Oracle® SOA Suite

Best Practices Guide

10*g* Release 3 (10.1.3.3.0)

E10971-01

December 2007

ORACLE®

Oracle SOA Suite Best Practices Guide, 10*g* Release 3 (10.1.3.3.0)

E10971-01

Contributors: Vikis Anand, Dave Berry, Marc de Brouwer, Christophe Dupupet, Troy Hedges, Jeff Hutchins, Francis Ip, Mohan Kamath, Mark Kennedy, Olivier Le Diouris, FX Nicolas, Natascha Schoenfeld, Bo Stern, Albert Tam, Clemens Utschig-Utschig

Contents

Preface xvii

Audience xvii
Documentation Accessibility xvii
Related Documents xviii
Conventions xix

Part I Oracle SOA Suite Components

1 Oracle BPEL Process Manager

Threading 1-1
What is the Relationship Between ReceiverThreads and dspMaxThreads? 1-1
How Do dspMaxThreads and dspInvokeAllocFactor Throttle Requests? 1-3
What are the Different Types and Responsibilities of MDBs in Oracle BPEL Server? 1-4
Transactions 1-4
What are the Transaction Boundaries in a Flow? 1-4
How Do I Force a Rollback in a BPEL Flow? 1-5
What is the Impact of the transaction=participate Property on BPEL Transactionality? 1-6
Why Does the Instance of an Invoked Process Not Display in Oracle BPEL Control? 1-8
Can I Receive a Transaction Timeout If Invoking a Synchronous Transient Process? 1-9
How Do I Modify Transaction Timeout Settings for Oracle BPEL Server? 1-9
What are the Transaction Semantics with Oracle ESB - Oracle BPEL Process Manager Calls or Oracle BPEL Process Manager - Oracle ESB Calls? 1-10
Behavior in 10.1.3.1.0 1-10
Behavior in 10.1.3.3.0 1-12
BPEL Process Configuration 1-14
How Does syncMaxWaitTime Work and Why Do I Not See My Synchronous Requests Timing Out? 1-14
How Does a Nonblocking Invoke Work? 1-16
Recovery and High Availability 1-18
How Does an Invoke Message and Activity Display in the Manual Recovery Page? 1-18
How Do I Implement an Automated Recovery Mechanism (10.1.*x*)? 1-20
How Does Oracle BPEL Server Handle a RAC Node Failover? 1-21
Deployment 1-21
How Do I Resolve a Process Lock Timeout Error? 1-22
Scheduling and Activity Expiration 1-24

How Does Oracle BPEL Server Use Quartz? 1-24
How Do I Tune the Performance of Quartz? 1-25
Why Does My Wait Activity and onAlarm Branch of a Pick Activity Not Expire? 1-26
Performance 1-27
Is Much Overhead Incurred when Calling a BPEL Subprocess? 1-27
Flow Logic 1-28
Which Error Conditions Cannot be Handled with a Fault Handler? 1-28
Why Do I Receive Empty and Zero Node Selection Failure Faults? 1-28
Fixing a Zero Node/Empty Node Selection Failure Fault 1-29
Methodology 1-30
Should I Implement a Daemon using Oracle BPEL Process Manager? 1-30
How Do I Handle Large XML Documents within BPEL (10.1.3.3)? 1-34
How Do I Model a Two Stage Real-Time Response Process and Continue Using Oracle BPEL Server Scalability? 1-35
When Do I Use Embedded Java within BPEL? 1-36
Java Embedding Example 1 1-36
Java Embedding Example 2 1-36
Deployment Descriptor Properties 1-36

2 Oracle Enterprise Service Bus

Third-Party Compatibility and Certifications 2-1
Can Oracle ESB Run in a Third-Party J2EE Container (JBoss, WebSphere, WebLogic)? 2-2
Can I Use a Third-Party JMS Provider (TIBCO, WebSphereMQ, SonicMQ) as the Internal Oracle ESB Transport? 2-2
Can Oracle ESB Send to or Receive from a Third-party JMS Provider (TIBCO EMS, WebSphere JMS)? 2-2
What Databases are Supported as a Storage Backend for the ESB Server? 2-2
On Which Operating Systems Does Oracle ESB Run? 2-2
How Can I Use an Oracle 10.1.2 Database Server on a Windows Client Workstation with Oracle ESB and Oracle SOA Suite? 2-2
Can I Consume Oracle ESB Services from .NET Clients? 2-3
Which Adapters are Available with Oracle ESB? 2-3
How Do I Connect to a Protocol or Application for which Oracle Does Not Have an Adapter? 2-3
Can You Create an ESB-to-ESB Bridge Across a Firewall? 2-3
How Do I Call an Oracle ESB Routing Service from ADF? 2-3
Can Oracle ESB Handle Inbound HTTP Posts? 2-4
Does Oracle ESB Support WSIF and are Samples and Additional Information Available? 2-4
Does Oracle ESB Support RPC Style SOAP Services? 2-4
Core Functionality 2-4
What Type of Transaction Support Does Oracle ESB Have? 2-4
More Transaction Use Case Questions and Answers 2-5
Is a Database Installation Required for Oracle ESB Only If You Do Not Use JMS/AQ? 2-5
Does Oracle ESB Support Multipart Messages or Binary Attachments? 2-6
Besides Being a Logical Grouping of ESB components, What Else Does an ESB System Provide? 2-6
How Can I Send an E-mail Notification from Oracle ESB? 2-6

If a Routing Service Has Two Asynchronous Services, What Happens If Both Fail? Do One or Two Error Hospital Messages Get Generated? 2-6
Does Oracle ESB Support a Management API like Oracle BPEL Process Manager Provides? 2-7
Is Service Communication in the Same Java Container Optimized to Avoid the Overhead of TCP Connections and HTTP Request Formatting? 2-7
What Are the Best Practices for Oracle ESB User Accounts And Roles? 2-7
Does Oracle ESB Support Oracle BAM? 2-7
How Do I Manage the Timeout Duration for a Synchronous Request-Response from a SOAP Invocation Service? 2-7
What XML document Size Does Oracle ESB support? 2-8
Does Oracle ESB Support Message Ordering? 2-8
Does Oracle ESB Support the JCA Adapter rejectedMessageHandlers Property Like Oracle BPEL Process Manager? 2-8
How Can I Set Up Oracle ESB for High Availability? 2-8
Does Oracle ESB Support Filtering or Setting Message Headers? 2-8
Can Oracle ESB Pass Security Credentials When Invoking an External Web Service? 2-9
Can I Resubmit Multiple Error Instances at the Same Time? 2-9
What Is the Oracle ESB Number of Listeners Feature? 2-9
Can I Register an Oracle ESB Project from the Command Line with Ant Scripts? 2-9
Can Automatic Retry Be Configured in the Following Scenario? 2-9
Are There Known Issues with Oracle ESB WSDLs? 2-10
How Can I Disable the Tracking Messages for One or More Services? 2-10
What Is the Correct Way to Purge ESB Instance Data? 2-10
How Can I Propagate the File Name from Input File Adapter to Output File Adapter Using JCA Headers? 2-12
Does Oracle ESB Support an Inbound E-mail Adapter? 2-12
Why are BPEL Processes Displayed in Oracle ESB Control? 2-12
    Should Every BPEL Process Display in Oracle ESB Control? 2-12
    Are There Known Issues with Synchronization of Oracle ESB Control? 2-12
Can I Install the Oracle ESB and BPEL Components Without the Rest of Oracle SOA Suite? 2-12
Is the reset.bat/reset.sh Script a Supported Command and Where Is It Documented? 2-12
    Environment Variables Sample 2-13
How Do You Handle Multiple BPEL Versions When Using an ESB SOAP Service to Call BPEL from Oracle ESB? 2-13
    Oracle BPEL Process Manager WSDL Examples 2-13
How Do I Override a WSIF Invocation of an Oracle BPEL Process Manager Service from an Oracle ESB Service and Set the Endpoint Property Location? 2-13
Is There a Property for Pointing at Alternate Locations for a WSDL of a SOAP Service Endpoint? 2-13
What Is the Separator for Different SOAP Endpoints When Configuring the Location Endpoint Property? 2-14
Does Oracle ESB Support Functionality Similar to Oracle BPEL Process Manager Dynamic Partner Links? 2-14
Can You Guarantee Once-and-Once-Only Processing in Oracle ESB Assuming the Failure of the Processing Node? 2-14

