search capabilities.

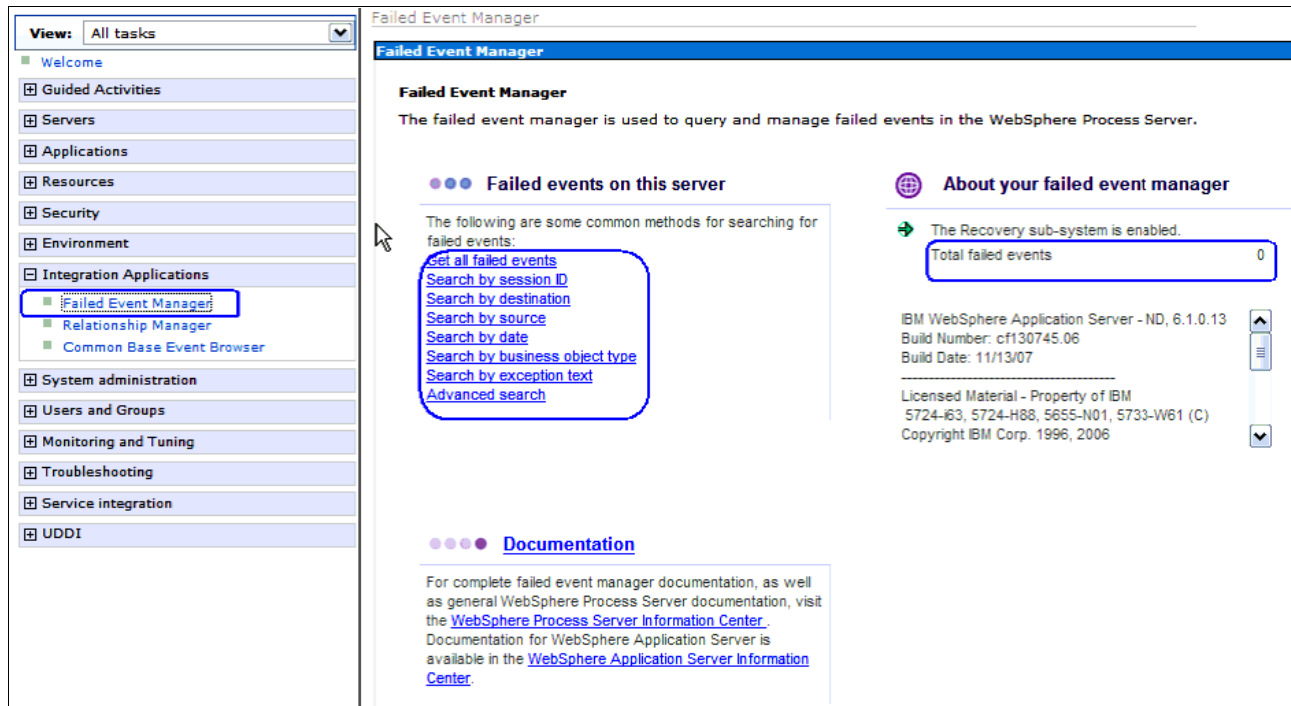


Figure A-17 Failed Event Manager user interface

Events that match the search criteria are displayed. They can then be selected and resubmitted as shown in Figure A-18.

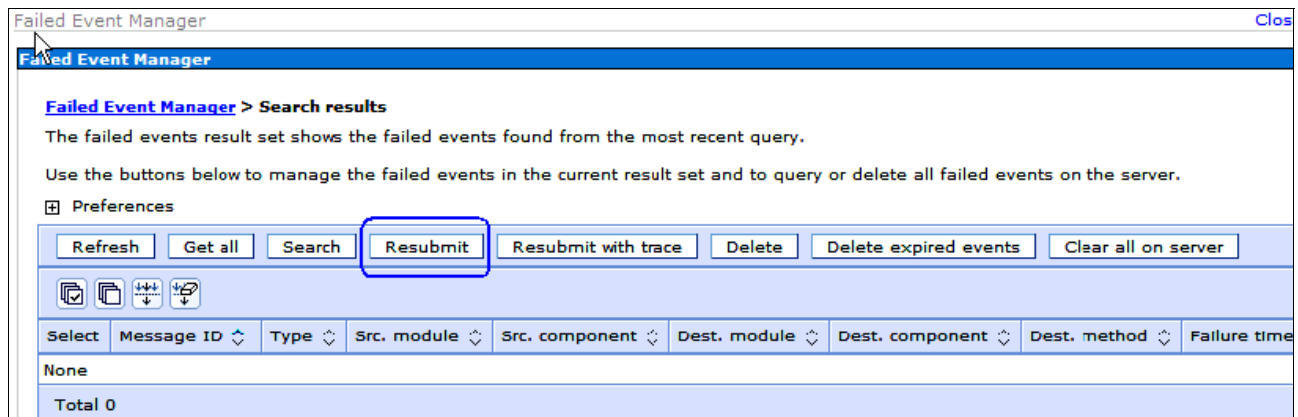


Figure A-18 Failed Event Manager search results

Business Process Choreographer maintenance

Scripts are provided for use in maintaining the Business Process Choreographer database. These scripts are in the *install_root/ProcessChoreographer/admin* directory.

Note: When using the scripts on a command line, make sure that the SOAP client timeout is set high enough to complete the requested operation for the **wsadmin** client.

Deleting process templates that are not in use

Use the **deleteInvalidProcessTemplate.py** script to remove, from the database, the templates and all objects that belong to the templates that are not in any corresponding valid application in the WebSphere configuration repository. This situation can occur if an application installation was cancelled or not stored in the configuration repository by the user.

Other scripts are available as explained in “Deleting process templates that are unused” in the WebSphere Process Server Information Center at the following address:

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

Deleting human task templates that are not in use

Use the **deleteInvalidTaskTemplate.py** script to remove human task templates and all objects that belong to them that are not in any corresponding valid application in the WebSphere configuration repository. This situation can occur if an application installation was canceled or not stored in the configuration repository by the user.

For more information, see “Deleting human task templates that are used” in the WebSphere Process Server Information Center at the following address:

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

Deleting completed process instances

Use the **deleteCompletedProcessInstances.py** script to delete completed process instances. A top-level process instance is considered completed if it is in one of the following end states:

- ▶ Finished
- ▶ Terminated
- ▶ End
- ▶ Failed

You can specify the criteria to selectively delete top-level process instances and all their associated data, such as activity instances, child process instances, and inline task instances, from the database.