Can I Use Custom XPath Functions in Oracle ESB? 2-14
What Is Error Hospital and Do You Have to Set It Up? 2-15
Does Oracle ESB Support Messages with No Namespace Prefix? 2-15
How Can I Get an Instance ID for Oracle ESB? 2-15
Can I Invoke a BPEL Process Locally (Using Local EJB References Instead of RMI) from Oracle ESB in 10.1.3.3? 2-16
What Can I Do with Oracle ESB that I Cannot Do with Oracle BPEL Process Manager? 2-17
Performance 2-17
How Fast Is Oracle ESB? 2-17
Are There Any Recommendations to Follow to Maximize Oracle ESB Performance? 2-18
General Performance Recommendations 2-18
Web Services SOAP/HTTP 2-18
OC4J JMS 2-18
JMS Adapter 2-18
OC4J AS JMS Provider for File Persistence and In-Memory 2-19
Oracle ESB Sizing Questions 2-19
Use Case 1 2-19
Questions 2-20
Response 2-20
Use Case 2 2-20
Response 2-21
Project Lifecycle 2-21
Can I Retire Oracle ESB Projects Like in Oracle BPEL Process Manager? 2-21
Does Oracle ESB Support Versioning of Services Like Oracle BPEL Process Manager? 2-21
Troubleshooting 2-21
Basic information to Provide When Reporting A Problem 2-22
ORA-00018: maximum number of sessions exceeded (Oracle XE Running Out Of Processes) 2-23
Helpful Troubleshooting Tools - Web Services 2-23
How Can I Enable End-to-End Instance Tracking between Oracle ESB and Oracle BPEL Process Manager in the OrderBooking Demonstration? 2-24
I Cannot Register Projects or Connect to Oracle ESB Control — How Do I Check that the Server is Running and On Which Port? 2-24
Ensure the Container is Running 2-24
Test that Oracle ESB Control is Functional 2-24
Test Other Projects 2-24
Identify the Port on Which the Oracle ESB Process is Listening 2-25
I See Inconsistent Data in Oracle ESB Control and the DefaultSystem Group is Missing (How to Reset Your Slide/webDAV Cache) 2-25
Why Does Oracle ESB Control Not Display Services in the Services Panel or Instance Tracking Information? 2-26
I Recreated the oraesb Schema and Now the DefaultSystem is Missing. How Do I Create It? 2-27
What Logging Facilities are Available for Oracle ESB in an Application Server Installation? 2-27
How Do I Fix the Registration Error "system and service group that the project was using was already present in the server"? 2-27
Why Am I Getting Error IOException ... access error ... Database is set to dirty, this may

mean it is corrupt? 2-28
Installation 2-28
How Do I Change Oracle ESB Schema Creation (IRCA Utility) to Use My Own Tablespace? 2-28
Transaction Semantics In Oracle ESB 2-29
Same System — All Asynchronous 2-29
Same System — All Synchronous 2-29
Same System — Synchronous and Asynchronous 2-29
Multisystem — All Asynchronous (Systems A and B) 2-29
Multisystem — All Synchronous 2-30
Multisystem — Both Synchronous and Asynchronous 2-30
Oracle BPEL Process Manager — Oracle ESB Pattern 2-30
Synchronous BPEL Process Calling Synchronous Oracle ESB Routing (Oracle ESB Native Binding) 2-30
Synchronous BPEL Process Calling Synchronous Oracle ESB Routing (SOAP Binding) 2-31
Asynchronous BPEL Process Calling Asynchronous Oracle ESB Routing (Oracle ESB Native /SOAP binding) 2-31
Adapters and Transactional Behavior 2-31
Transactional Behavior Configuration 2-31
Nontransactional Behavior Configuration 2-32
Use Case for Using Pure SQL with Dynamic Routing in Oracle ESB 2-32
Setting Up the Use Case for Using Pure SQL with Dynamic Routing in Oracle ESB 2-33
Setting Up the Database 2-33
Setting Up the Data Sources 2-34
Setting Up the Database Adapter 2-37
Modeling the Use Case with ESB Designer 2-40
Creating the Routing Service 2-43
Deploying the Use Case 2-46
Testing the Use Case 2-47

3 Oracle Technology Adapters

Release 10.1.3.3 Enhancements 3-1
Installation Issues 3-1
Adapters are Not Getting Installed 3-2
AQ Adapter 3-2
Can I Deploy and Run an AQ Adapter Process Using a Different Schema from the Design Time Schema? 3-2
Database Adapter 3-2
XA Configuration - Two Database Adapter Invokes Committing or Rolling Back as a Unit 3-3
Configuring the Database Adapter for Global Transaction Participation 3-3
True XA: Two-Phase (XA) Versus One-Phase (Emulated) Commit 3-3
Both Invokes in the Same Global Transaction 3-4
Failure Must Cause a Rollback 3-4
Related Questions 3-4
Can I Guarantee that Both Invokes Use the Same SQL Connection? 3-5
Does a Scope Form a Transaction Boundary? 3-5

Why Does No Instance Appear in Oracle BPEL Control Upon Failure After I Configured the Process for XA? 3-5
How Do You Configure the Transaction participate Property? 3-5
How Do You Throw a Rollback Fault in BPEL? 3-5
How Do I Learn More about XA? 3-5
How Do I Make a Database Adapter Receive (Polling) and Invoke Part of the Same XA Transaction? 3-5
Database Adapter Cannot Survive Database Restarts 3-6
Inbound Polling Gets Stuck and Cannot Close Connection Exceptions 3-6
Not Automatically Retrying Invokes After Remote Faults 3-6
Exception Misclassification 3-8
Missing Instances or Instances Accessible from the Perform Manual Recovery Option. 3-8
Explicit Schema Name References Causing Problems 3-9
Missing Query By Example Support in 10.1.3 3-9
Foreign Key and Primary Key Constraints with 1-1 & M-M Master-Detail Scenarios 3-10
The Delete Polling Strategy Cannot Effectively Be Used 3-10
Primary Key Constraints With Merge 3-11
Configuring Private Ownership 3-11
Printing Thread on Log Messages 3-12
Database Adapter Tuning Documentation 3-12
Troubleshooting 3-12
Find the Root Exception 3-12
What Exception Do I Look For? 3-13
Finding Out What Is Occurring 3-13
DB2 Issue: Journaling Disabled 3-13
Iterating through the Rows Returned by the Database Adapter 3-14
File Adapter 3-16
Is There Still a 7 MB File Size Limit for the File Adapter? 3-16
Can the File Adapter Be Used to Access Windows Network Folders? 3-16
How Do I Append to an Existing File with the File Adapter? 3-17
How Do I Read the File Name of the Generated File from a Write Operation in BPEL? 3-17
How Do I Control the Size of a Rejected Message in the File Adapter? 3-17
Does the File Adapter Support the Debatching of Large XML files? 3-17
Messages Are Lost When Using Time-Pattern in the Outbound Partner Link File Name Because Messages Created with the Same Timestamp Overwrite One Another 3-18
Can I Be Notified When the File Adapter is Done Debatching a File? 3-18
How Do I Set Up the File Adapter in a High Availability Environment? 3-19
How Do I Configure the Number of Threads for the File Adapter? 3-19
How Do I Guarantee the Ordering of Messages Processed by the File Adapter and BPEL 3-19
FTP Adapter 3-21
Will All File Adapter Features Work with the FTP Adapter? 3-21
Does the FTP Adapter Support the ACTIVE Mode of Connections? 3-21
How Can I Set Up the FTP Adapter to Use Connection Pooling? 3-21
Can I Have an FTP Adapter Partner Link for an Invoke Operation to Retry If Connection-Related Errors Exist? 3-21
JMS Adapter 3-22
How Do I Create a JMS Dequeue Operation in a Global Transaction? 3-22

XA Configuration 3-22
Configuring JMS Adapter for Global Transaction Participation 3-22
Both Invokes in the Same Global Transaction 3-23
Failure Must Cause Rollback........ 3-23
How Do You Configure Transaction Participation?........ 3-23
Related Questions 3-24
Why Do I Receive a 'CCI Local Transaction COMMIT failed' Error When Using XA?........ 3-24
Does the JMS Adapter Work with the WebLogic JMS Provider? 3-25
WebLogic JMS (For All Tested Versions) 3-25
WebLogic 9.*x* JMS Specific Configuration........ 3-26
Adapter Framework 3-26
Changing Inbound Adapter Parameters at Runtime........ 3-27
How Do I Throttle Inbound Message Flow? 3-27
Adapter Clustering — JGroups Configuration 3-28
Inbound Message Flow Multiplexing (Server Distribution) 3-29
Scheduling Endpoint Activation........ 3-29
Tuning JCA Connection Cache 3-30
Adapter SDK 3-31
Rejection Handlers 3-32
File System-Based Rejection Handler 3-32
RAW Oracle Advanced Queue-Based Rejection Handler 3-32
BPEL Process Rejection Handler 3-32
WSIF-Based Rejection Handler 3-33
Adapter Fatal Error Failover BPEL Process 3-33
Dynamic BPEL Process Routing Support by the Adapter Framework 3-34
Motivation........ 3-34
Example Configuration 1 3-34
Example Configuration 2 3-35
Enabling Debug Logging for Detailed Troubleshooting........ 3-37

4 Oracle BPM Human Workflow

Using the Workflow Services API Reference 4-1
Configuring the Workflow Services API Reference........ 4-1
Java Interface that Exposes the Methods to Get the Task List, Task Details, and Update Outcomes........ 4-2
Implementation and Usage in main() Method 4-3
Task Utility that Prints Payload Contents 4-6
Workflow Modeling 4-7
Routing 4-7
Chaining 4-8
Callbacks........ 4-8
Escalations........ 4-9
Notifications........ 4-9
Resource Bundles 4-9
Attachments 4-10

5 Oracle B2B

E-Commerce 5-1
Vision 5-1
Misconceptions 5-2
Concepts: Components 5-2
Concepts: Acknowledgments Types 5-2
Oracle B2B 5-3
How Does Oracle B2B Fit into Oracle SOA Suite? 5-3
Typical Topology 5-4
Use Case: Outbound Purchase Order 5-4
Interoperability 5-6
Routing ID/Consumer 5-6
IP_MESSAGE_TYPE 5-6
Adapter Service 5-7
Vertical Required Data 5-7
Errors Diagnostics 5-8
Trading Partner Identification 5-8
Typical Oracle B2B Errors 5-9
Inbound Direction 5-9
Outbound Direction 5-9
General Check List 5-10
Oracle Technology Network Documentation 5-10
Typical Errors 5-11

6 Oracle Business Activity Monitoring

How Do I Configure Oracle BAM with the AQ Adapter? 6-2
How Do I Configure Oracle BAM with IBM WebSphere MQ Messages? 6-2
How Do I Configure Oracle BAM with OC4J Version 10.1.3? 6-2
How Do I Design BPEL Process Sensors to Send Events to Oracle BAM? 6-3
How Do I Design BPEL Process Sensors for Oracle BAM Sensor Actions? 6-3
How Do I Design and Use BPEL JMS Sensors to Publish Events to Oracle BAM? 6-3
How Do I Configure Oracle BAM to Read Data from External Data Sources? 6-4
How Do I Read External Tables into Oracle BAM Using Enterprise Link Plans? 6-4
How Do I Configure Oracle Enterprise Manager with Oracle BAM? 6-4
How Do I Beautify Reports with Good Color Schemes? 6-5
How Do I Build a Simple Report? 6-5
How Do I Create a Custom HTML Body in a Report for Custom Display? 6-5
What are the Oracle BAM View Types? 6-5
How Do I Design a Report Showing the Ratio of Two Aggregated Values? 6-5
How Do I Design Predictive Alerts? 6-6
How Do I Call External Web Services through Oracle BAM? 6-6
How Do I Call External URLs? 6-6
How Do I Collect and Parse Incoming Data? 6-6
How Do I Configure a Plan to Run Continuously? 6-7
What are the Deployment Recommendations? 6-7
How Do I Configure Oracle BAM for LDAP Authentication and Authorization? 6-7
How Do I Migrate Oracle BAM to Different Environments? 6-7

What are the Best Practices for Oracle BAM Repository Maintenance? 6-8
How Do I Use Oracle BAM without Enterprise Link? 6-8

7 Oracle Data Integrator

Repositories 7-1
Number of Database Schemas for the Repositories 7-1
ID of the Work Repository 7-2
Name of the Work Repository 7-2
Number of Work Repositories 7-2
Types of Repositories (Development versus Execution) 7-2
Promotion from Development to QA and from QA to Production 7-2
Database to Use for the Repositories 7-3
Remote Connectivity to the Repositories from the Graphical User Interface 7-3
Agents 7-3
Parameters for the Agent 7-3
Starting the Agent 7-4
Agent Location 7-4
Sizing for the Agent 7-4
Agent Overload 7-4
Agent Connectivity to the Repository 7-4
JDBC Drivers 7-5
Version of the JDBC Drivers 7-5
JDBC Driver Types 7-5
Topology 7-5
Definition of the Data Servers 7-6
Connectivity to the Databases 7-6
User Name to Connect to a Database 7-6
Default Schemas 7-6
Creation of a New Technology 7-6
Test Agent Connectivity 7-7
Designer 7-7
Markers 7-7
Database Sequence Versus ODI Sequence 7-7
Metadata Project 7-8
Modification of the Knowledge Modules 7-8
Choosing the Appropriate Knowledge Module 7-8
Hardcoded Table Names 7-8
Documentation 7-9
Complex and Reusable Transformations 7-9
Number of Interfaces Per Folder 7-9
Name of the Interfaces 7-9
XML or JMS/XML Synchronization 7-9
Commit Multiple Interfaces Only If the Last One Is Successful 7-10
Variables in a Package 7-10
Contexts in the Interfaces, Knowledge Modules, and Procedures 7-10
Operator 7-10
Logs Purge 7-10