For more information about this script, see “Deleting completed process instances” in the WebSphere Process Server Information Center at the following address:

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

See also “Deleting the completed process instances” on page 62.

Resolving indoubt transactions

Use the task in this section to resolve indoubt transactions and the messages associated with them. Use it only if other procedures, such as restarting the server in recovery mode (see “Starting the server in recovery mode” on page 51), have been tried unsuccessfully.

Transactions can become stuck in the indoubt state indefinitely due to an exceptional circumstance such as the removal of a node causing messaging engines to be destroyed. When a transaction becomes indoubt, it must be committed or rolled back so that normal processing by the affected messaging engine can continue.

You can use the administrative console to display the messages that indicate the cause of the problem. You can find details about this problem in “Listing messages on a message point” in the WebSphere Application Server Network Deployment Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjo0016_.html

If messages are involved in an indoubt transaction, the identity of the transaction is shown in a panel that is associated with the message. You can then resolve the transaction in two ways:

- ▶ Using the server’s transaction management panels
- ▶ Using methods on the messaging engine’s MBean

First attempt to resolve the indoubt transaction by using the application server transaction management panels. You navigate to these panels in the administrative console by clicking **Servers** → **Application servers** → **server name** → **Container Services** → **Transaction Service** → **Runtime** → **Imported prepared transactions - Review**.

If the transaction identity is displayed in the resulting panel, you can commit or roll back the transaction. Choose the option to roll back the transaction.

If the transaction identity is not displayed in the panel, the transaction identity was not enlisted with the Transaction Service on the server. In this case only, use methods on the MBean to display a list of the identities of the indoubt transactions managed directly by the messaging engine.

You can use the following methods on the messaging engine’s MBean to obtain a list of transaction identities (xid) and to commit and roll back transactions:

- ▶ `getPreparedTransactions()`
- ▶ `commitPreparedTransaction(String xid)`
- ▶ `rollbackPreparedTransaction(String xid)`

To invoke the methods, you can use a **wsadmin** command. For example, you can use a command in the following form to obtain a list of the indoubt transaction identities from a messaging engine’s MBean:

```
wsadmin> $AdminControl invoke [$AdminControl queryNames type=SIBMessagingEngine,*]  
getPreparedTransactions
```

Alternatively, you can use a script such as the one shown in Example A-2 to invoke the methods on the MBean.

Example: A-2 Script to invoke the MBean methods

```
foreach mbean [$AdminControl queryNames type=SIBMessagingEngine,*] {
  set input 0

  while {$input >=0} {
    set xidList [$AdminControl invoke $mbean getPreparedTransactions]

    set meCfgId [$AdminControl getConfigId $mbean]
    set endIdx [expr {[string first "(" $meCfgId] - 1}]
    set me [string range ${meCfgId} 0 $endIdx]

    puts "----Prepared Transactions for ME $me ----"
    set index 0
    foreach xid $xidList {
      puts "  Index=$index XID=$xid"
      incr index
    }
    puts "----- End of list -----"
    puts "Select index of XID to commit/rollback (-1 to continue) :"
    set input [gets stdin]

    if {$input < 0 } {
      puts "No index selected, going to continue."
    } else {
      set xid [lindex $xidList $input]
      puts "Enter c to commit or r to rollback XID $xid"
      set input [gets stdin]
      if {$input == "c"} {
        puts "Committing xid=$xid"
        $AdminControl invoke $mbean commitPreparedTransaction $xid
      }
      if {$input == "r"} {
        puts "Rolling back xid=$xid"
        $AdminControl invoke $mbean rollbackPreparedTransaction $xid
      }
    }
    puts ""
  }
}
```

This script lists the transaction identities of the transactions together with an index. You can then select an index and commit or roll back the transaction that corresponds to that index.

In summary, to identify and resolve indoubt transactions:

1. Use the administrative console to find the transaction identity of indoubt transactions.
2. If a transaction identity appears in the transaction management panel, commit or roll back the transactions as required.
3. If a transaction identity is not displayed in the transaction management panel, use the methods on the messaging engine's MBean. For example, use a script to display a list of transaction identities for indoubt transactions.

For each transaction:

- a. Enter the index of the transaction identity of the transaction.
 - b. Enter c to commit the transaction.
 - c. Enter r to roll back the transaction.
4. To check that transactions are no longer indoubt, restart the server and use the transaction management panel or use the methods on the messaging engine's MBean.

Reviewing the diagnostic log for DB2

On your UNIX system, type the following command:

```
tail -f /home/db2inst1/sqllib/db2dump/db2diag.log
```

If you see lines similar to those in Example A-3, part of the database is unresponsive. In this example, looking at the DB line, you can see that WPRCSDB is experiencing full transaction logs.

Example: A-3 DB2@ diagnostics log (db2diag.log)

```
2008-04-03-11.57.18.988249-300 I1247882009G504    LEVEL: Error
PID      : 16020                                TID : 3086133792  PROC : db2agent (WPRCSDB) 0
INSTANCE: db2inst1                            NODE : 000        DB : WPRCSDB
APPHDL   : 0-658                                APPID: 9.5.99.208.24960.080403084643
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, data protection services, sqlpWriteLR, probe:6680
RETCODE  : ZRC=0x85100009=-2062548983=SQLP_NOSPACE
          "Log File has reached its saturation point"
          DIA8309C Log file was full.
```



```
2008-04-03-11.57.18.994572-300 E1247882514G540    LEVEL: Error
PID      : 16020                                TID : 3086133792  PROC : db2agent (WPRCSDB) 0
INSTANCE: db2inst1                            NODE : 000        DB : WPRCSDB
APPHDL   : 0-658                                APPID: 9.5.99.208.24960.080403084643
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, data protection services, sqlpgResSpace, probe:2860
MESSAGE  : ADM1823E The active log is full and is held by application handle
          "274". Terminate this application by COMMIT, ROLLBACK or FORCE
          APPLICATION.
```

Another way to view the db2diag logs is to log in as the DB2 user and run **db2diag** as follows:

```
su -l db2inst1
db2diag | less
```

Deleting the completed process instances

Example A-4 on page 63 shows a script that wrappers the provided **deleteCompletedProcessInstances.py** script. This script is also included in the download materials for this paper. For more information about these materials, see Appendix B, "Additional material" on page 77.