Operations 7-11
ODI Version Migrations 7-11
Real-Time Scenarios: Definition of a Looping Process 7-11

8 Oracle SOA Suite Security

Integrating with External LDAP Providers 8-1

Part II Oracle SOA Suite Performance

9 Best Practices for a JMS-to-Database Scenario

Overview 9-1
Requirements 9-2
Functional Requirements 9-2
Performance Requirements 9-2
Functional Design 9-2
Description 9-3
Process Map 9-3
Activity Map 9-5
Pseudocode 9-5
Performance Configurations 9-6
Architecture 9-6
Optimize the Enqueue Application (Data Pump) 9-8
Objectives 9-9
Best Practices 9-9
Optimize the Inbound Queue 9-9
Objectives 9-9
Best Practices 9-10
Optimize the JMS Providers 9-11
Objectives 9-12
Best Practices 9-12
Optimize the Inbound JMS Adapter 9-13
Objectives 9-13
Best Practices 9-13
Optimize the BPEL Process 9-16
Objectives 9-16
Best Practices 9-16
Optimize Oracle BPEL Server and OC4J 9-18
Objective 9-18
Best Practices 9-18
Optimize the Outbound Layer 9-19
Objectives 9-19
Best Practices 9-19
Create the BPEL Cluster 9-20
Objectives 9-20
Best Practices 9-20
Other Recommendations 9-21

Test Maximum Inbound Queue Throughput with Java Programs 9-21
Test Maximum Oracle BPEL Process Manager and Adapter Throughput with a Preloaded Queue 9-21
Performance Tuning 9-22
Final Numbers 9-22
Measurement 9-22
Throughput by Orders 9-22
Throughput by BPEL Instances 9-22
Performance Values by Configurations 9-23

10 Oracle BPEL Process Manager Performance

Performance Concepts and Terminology 10-1
Breakpoint Activities 10-2
Idempotent Activities 10-2
Durable Versus Transient Processes 10-2
One-Way Invocation Versus Two-way Invocation (Synchronous And Asynchronous Processes) 10-2
Dehydration 10-3
Delivery Service 10-4
Threads and Transactions 10-5
Threading and Transactions for an Asynchronous Durable Process 10-5
Threading and Transactions for a Synchronous Transient Process 10-6
Relationship Among Some Performance Settings 10-7
Using the File Adapter to Process Large File Sizes 10-7
Overview 10-8
Use Case and Requirements 10-8
Initial Design and Problems 10-8
Diagnosis 10-9
Best Practices 10-9
Debatch the Incoming File with a Proper Native Schema 10-9
Structure of the Inbound Data File 10-9
Define the Native Schema File 10-10
Sample XML Message 10-11
Set the Message Publish Size to a Proper Value 10-14
Avoid Unnecessary Updates of Large Size Payloads 10-14
Final Numbers 10-15
Configuring Your Environment to Optimally Process Large Payloads 10-15
Functional Requirements 10-15
Performance Configuration 10-15
Optimize the Message Sending Application 10-15
Optimize the BPEL Server 10-15
Create a BPEL Cluster 10-17
Optimize the BPEL Process 10-17
Objectives 10-17
Best Practices 10-17
Optimize the Dehydration Store Database 10-18
Objectives 10-18

Best Practices 10-18
Benchmark and Architecture 10-19
What is a Typical Throughput (Transaction Rate) for Oracle BPEL Server? 10-19
What is the Maximum Number of Activities in a BPEL Process? 10-19
Is Oracle BPEL Process Manager Designed to Support Synchronous Processes? 10-19
Clustering, Load Balancing, Failover, and Recovery 10-19
Do I Use Oracle Application Server Clustering or Oracle BPEL Process Manager Clustering? 10-20
How Does a Oracle BPEL Process Manager Cluster Work with DCM/OPMN? 10-20
I Have Problems Clustering Oracle BPEL Process Manager on Top of an Oracle Application Server Cluster 10-20
Can I Cluster an Oracle BPEL Process Manager Process to Two Domains of the Same Oracle BPEL Server, Instead of Two Oracle BPEL Servers? 10-20
Can I Add New Nodes to Oracle BPEL Process Manager Clustering without Shutting Down the Dehydration Database and other Oracle BPEL Process Manager Nodes? 10-20
Must I Trade Performance with Reliability? 10-21
When a BPEL Node Crashes, How Do the Other Nodes Get the Instance and Continue . 10-21
How Do I Perform a Manual Recovery? 10-21
Where Do Incoming Messages That Do Not Invoke BPEL Processes or Display in Oracle BPEL Control Go? 10-22
What Causes the Asynchronous Invoke or Callback Messages to Permanently Remain at the Recovery page of Oracle BPEL Control? 10-22
Do Synchronous Messages Display in the Recovery Page? 10-23
Stability and Memory 10-23
What Causes an Out-of-Memory Error? 10-23
Performance Tuning 10-23
Is a Dehydrated BPEL Process Instance Using Oracle BPEL Process Manager Resources? 10-24
Do Synchronous Processes Dehydrate? 10-24
Can I Improve Asynchronous Process Performance by Disabling Dehydration? 10-24
Does Disabling completetionPersistPolicy Cause Out-Of-Memory Errors Because Everything Remains in Memory Instead of Being Dehydrated? 10-24
Transactions Related 10-24
If Top Level Processes Invoke Second Level Subprocesses, Can the Main Process Propagate a JTA Transaction Context to the Subprocesses? 10-24
Dehydration Store Management 10-25
The Dehydration Database is Running Out of Tablespace. Is it Safe to Clean Up the Dehydration Store? 10-25
How to I Clean Up the Dehydration Store? 10-25
What Should I Do If I Receive Error ORA-01691 in the domain.log File? 10-25
What Should I Do If I Receive Error ORA-05002 in the domain.log File? 10-25
Load Throttling 10-26
How Do I Throttle the Load in BPEL Processes? 10-26

11 Oracle Human Workflow Performance

Minimizing Client Response Time 11-1
Choose the Right Workflow Service Client 11-1

Narrow Qualifying Tasks Using Precise Filters 11-2
Retrieve Subset of Qualifying Tasks (Paging)........ 11-2
Fetch Only the Information That Is Needed for a Qualifying Task........ 11-3
Reduce the Number of Return Query Columns........ 11-3
Tune the Identity Provider 11-4
Tune the Database........ 11-4
Improving Server Performance 11-4
Archive Completed Instances Periodically 11-5
Use Java Callbacks Instead of BPEL Callbacks........ 11-5
Suppress Logging and Debugging Messages 11-5
Minimize Notification Overhead........ 11-6
Cluster Nodes 11-6
Completing Workflows Faster........ 11-6
Use Workflow Reports to Monitor Progress........ 11-6
Specify Escalation Rules........ 11-7
Specify User and Group Rules for Automated Assignment........ 11-7
Use Task Views to Prioritize Work........ 11-7

Index

Preface

This guide describes Oracle SOA Suite best practices and common troubleshooting procedures.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This manual is intended for users of Oracle SOA Suite.