By editing and placing correct user names, passwords, and paths in this script, you can delete "chunks" of process instances from the development environment.

Example: A-4 loopDeleteProcessInstances.py script

```
### Author: Ryan Claussen (rtclauss@us.ibm.com)
### Filename: loopDeleteProcessInstances.py
import time
import sys
import os

### This script accepts two arguments:
### - a timestamp of when to start deleting completed processes (in UTC)
### - an integer (seconds) which tells the script how much to step forward in time during each
### iteration
### Timestamp appears as: 2008-03-13T21:23:32
### (this is yyyy-mm-ddThh:MM:SS format)

### ***YOU MUST TAYLOR THIS SCRIPT TO YOUR ENVIRONMENT***
### You must properly format the invocation of the "deleteCompletedProcessInstances.py" script
### below.
### Additional information about the "deleteCompletedProcessInstances.py" script can be found in
### the WPS info center.
### See the "EDIT HERE" section below to identify necessary changes for your environment.

### run as:
### wsadmin.sh -username <username> -password <password> -f loopdelete.py <Timestamp to start
### at> <time in seconds to increment timestamp>

### For example:
### wsadmin.sh -username admin -password admin -f loopdelete.py 2008-03-14T12:43:23 600
### steps back in time 10 minutes during each iteration.

### Split the time argument into a python tuple for time.mktime
ta = str(sys.argv[0]).split("T")
y = int(ta[0].split("-")[0])
m = int(ta[0].split("-")[1])
d = int(ta[0].split("-")[2])

H = int(ta[1].split(":")[0])
M = int(ta[1].split(":")[1])
S = int(ta[1].split(":")[2])

### The three extra arguments in the tuple below; 0,2,-1, represent day of week, day number(out
### of 366), and if daylight savings time is in effect (-1 means unknown).
### deleteCompletedProcessInstance.py doesn't care about these values that's why they're hard
### coded here.
t1 = time.mktime((y,m,d,H,M,S,0,2,-1))
### Add the increment interval to the start time
t2 = t1 + int(sys.argv[1])
### Set localtime
lt = time.mktime(time.gmtime())

### Repeatedly loop and call the deleteCompletedProcessInstances.py script while incrementing
### the -completedBefore argument
while 1:
    if(t2 > lt):
        break
```

```

### Yes, use time.localtime() below or expect weird results. If you use time.gmtime() it will
###convert UTC time to UTC time (ie. it will add/subtract x hours, depending on your time
### zone, to the UTC time).
print "Trying to delete all instances before " +
      time.strftime("%Y-%m-%dT%H:%M:%S",time.localtime(t2)) + " UTC"
### EDIT HERE ###
    # Uncomment and edit the following line for Windows:
    o =os.system("d:/ibm/WID61/runtimes/bi_v61/bin/wsadmin.bat -username admin -password
admin -f
d:/ibm/WID61/runtimes/bi_v61/ProcessChoreographer/admin/deleteCompletedProcessInstances.py
-node widNode -server server1 -finished -completedBefore " +
time.strftime("%Y-%m-%dT%H:%M:%S",time.localtime(t2)))

# Uncomment and edit the following line for *nix
# o = os.system("/opt/ibm/WebSphere/ProcServer/bin/wsadmin.sh -username admin -password
admin -f
/opt/ibm/WebSphere/ProcServer/ProcessChoreographer/admin/deleteCompletedProcessInstances.py
-cluster default.AppTarget -finished -completedBefore " +
time.strftime("%Y-%m-%dT%H:%M:%S",time.localtime(t2)))

t2 = t2 + float(sys.argv[1])

```

Carefully selecting an adequate time slice prevents SOAP timeout exceptions when communicating with the deployment manager. The “adequate time slice” of administrable instances depends on many factors including the following factors:

- ▶ JVM tuning and memory allocations
- ▶ Transaction log configuration for the database server
- ▶ SOAP connection time-out configuration

For example, after you alter the script, you might run the following command:

```
wsadmin.<bat|sh> -user <USERNAME> -password <PASSWORD> -f
loopDeleteProcessInstances.py 2008-04-02T21:00:00 3600
```

This command runs the **deleteCompletedProcessInstances.py** script while increasing the completed before time stamp by one hour (60 minutes * 60 seconds) after every execution.

The **deleteCompletedProcessInstances.py** script has a timestamp parameter that you can use to control the number of instances that are deleted. The smaller the interval is, the fewer instances are deleted per invocation of the **deleteCompletedProcessInstances.py**. This script can be useful in situations where deletion of multiple process instances encounter transaction timeouts. Transaction timeouts during process deletion is the result of the following most common causes:

- ▶ An untuned database
- ▶ An overloaded system
- ▶ Attempting to delete too many process instances at once

Note: The **loopDeleteProcessInsance.py** script's timestamp parameter should be expressed in Coordinated Universal Time (UTC).

Determining the number of finished process instances within a given period of time

You can use the PROCESS_INSTANCE view and the Business Process Choreographer Explorer to determine the number of process instances that have finished within a given period of time. See “Using Business Process Choreographer’s predefined views” on page 41 for more information regarding views.

The COMPLETED column in the PROCESS_INSTANCE view shows time in UTC (Figure A-19). For example, in the following statement, 3 is the state code for a finished process:

```
SELECT template-name, name, completed FROM DB2INST1.Process_INSTANCE where state=3
order by completed;
```

TEMPLATE_NAME	NAME	COMPLETED
AccountCreation	_PI:90030119.bad212ba.2f9cfaf6.ed1a0040	May 5, 2008 8:45:03 PM 085000
AccountCreation	_PI:90030119.bad21267.2f9cfaf6.ed1a0031	May 5, 2008 8:45:05 PM 556000
AccountCreation	_PI:90030119.bad2131a.2f9cfaf6.ed1a0051	May 5, 2008 8:45:49 PM 880000
AccountCreation	_PI:90030119.bad21cac.2f9cfaf6.ed1a010f	May 5, 2008 8:45:52 PM 034000

Figure A-19 PROCESS_INSTANCE view

Use UTC when querying the PROCESS_INSTANCE view to determine the number of process instances that exist within a given period of time as shown in the example Figure A-20.

```
SELECT template_name, count(*) FROM DB2INST1.PROCESS_INSTANCE where
state=3 and
completed > TIMESTAMP('2008-05-05-20.45.00.000000') and
completed < TIMESTAMP('2008-05-05-20.46.00.000000')
group by template_name;
```

Figure A-20 Select statement to query the PROCESS_INSTANCE view

Six process instances finished in the specified time as shown in Figure A-21.

TEMPLATE_NAME	2
AccountCreation	6

Figure A-21 Viewing the AccountCreation process instances

You can also use the Business Process Choreographer Explorer to determine the number of process instances that finished within a given period of time.

Important: Keep the result sets small when using the Business Process Choreographer Explorer.

For example, in Figure A-22, we define a custom search that shows the instances completed within a given period of time. In this case, the user interface adjusts for local time, not UTC.

Search For Process Instances And Define Personalized Views

Use this page to search for process instances and to define personalized views. [i](#)

View Name Description

Process Criteria Task Criteria Property Filters User Roles View Properties

[Process Template Filters](#) **Process Instance Filters [i](#)** [Activity Instance Filters](#)

State
Compensated
Compensation Failed
Failed
Failing
Finished
Running
Suspended
Terminated
Terminating

Name

Started After Date ☐ Date: Time:
(yyyy-mm-dd) (hh:mm:ss)

Started Before Date ☐ Date: Time:
(yyyy-mm-dd) (hh:mm:ss)

Completed After Date ☒ Date: Time:
(yyyy-mm-dd) (hh:mm:ss)

Completed Before Date ☒ Date: Time:
(yyyy-mm-dd) (hh:mm:ss)

Figure A-22 Searching for process instances and defining personalized views

Select **Completed** as one of the selected columns to be displayed as shown in Figure A-23.

Search For Process Instances And Define Personalized Views

Use this page to search for process instances and to define personalized views. [?](#)

[Search](#) [Save](#) [Reset](#)

View Name Description

[Process Criteria](#) [Task Criteria](#) [Property Filters](#) [User Roles](#) [View Properties](#)

[List Columns](#) [List Properties](#) [View Settings](#)

List Columns For Properties

- Completed
- Created
- Description
- Parent Name
- Process Instance ID
- Process Instance Name
- Process Template Name
- Resumes
- Started
- Starter
- State
- Top-Level Name

[Add](#)

List Column For Custom Properties

Property Name

Selected List Columns

- Process Instance Name
- Process Template Name
- State
- Completed
- Started
- Parent Name
- Top-Level Name

[Up](#) [Down](#) [Remove](#)

Figure A-23 Selecting the Completed column

Note that Business Process Choreographer Explorer limits the number of rows that are returned.

Do not use the Business Process Choreographer Explorer if you need to work with large result sets. Instead query the PROCESS_INSTANCE view as shown in Figure A-24.

Search For Process Instances And Define Personalized Views

Use this page to search for process instances and to define personalized views. i

Search

Save

Reset

View Name

Description

Process Criteria

Task Criteria

Property Filters

User Roles

View Properties

List Columns

List Properties i

View Settings

Rows On One Page

20

Maximum Results

1000

Default is 20000.

Figure A-24 Identifying the results set size

As expected, the Business Process Choreographer Explorer and the PROCESS_INSTANCE view query each show that six process instances finished within the given period of time (shown in Figure A-25).

Search Results for Process Instances

Use this page to work with the results of your search for process instances. i

Compensate

Terminate

Delete

Suspend

Resume

Restart

Activities

Related Processes

<input type="checkbox"/>	Process Instance Name ⌵	Process Template Name ⌵	State ⌵	Completed ⌵
<input type="checkbox"/>	_PI:90030119.bad21267.2f9cfaf6.ed1a0031	AccountCreation	Finished	5/5/08 3:45:05 PM
<input type="checkbox"/>	_PI:90030119.bad212ba.2f9cfaf6.ed1a0040	AccountCreation	Finished	5/5/08 3:45:03 PM
<input type="checkbox"/>	_PI:90030119.bad213d2.2f9cfaf6.ed1a0070	AccountCreation	Finished	5/5/08 3:45:58 PM
<input type="checkbox"/>	_PI:90030119.bad21cac.2f9cfaf6.ed1a010f	AccountCreation	Finished	5/5/08 3:45:52 PM
<input type="checkbox"/>	_PI:90030119.bad2131a.2f9cfaf6.ed1a0051	AccountCreation	Finished	5/5/08 3:45:49 PM
<input type="checkbox"/>	_PI:90030119.bad21f00.2f9cfaf6.ed1a015b	AccountCreation	Finished	5/5/08 3:45:59 PM

Items found: 6

Items selected: 0

Page 1 of 1

Items per page: 20 ⌵

Figure A-25 Search results

Business Process Choreographer Explorer tips

The Business Process Choreographer Explorer provides a user interface for administrators to manage business processes and human tasks. The following line shows the typical Business Process Choreographer Explorer URL:

```
http://yourhost:yourport/bpc
```

Figure A-26 shows the Business Process Choreographer Explorer.

The screenshot displays the Business Process Choreographer Explorer web application. The top navigation bar includes links for Welcome, PrimaryAdmin, Logout, Define Views, Customize, Help, and About. The left sidebar contains a tree view with categories: Process Templates (My Process Templates), Process Instances (Started By Me, Administered By Me, Critical Processes, Terminated Processes, Failed Compensations), Task Templates (My Task Templates), and Task Instances (My To-dos, All Tasks, Initiated By Me, Administered By Me, My Escalations). The main content area is titled "Process Instances Administered By Me" and includes a sub-header: "Use this page to work with process instances for which you are the process administrator". Below this are buttons for Compensate, Terminate, Delete, Suspend, Resume, Restart, Activities, Related Processes, and View. A table lists process instances with columns: Process Instance Name, Process Template Name, State, Started, and Parent. The table contains 20 rows of data, all for "AccountCreation" templates in a "Finished" state, with start times ranging from 5/5/08 4:39:12 PM to 5/5/08 4:39:29 PM. At the bottom, it shows "Items found: 8,951", "Items selected: 0", "Page 1 of 448", and a "Goto Page" field with the value "1".

Figure A-26 Business Process Choreographer Explorer

The Business Process Choreographer Explorer provides one option to check the health of the Business Process Choreographer database (BPEDB). If the Business Process Choreographer Explorer is unable or slow to retrieve data, this is an indication that there might be a database problem.