Documentation Accessibility

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following Oracle guides, which are available in the Oracle Application Server documentation section of the Oracle Technology Network (OTN):

`http://www.oracle.com/technology/documentation/appserver.html`

- *Oracle BPEL Process Manager Developer's Guide*
- *Oracle BPEL Process Manager Administrator's Guide*
- *Oracle Enterprise Service Bus Developer's Guide*
- *Oracle Enterprise Service Bus Quick Start Guide*
- *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide*
- *Oracle Application Server Adapter Concepts*
- *Oracle Application Server Enterprise Deployment Guide*
- *Oracle Application Server Performance Guide*
- *Oracle SOA Suite Tutorial*
- *Oracle SOA Suite New Features*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, visit OTN. You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

To see additional Oracle SOA Suite best practices technical notes, visit the Oracle SOA Suite Best Practices - Technical Notes site at OTN:

`http://www.oracle.com/technology/tech/soa/soa-suite-best-practices/index.html`

To download Oracle BPEL Process Manager documentation, technical notes, or other collateral, visit the Oracle BPEL Process Manager site at OTN:

`http://www.oracle.com/technology/bpel/`

To download Oracle ESB documentation, technical notes, or other collateral, visit the Oracle ESB site at OTN:

`http://www.oracle.com/technology/goto/esb`

To view frequently asked questions about Oracle BAM 10.1.3, visit the following Oracle BAM site at OTN:

`http://www.oracle.com/technology/products/integration/bam/10.1.3/htdocs/bam_1013_faq.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/`

See the *Business Process Execution Language for Web Services Specification*, available at the following URL:

`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp`

See the *XML Path Language (XPath) Specification*, available at the following URL:

`http://www.w3.org/TR/1999/REC-xpath-19991116`

See the *Web Services Description Language (WSDL) 1.1 Specification*, available at the following URL:

`http://www.w3.org/TR/wsdl`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Oracle SOA Suite Components

This part describes best practices for Oracle SOA Suite components.

This part contains the following chapters:

- Chapter 1, "Oracle BPEL Process Manager"
- Chapter 2, "Oracle Enterprise Service Bus"
- Chapter 3, "Oracle Technology Adapters"
- Chapter 4, "Oracle BPM Human Workflow"
- Chapter 5, "Oracle B2B"
- Chapter 6, "Oracle Business Activity Monitoring"
- Chapter 7, "Oracle Data Integrator"
- Chapter 8, "Oracle SOA Suite Security"

1 Oracle BPEL Process Manager

This chapter provides answers to frequently asked questions about Oracle BPEL Process Manager.

This chapter contains the following topics:

- Threading
- Transactions
- BPEL Process Configuration
- Recovery and High Availability
- Deployment
- Scheduling and Activity Expiration
- Performance
- Flow Logic
- Methodology
- Deployment Descriptor Properties

Note: If the section title does not explicitly specify an Oracle BPEL Process Manager version, the answer is applicable to all versions.

Threading

This section provides answers to frequently asked questions about threading.

This section contains the following topics:

- What is the Relationship Between ReceiverThreads and dspMaxThreads?
- How Do dspMaxThreads and dspInvokeAllocFactor Throttle Requests?
- What are the Different Types and Responsibilities of MDBs in Oracle BPEL Server?

See Also: The Oracle BPEL Process Manager Performance Tuning chapter of the *Oracle Application Server Performance Guide* for additional BPEL threading details

What is the Relationship Between ReceiverThreads and dspMaxThreads?

The `ReceiverThreads` property specifies the maximum number of message-driven beans (MDBs) that can process BPEL requests asynchronously across all domains. Each domain can allocate a subset of these threads using the `dspMaxThreads` property. However, the sum of `dspMaxThreads` across all domains must not exceed the `ReceiverThreads` value.

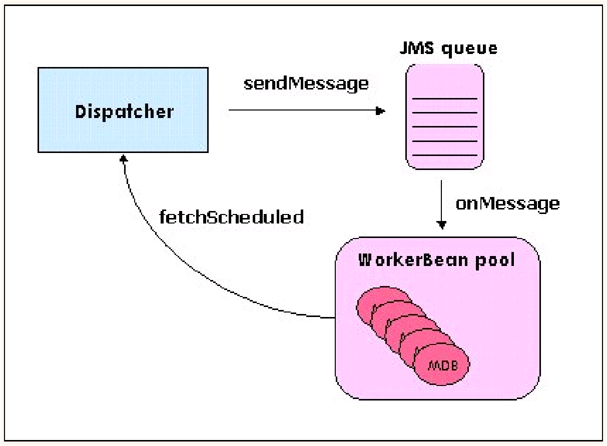
Both properties are set in the `orion-ejb-jar.xml` file. The location for this file is described in "How Do I Modify Transaction Timeout Settings for Oracle BPEL Server?" on page 1-9.

When a domain requires another thread to execute an activity asynchronously, it sends a Java Message Service (JMS) message to a queue. This message is then picked up by a WorkerBean MDB, which requests the dispatcher for work to execute. If the number of WorkerBean MDBs currently processing activities for the domain is sufficient, the dispatcher module may decide not to request another MDB. The decision to request an MDB is based on the following:

- The current number of active MDBs
- The current number pending (that is, where a JMS message has been sent, but an MDB has not picked up the message)
- The value of `dspMaxThreads`

Figure 1–1 provides an overview of the request process.

Figure 1–1 Thread Request Process



Setting both `ReceiverThreads` and `dspMaxThreads` to an appropriate value is important for maximizing throughput and minimizing thread context switching. If there are more `dspMaxThreads` specified than `ReceiverThreads`, the dispatcher modules for all the domains assume there are more resources they can request than actually exist. In this case, the number of JMS messages in the queue continues to grow as long as the request load is high, thereby consuming memory and CPU resources. If the value of `dspMaxThreads` is not sufficient for a domain's request load, throughput is capped. Another important factor to consider is the value for `ReceiverThreads`: more threads does not always correlate with higher throughput. The higher the number of threads, the more context switching the JVM must perform. For each installation, the optimal value for `ReceiverThreads` must be found based on careful analysis of the rate of Eden garbage collections and CPU utilization. For most installations, use a starting value of `40`. This value can be adjusted up or down accordingly. Values greater than `100` are rarely suitable for small to medium sized boxes and most likely lead to high CPU utilization for JVM thread context switching alone.

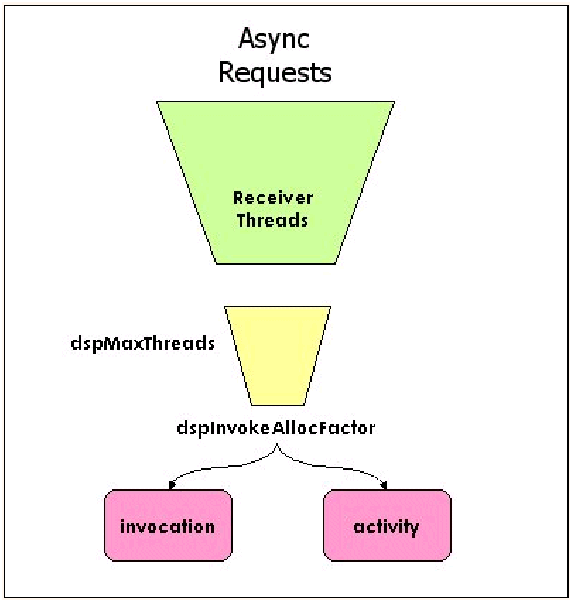
How Do dspMaxThreads and dspInvokeAllocFactor Throttle Requests?

As described in "What is the Relationship Between ReceiverThreads and dspMaxThreads?" on page 1-1, the `dspMaxThreads` property specifies the maximum number of threads a domain's dispatcher module can allocate to execute asynchronous requests. The dispatcher does not actually allocate a thread, but sends a JMS message that is picked up by a WorkerBean MDB. The maximum number of WorkerBean MDBs that executes requests for the dispatcher module is governed by the minimum number of `dspMaxThreads` and `ReceiverThreads` in the `orion-ejb-jar.xml` file.

The `dspInvokeAllocFactor` property controls the number of threads the dispatcher module allocates to execute invocation messages. For example, if `dspInvokeAllocFactor` is `0.4`, then `40%` of the allocated MDBs on average are processing an invocation message. It is important to note that the `40%` allocation ratio is noticeable only when the requests are equally distributed over activity and invocation messages; if there are no invocation messages, the allocated threads are used to process activity messages, and vice versa. The decision of which set of messages to choose from is determined by a random number (`java.util.Random` seeded with `System.currentTimeMillis()` when Oracle BPEL Server is started). If the randomly generated float value is less than `dspInvokeAllocFactor`, the invocation message set is selected. Otherwise, the activity message set is selected. The `dspInvokeAllocFactor` property is accessible from Oracle BPEL Control under **Manage BPEL Domain** > **Configuration** or the `domain.xml` file for your domain.

Figure 1–2 provides an overview of the request process.

Figure 1–2 Thread Request Process



The `dspInvokeAllocFactor` property is primarily used to control the rate at which new instances are created. If a low instance creation rate and high activity throughput are desired, use a low `dspInvokeAllocFactor` value. This setting limits the number of invocation messages picked up by WorkerBean MDBs by retasking these MDBs with activity execution. Again, if the number of activity messages generated is not high, the allocated WorkerBean MDBs execute invocation messages instead; they do not stay idle. Note that even if a `dspInvokeAllocFactor` value of either `0` or `1` is used, invocation and activity messages still get executed (the former when there are no activity messages, and vice versa). Still, it is not advised that such extreme settings be used if the request load is more uniform. Synchronous invocations are not processed asynchronously by the dispatcher. Therefore, there is no throttling mechanism available for these type of requests. Synchronous invocation requests from the client (either from Oracle BPEL Control or through SOAP) are typically executed by HTTP servlet threads, not WorkerBean MDBs. If a synchronous BPEL process is called from an asynchronous BPEL process, the executing thread is a WorkerBean MDB, but the synchronous request does not go through the dispatcher module.

What are the Different Types and Responsibilities of MDBs in Oracle BPEL Server?

Oracle BPEL Server uses two types of MDBs:

- WorkerBean — the primary executor of asynchronous requests in Oracle BPEL Server. When the dispatcher module for a domain must execute an activity asynchronously, it allocates a WorkerBean (through a JMS message, see "How Do dspMaxThreads and dspInvokeAllocFactor Throttle Requests?" on page 1-3) from the pool to perform the task. Tuning the number of `ReceiverThreads` for WorkerBeans is very important for maximizing throughput and minimizing JVM thread context switching.
- InvokerBean — the mechanism with which nonblocking invocations are carried out. When a synchronous invoke activity is executed, the caller thread blocks until a response is received from the partner. In certain situations, such as a flow with a synchronous invoke on several branches, it is undesirable for the execution of one invoke to wait until the previous invoke is finished. In this case, the invoke activities can be scheduled to be performed asynchronously by the InvokerBean MDB. Tuning the number of `ReceiverThreads` for InvokerBean is typically not necessary unless a large number of nonblocking invocations are required.

Transactions

This section provides answers to frequently asked questions about transactions.

This section contains the following topics:

- What are the Transaction Boundaries in a Flow?
- How Do I Force a Rollback in a BPEL Flow?
- What is the Impact of the transaction=participate Property on BPEL Transactionality?
- Why Does the Instance of an Invoked Process Not Display in Oracle BPEL Control?
- Can I Receive a Transaction Timeout If Invoking a Synchronous Transient Process?
- How Do I Modify Transaction Timeout Settings for Oracle BPEL Server?
- What are the Transaction Semantics with Oracle ESB - Oracle BPEL Process Manager Calls or Oracle BPEL Process Manager - Oracle ESB Calls?

What are the Transaction Boundaries in a Flow?

Oracle BPEL Server always executes requests within a JTA transaction. If a JTA transaction is present within the context when the request reaches Oracle BPEL Server, the server’s local transaction is enlisted into the global transaction. If a JTA transaction does not exist, one is created.

Oracle BPEL Server completes its local transaction either when it encounters a breakpoint activity (that is, a receive, onMessage, wait, or onAlarm activity) or reaches the end of the flow. Note that the JTA transaction must not necessarily be committed at this point. If the JTA transaction existed prior to the Oracle BPEL Server request, the transaction is committed when the initiator commits.

Figure 1–3 provides an overview of transaction boundaries.

Figure 1–3 Transaction Boundaries in a Flow

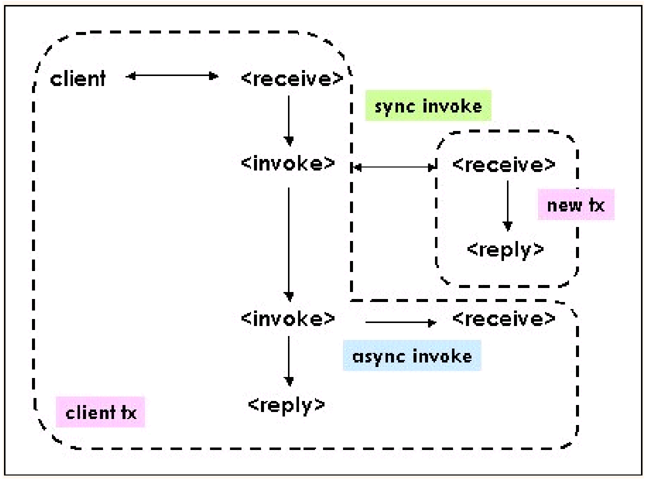


Figure 1–3 shows following actions:

- The client initiates the JTA transaction, either using the JTA Java API or specifying a transaction attribute in an EJB that subsequently calls the BPEL Locator API.
- If the `transaction=participate` property is specified in the `NormalizedMessage` header, the BPEL local transaction enlists itself in the incoming JTA transaction. This is performed by the EJB layer wrapped by the Locator API. Note that the `transaction=participate` setting is not the default. Therefore, a new transaction is normally created at this step.
- The invoke to the synchronous BPEL process is done within its own local transaction. This is done by specifying a transaction attribute of `RequiresNew` in the Oracle BPEL Server EJB method that handles the synchronous invocation.
- When the synchronous BPEL subprocess call returns, the inner transaction has been committed. Regardless of the result of the client JTA transaction, the synchronous BPEL flow is visible from Oracle BPEL Control.
- The invoke to the asynchronous BPEL process is performed within the client JTA transaction. The asynchronous BPEL process is not actually created here, but the invocation message is delivered to the delivery service module and subsequently stored in the `invoke_message` database table.
- The thread exits the BPEL EJB layer and returns to the client. When the client JTA transaction commits, the BPEL synchronous instance and asynchronous invocation message are committed to the database. If the JTA transaction is rolled back, the only object successfully inserted in the database is the synchronous BPEL subprocess.

How Do I Force a Rollback in a BPEL Flow?