Attempting to retrieve thousands of process instances or tasks is not wise if performance or database problems are suspected. A better option is to select a view that does not retrieve considerable data, such as My Process Templates, or limit the amount of data retrieved for another view.

Customization options (shown in Figure A-27) provide the capability to set the login view. You might want to set this to a view that does not consume considerable server or database resources.

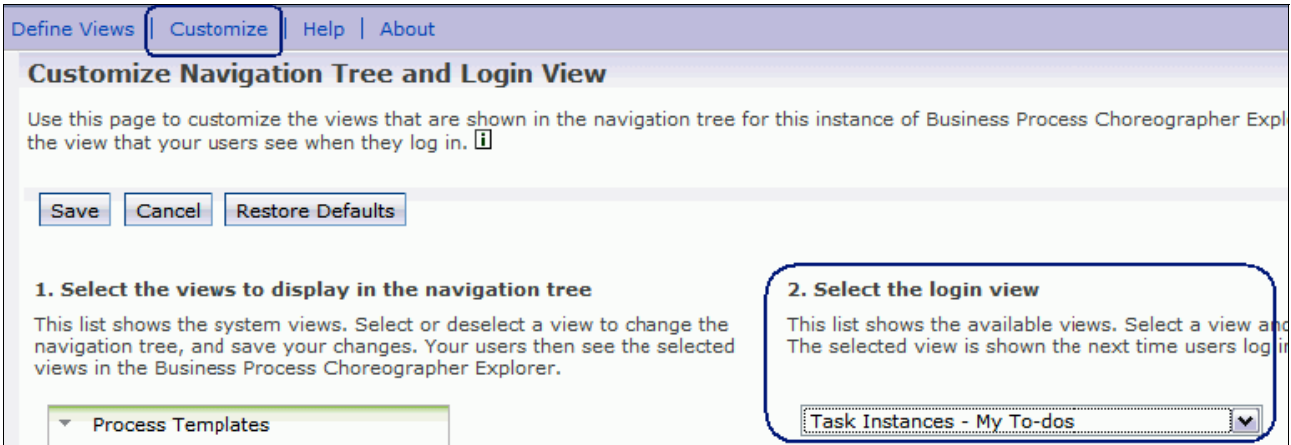



Figure A-27 Customization options

The search options icon () provides the capability to scope the data that is returned and limit the maximum number of results. Scoping the data and limiting the results are important strategies if the system is experiencing poor performance, out of memory errors, or transaction timeouts. Figure A-28 shows an example of scoping the data.

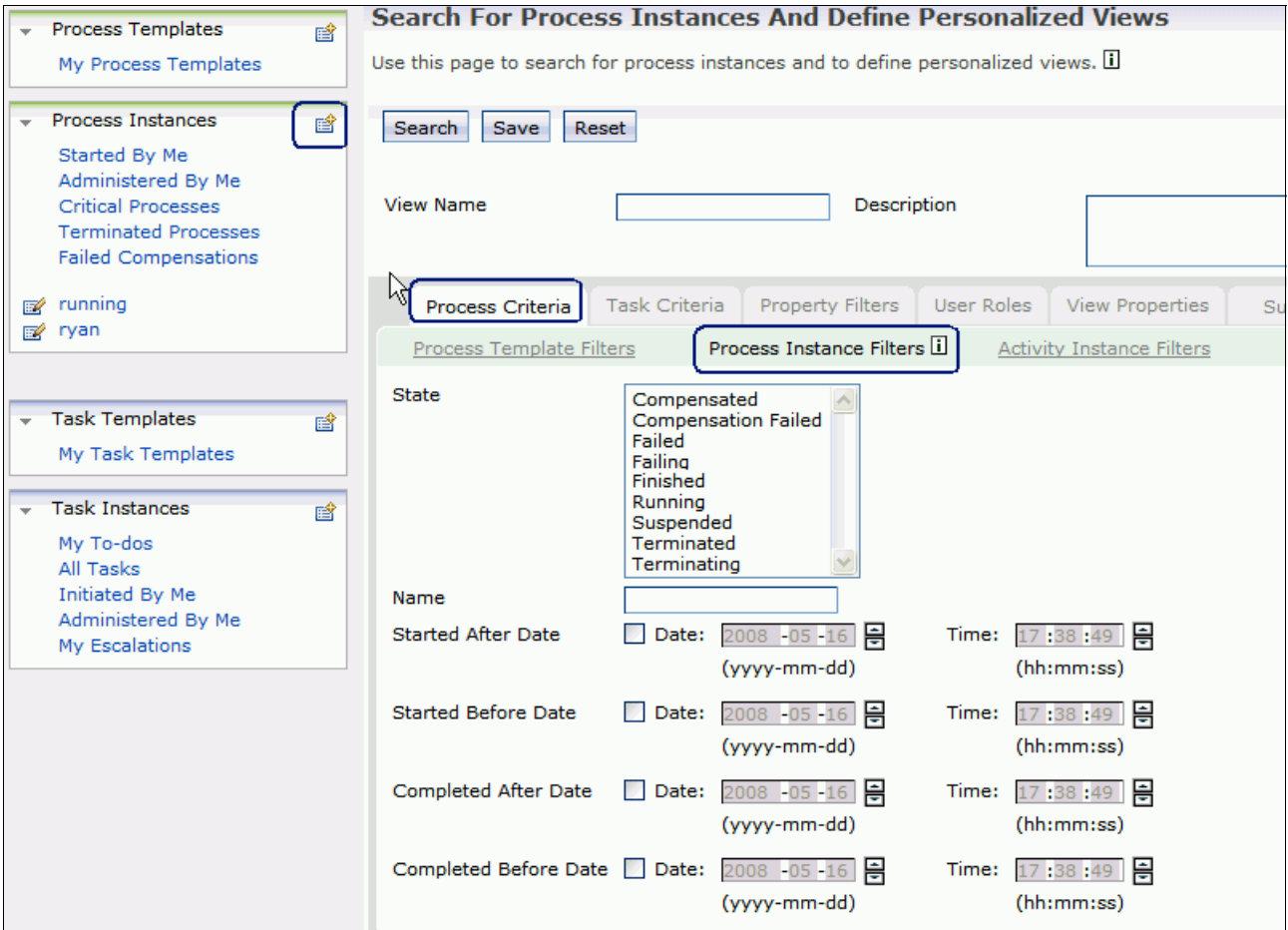


Figure A-28 Process Instances - Business Criteria

Figure A-29 shows an example of limiting results.