There are two mechanisms to force a rollback of a BPEL JTA transaction:

- Explicitly — throw a `bpelx:rollback` fault within your flow:

  ```
  <throw name="Throw" faultName="bpelx:rollback"/>
  ```

- Invoke a partner link that marks the JTA transaction for rollback only.
  - For example, in Oracle SOA Suite for 10.1.3.1.0, if Oracle BPEL Process Manager invokes an Oracle Enterprise Service Bus (ESB) flow that in turn fails to call a Web service endpoint, the JTA transaction is marked for rollback. Since the Oracle ESB flow enlists its local transaction in the JTA transaction, a rollback on the JTA transaction impacts Oracle BPEL Process Manager's ability to dehydrate (as the JTA transaction is used by Oracle BPEL Process Manager for persistence).
  - The above-mentioned scenario is also possible if an EJB with transaction attribute `Required` is invoked. Any modifications to the JTA transaction status impact Oracle BPEL Process Manager's dehydration status.
  - In the 10.1.3.1.0 version of the database adapter, any error thrown from the database causes the TopLink layer to mark the JTA transaction for rollback. The types of errors that can cause this behavior are a primary key violation, a full tablespace, and so on. Any errors that result from attempting to connect to the database from the TopLink layer do not cause the transaction to be rolled back. For example, an incorrect connect URL does not cause a rollback.
  - In the 10.1.3.3.0 version of the database adapter, the TopLink layer has been changed to *not* mark the transaction for rollback if the integrity of the transaction has not been compromised. For example, if a single insertion fails, the transaction is not marked for rollback. However, if in a sequence of two inserts, the first one succeeds but the second one fails, the transaction is rolled back because not doing so breaks the TopLink unit-of-work concept.

What is the Impact of the transaction=participate Property on BPEL Transactionality?

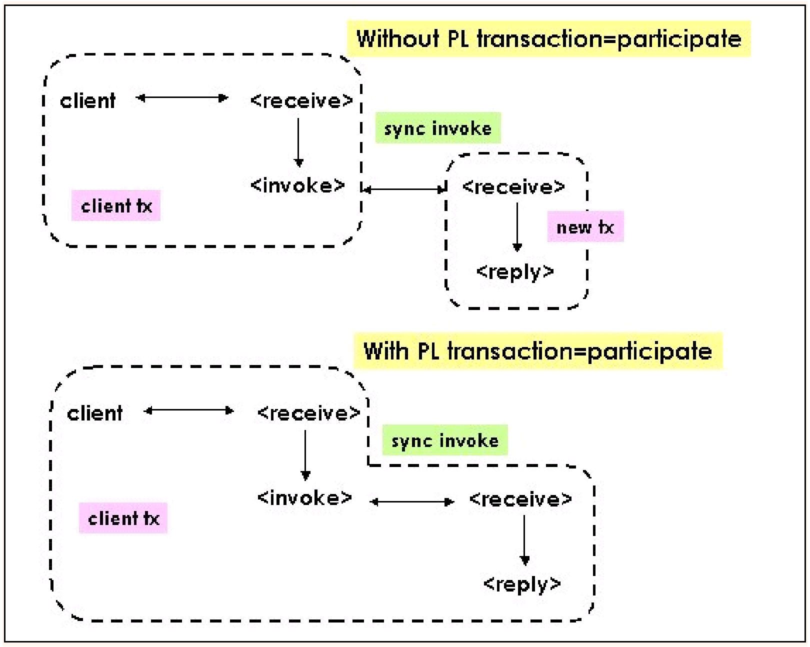
The `transaction=participate` property can be applied in two contexts:

- As a partner link property for a synchronous BPEL subprocess invocation. This has the impact of including the BPEL subprocess in the parent JTA transaction. The default behavior is for synchronous BPEL subprocess invocations to be performed in their own transaction (think of the EJB transaction attribute `RequiresNew`). See "What are the Transaction Boundaries in a Flow?" on page 1-4 for more information.

  ```
  <BPELSuitcase>
  <BPELProcess id="BankTransferFlow" src="BankTransferFlow.bpel">
  <partnerLinkBindings>
  <partnerLinkBinding name="client">
  <property name="transaction">participate</property>
  </partnerLinkBinding>
  </partnerLinkBindings>
  ...
  </BPELProcess>
  </BPELSuitcase>
  ```

  Figure 1–4 provides an overview of this property as a partner link property.

Figure 1–4 transaction=participate Property as a Partner Link Property

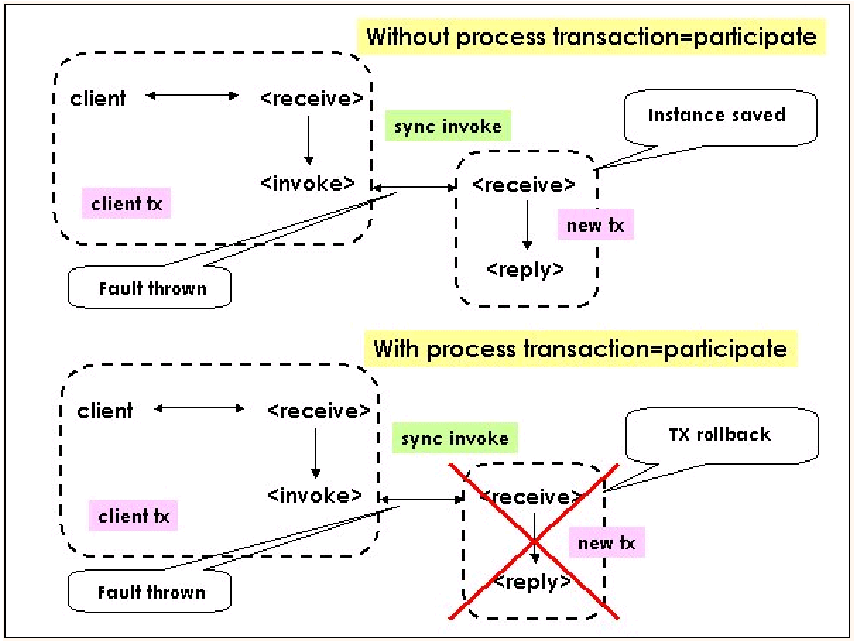


- As a process property. This marks the local transaction for rollback. If this property is not specified, the BPEL subprocess instance is closed, faulted, and the instance is persisted and visible from Oracle BPEL Control. If this property is specified, the BPEL subprocess instance is not persisted and is not visible from Oracle BPEL Control.

```
<BPELSuitcase>
<BPELProcess id="BankTransferFlow" src="BankTransferFlow.bpel">
...
<configurations>
<property name="transaction">participate</property>
</configurations>
</BPELProcess>
</BPELSuitcase>
```

Figure 1–5 provides an overview of this property as a process property.

Figure 1–5 transaction=participate Property as a Process Property



As mentioned in the second point, if the caller partner link specifies `transaction=participate` and the subprocess also specifies `transaction=participate`, the subprocess rolls back the client JTA transaction.

Why Does the Instance of an Invoked Process Not Display in Oracle BPEL Control?

If the process invoked is asynchronous, the message headers and payload are stored in the `invoke_message` table. The invocation message is then processed by a background thread to create an asynchronous BPEL process instance. If an error is encountered while processing the invocation message, the transaction is rolled back and the invoke message remains in its unhandled state in the `invoke_message` table. See "How Does an Invoke Message and Activity Display in the Manual Recovery Page?" on page 1-18 for more details on this scenario.