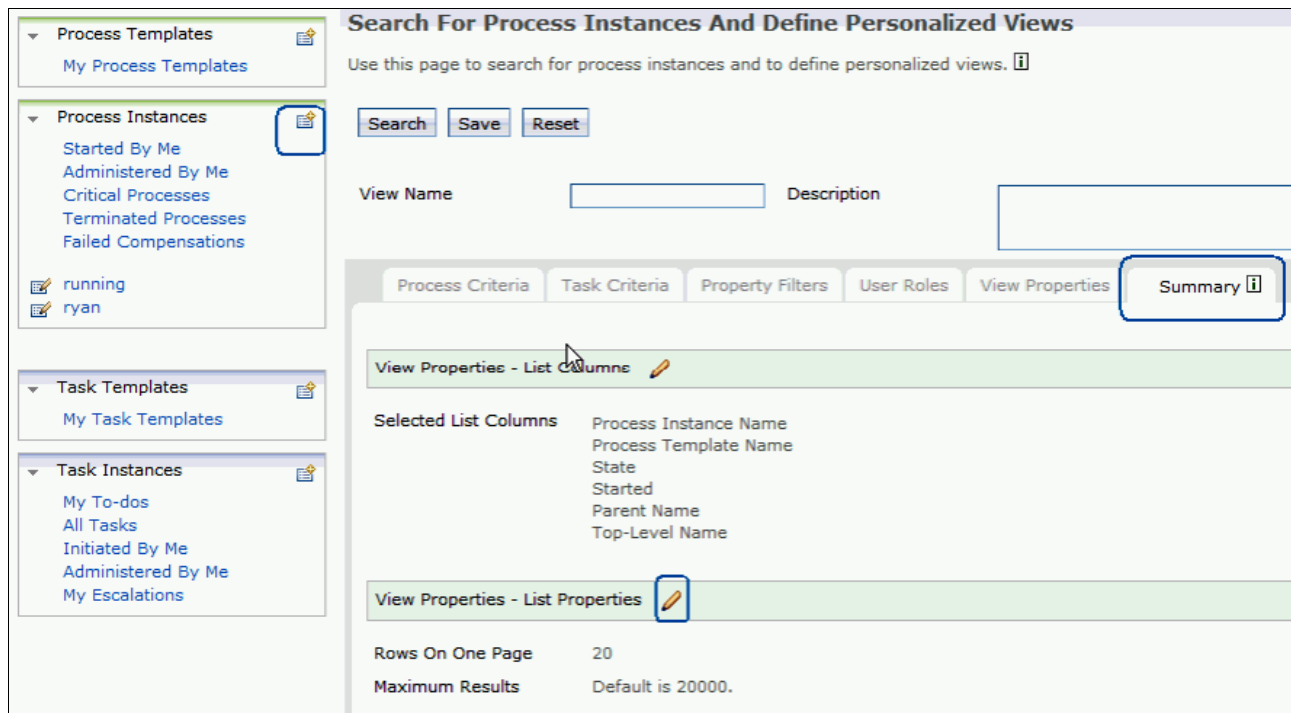


Figure A-29 Process Instances - List properties

You can also use the Business Process Choreographer Explorer to identify process instances that are in need of manual repair. The following process instances are in need of repair:

- ▶ Critical processes
- ▶ Processes in a running state with activities in a running state in the absence of any related messages in the following system locations:
 - SCA queues
 - BPE internal queue
 - BPE hold queue
 - BPE retention queue
 - Failed Event Manager

Critical processes are instances in a running state that have activities in a stopped state. This situation can occur if an activity is defined to *not continue on error*, as shown in Figure A-30.

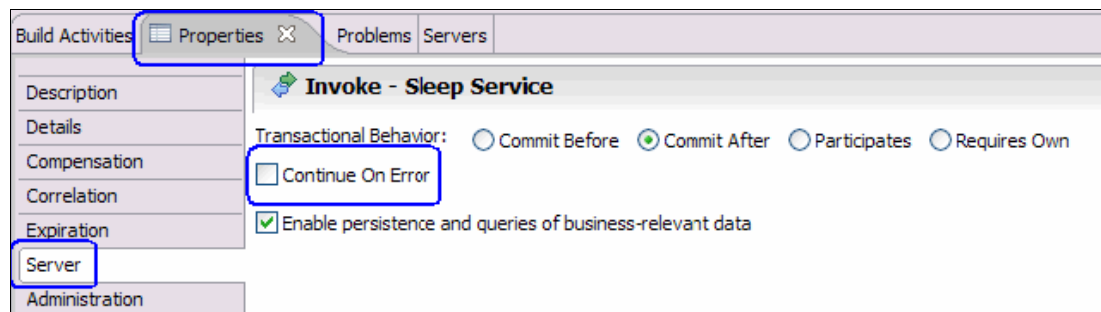


Figure A-30 Invoking the Sleep Service

The Business Process Choreographer Explorer has a view to display critical processes, which is shown in Figure A-31.

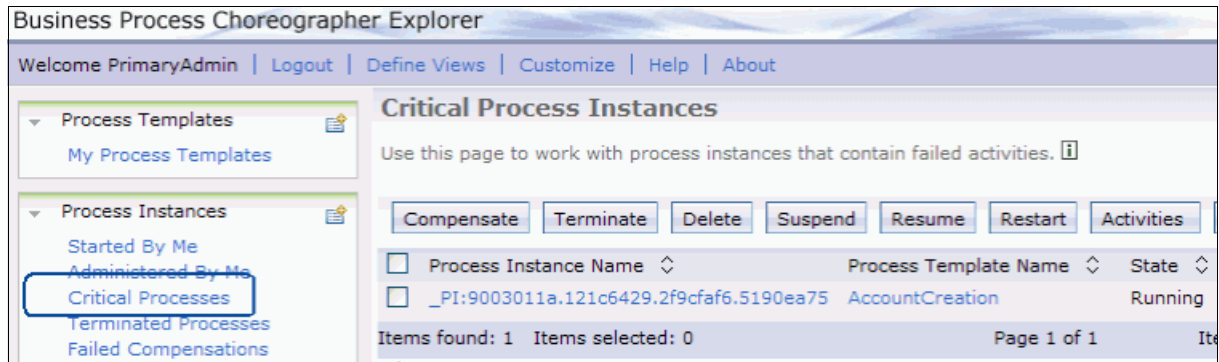


Figure A-31 WebSphere Integration Developer - Business Process Choreographer Explorer critical processes

See “Repairing processes and activities” in the WebSphere Process Server Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/t7procrepair.html>

The other category in need of repair is processes that are in a running state with activities in a running state even though there is an absence of any related messages in the following locations:

- ▶ SCA queues
- ▶ BPE internal queue
- ▶ BPE hold queue
- ▶ BPE retention queue
- ▶ Failed Event Manager

Consider a scenario where manual repair is performed on one of these process instances. Let us assume that all process instances that started before 08 May are expected to have completed by now. A custom search can be used to identify running process instances that were started before a specified date (Figure A-32 on page 73).

Welcome PrimaryAdmin | Logout | Define Views | Customize | Help | About

Search For Process Instances And Define Personalized Views

Use this page to search for process instances and to define personalized views. [i](#)

[Search](#) [Save](#) [Reset](#)

View Name Description

Process Criteria Task Criteria Property Filters User Roles View Properties

Process Template Filters Process Instance Filters [i](#) Activity Instance Filters

State: Compensated Compensation Failed Failed Failing Finished **Running** Suspended Terminated Terminating

Name

Started After Date ☐ Date: Time:
(yyyy-mm-dd) (hh:mm:ss)

Started Before Date ☒ Date: Time:
(yyyy-mm-dd) (hh:mm:ss)

Figure A-32 Custom search by date

There are not any related messages at the locations specified in the previous list. Therefore, we select a process instance and view its activities as shown in Figure A-33.

Search Results for Process Instances

Use this page to work with the results of your search for process instances. [i](#)

[Compensate](#) [Terminate](#) [Delete](#) [Suspend](#) [Resume](#) [Restart](#) **[Activities](#)**

<input type="checkbox"/>	Process Instance Name	Process Template Name	State
<input checked="" type="checkbox"/>	_PI:90030119.baef067d.c49cfaf6.7482df6c	AccountCreation	Running
<input type="checkbox"/>	_PI:90030119.baef0913.c49cfaf6.7482dfab	AccountCreation	Running

Figure A-33 Search results - Viewing activities

The SleepService activity was activated on 05 May. In this scenario, SleepService is an automated activity that invokes a request-response operation. The operation should have completed in a short period of time. The process is waiting for a response that does not exist. Therefore, it is in need of repair.

Because this activity can be safely restarted, we select it and click **Restart**. If this is an activity that cannot be safely restarted, then it must be forced to complete.

When the activity is restarted, it is invoked a second time. The response should be delivered back to the process, and the process should complete normally (Figure A-34).

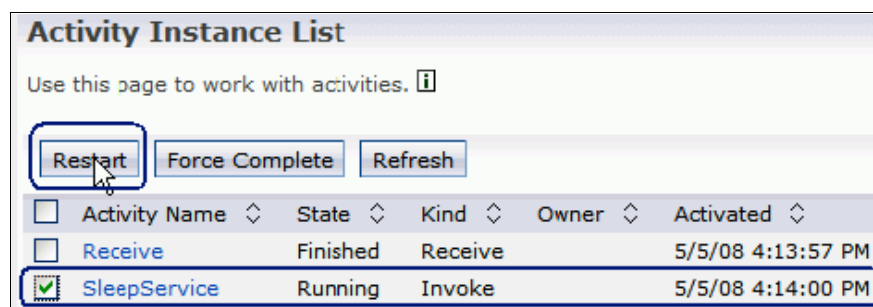


Figure A-34 Activity Instance List for SleepService

IBM Support Assistant

The IBM Support Assistant is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. With IBM Support Assistant, you can perform the following tasks:

- ▶ Search for an answer to your question or problem in many different locations at the same time.
- ▶ Get speedy access to critical product information.
- ▶ Run free troubleshooting and diagnostic tools on your troublesome application.
- ▶ Shorten the amount of time it takes to resolve your problem with automated data gathering and submission.

For more information about IBM Support Assistant, see the following Web address, which includes a link to download the latest version of the workbench:

<http://www.ibm.com/software/support/isa/>

IBM Support Assistant includes specific support for a large number of IBM products and versions of IBM products including the following products:

- ▶ WebSphere Process Server
- ▶ WebSphere Enterprise Service Bus
- ▶ WebSphere Integration Developer
- ▶ WebSphere Application Server
- ▶ WebSphere Business Monitor
- ▶ WebSphere Business Modeler
- ▶ WebSphere Service Registry and Repository

The product-specific support includes diagnostic aids and tools to automatically gather log, trace, and configuration data and send it to IBM as part of a PMR. This also works across a WebSphere cell. Many of the WebSphere Application Server tools are also applicable to WebSphere Process Server and WebSphere ESB. The following tools are some that are available with IBM Support Assistant:

- ▶ IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer
- ▶ IBM Pattern Modeling and Analysis Tool for Java Garbage Collector
- ▶ Log Analyzer

- ▶ Memory Dump Diagnostic for Java (MDD4J)
- ▶ Symptom Editor
- ▶ ThreadAnalyzer

For additional information about these tools, see the following Web address:

<http://www.ibm.com/software/support/isa/isa40/tools.html>

IBM Support Assistant updates are made throughout the year and are not tied to product release schedules. It also includes IBM Guided Activity Assistant (IGAA), which is a step-by-step guide to information and tools to resolve problems. You can find more information at the following address:

http://www.ibm.com/developerworks/websphere/techjournal/0705_supauth/0705_supauth.html

In IBM Support Assistant V4, WebSphere Application Server support for the IGAA function is available on the Guided Troubleshooter tab, which is shown in Figure A-35.

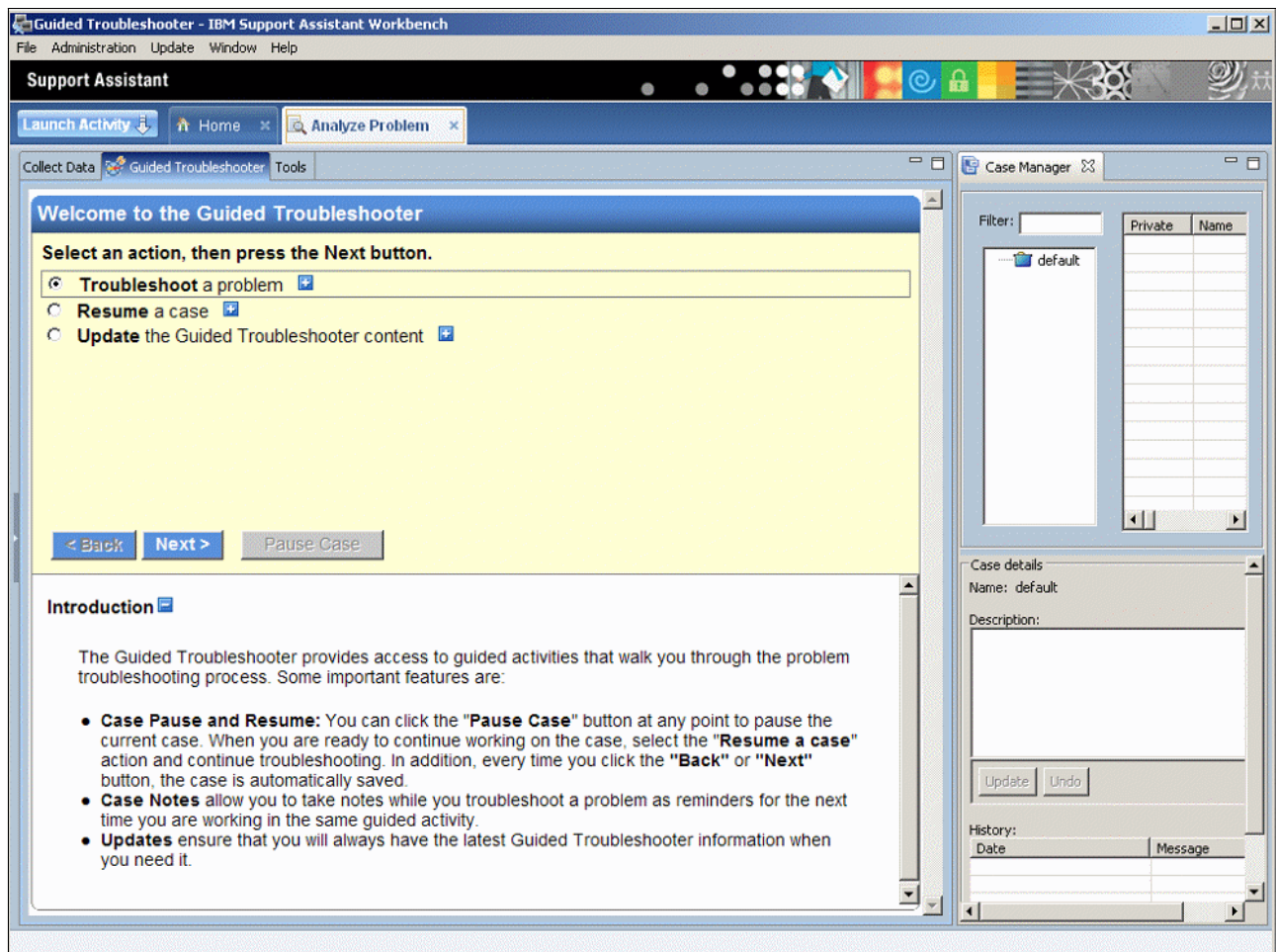


Figure A-35 IGAA - Guided Troubleshooter

Recovering the messaging subsystem

If the messaging system experiences problems, you might need to recover the underlying messaging subsystem. Typically this involves checking the state of various queues but can include analyzing the integration bus infrastructure. For information about how to recover the messaging subsystem, see “Troubleshooting service integration message problems” in the WebSphere Application Server Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.webSphere.pmc.zseries.doc/tasks/tju_msg_probs.html



Additional material

This paper refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this paper is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/REDP4466>

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 IBM Redpaper form number, REDP4466.

Using the Web material

The additional Web material that accompanies this paper includes the following files:

<i>File name</i>	<i>Description</i>
Observation Report.xls	A sample observational report (Excel spreadsheet). See “Solution-specific problem determination methodology” on page 13.
loopDeleteProcessInstances.py	See “Deleting the completed process instances” on page 62.
stopMDB.py	See “Stopping the processing of asynchronous events” on page 52.

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 79. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM WebSphere Business Process Management V6.1 Performance Tuning*, REDP-4431
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

Online resources

These Web sites are also relevant as further information sources:

- ▶ *IBM WebSphere Developer Technical Journal: Transactional high availability and deployment considerations in WebSphere Application Server V6*
http://www.ibm.com/developerworks/websphere/techjournal/0504_beaven/0504_beaven.html
- ▶ WebSphere Process Server Information Center
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wps.610.doc/welcome_top_wps.htm
- ▶ WebSphere Application Server Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>
- ▶ IBM Support Assistant V4.0 Tools
<http://www.ibm.com/software/support/isa/isa40/tools.html>
- ▶ The Support Authority: Introducing the IBM Guided Activity Assistant
http://www.ibm.com/developerworks/websphere/techjournal/0705_supauth/0705_supauth.html

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



IBM WebSphere Process Server Best Practices in Error Prevention Strategies and Solution Recovery



Redpaper™

Preventive practices

Strategies for recovery

Troubleshooting tips

IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus (ESB) are middleware servers that are optimized to enable the execution and management of business process management (BPM) and service-oriented architecture (SOA) solutions. They are built on the foundational capabilities of IBM WebSphere Application Server.

Middleware systems execute under various conditions, not all of which are traditionally “good path” conditions. Many of the key features within these products are in place to deal with the uncertainty that can arise through what might appear to be normal operations.

The topic of system analysis and recovery is a broad topic, for which entire books have been written. This IBM Redpaper publication provides basic guidance on how to handle ordinary and extraordinary conditions that might arise. We base this paper on the assumption that you are familiar with the WebSphere Process Server and WebSphere ESB products, the basic architectural principles upon which they build, and the basic kinds of applications that they execute. We also assume that you have a base understanding of integration projects, planning, and implementations.

Unless otherwise specified, the information is relevant to all versions of WebSphere Process Server and WebSphere ESB starting with V6.1.0. Although you will find that most of the information is applicable to the V6.0.2 product line, the internal validation of IBM was conducted on a V6.1.0 WebSphere Process Server cell with the standard *golden topology*. Results and actions might vary slightly on releases prior to V6.1.0.

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