If the process invoked is synchronous, the cause for the missing instance is likely a transaction rollback. The reasons for a transaction rollback are as follows:

- Third party manipulation of the JTA transaction. As described in "How Do I Force a Rollback in a BPEL Flow?" on page 1-5, a failed database adapter invocation can cause the TopLink layer to mark the JTA transaction for rollback. Because Oracle BPEL Server relies on the JTA transaction for persistence (not just as a signaling mechanism between transaction-aware modules), any changes to the status of the transaction impact Oracle BPEL Server's ability to dehydrate the instance.
- Transaction timeout. The synchronous process may be invoking one or more backend synchronous services; the total time of all the backend calls may exceed the timeout value for the transaction. See "How Do I Modify Transaction Timeout Settings for Oracle BPEL Server?" on page 1-9 for details on how to increase the transaction timeout value.
- Failure to dehydrate the synchronous instance to the database. If the `completionPersistPolicy` is set to `on`, the instance attempts to save itself before returning to the caller. Typically, errors are encountered during this step if the database connection has been severed or the tablespace is full. Some customers

have encountered problems with saving the audit trail of the instance. For example, there are problems with JDBC batching in 10.1.3.1.0 that the audit trail persistence code uses. When a synchronous invocation does not result in an instance visible from Oracle BPEL Control (assuming that `completionPersistPolicy` is set to a value of `on`, `deferred`, or `faulted`), use the following steps to find the underlying error:

– Enable all loggers to `debug` mode.

– Resubmit the invocation message.

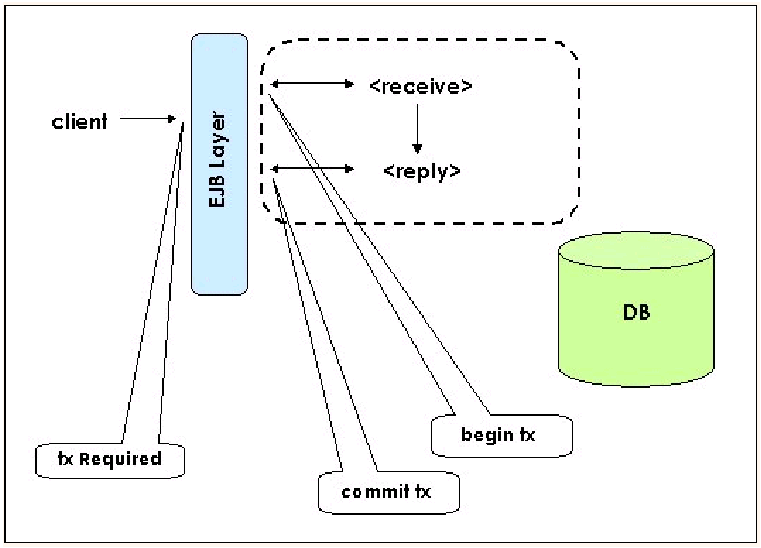
– Check the Oracle Process Manager and Notification Server (OPMN) or command-line log for the underlying exception stack trace.

Can I Receive a Transaction Timeout If Invoking a Synchronous Transient Process?

Yes. If you are invoking a synchronous transient process, even with `completionPersistPolicy=off` and `inMemoryOptimization=true` in the `bpel.xml` file, a JTA transaction is still created. If no transaction context is passed into the BPEL EJB layer, a new JTA transaction is started because EJB transaction attribute `Required` is set on the `EngineBean` method.

Figure 1–6 provides an overview.

Figure 1–6 Transaction Creation



Even if the BPEL process does not dehydrate itself, the transaction with the database has already been started. Once the request returns to the EJB layer, a commit is issued to the database. If the transaction timeout threshold has been reached, a transaction timeout exception is raised, regardless of whether Oracle BPEL Server has actually issued any data manipulation language (DML) statements to the database.

How Do I Modify Transaction Timeout Settings for Oracle BPEL Server?

The default transaction timeout value is `60` seconds. This value may not be sufficient for some synchronous processes. This is typically not because of the length of the BPEL flow, but because of the various endpoints that BPEL calls. Therefore, some tuning of the value may be necessary. To change the transaction timeout setting for Oracle BPEL Server only, follow these steps:

1. Unzip the `ejb_ob_engine.jar` file located under *`SOA_ORACLE_HOME`*`/j2ee/home/applications/orabpel`.
2. Modify the `META-INF/orion-ejb-jar.xml` file inside by changing all values of `transaction-timeout` on all the EJBs.

```
<session-deployment name="CubeEngineBean" min-instances="100"
 transaction-timeout="60" copy-by-value="false"
 location="ejb/collaxa/system/CubeEngineBean">
<resource-ref-mapping name="jms/collaxa/BPELInvokerQueueFactory"
 location="BPELjms/BPELInvokerQueueFactory" />
<resource-env-ref-mapping name="jms/collaxa/BPELInvokerQueue"
 location="BPELjms/BPELInvokerQueue" />
</session-deployment>
```

3. Rejar `ejb_ob_engine.jar` and restart the OC4J container.

An easier approach is to change the transaction timeout for the entire OC4J container. Follow these steps:

1. Remove all the `transaction-timeout` attributes from the `META-INF/orion-ejb-jar.xml` file (see the previous steps for accessing this file).
2. Change the `transaction-timeout` value in *`SOA_ORACLE_HOME`*`/j2ee/home/config/transaction-manager.xml`.

```
<transaction-manager
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/transact
ion-manager-10_0.xsd"
transaction-timeout="30"
max-concurrent-transactions="-1">
```

3. Restart the OC4J container.

This approach makes changing the transaction timeout value easier in the future. However, the changes apply to all applications that have been deployed on the container, not just to Oracle BPEL Process Manager.

What are the Transaction Semantics with Oracle ESB - Oracle BPEL Process Manager Calls or Oracle BPEL Process Manager - Oracle ESB Calls?

This section describes the transaction semantics for calling Oracle ESB — Oracle BPEL Process Manager and Oracle BPEL Process Manager — Oracle ESB in the following versions:

- Behavior in 10.1.3.1.0
- Behavior in 10.1.3.3.0

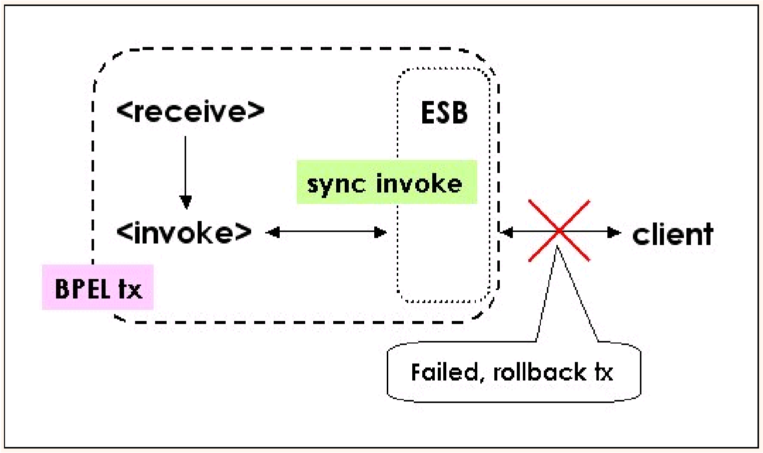
Behavior in 10.1.3.1.0

If Oracle BPEL Process Manager starts a JTA transaction and invokes an Oracle ESB service through the WSIF Oracle ESB binding (as opposed to SOAP, in which case each side has its own transaction), the Oracle ESB service enlists itself in the existing JTA transaction. Positive runs complete as expected, with the Oracle BPEL Process Manager transaction successfully issuing a commit when the Oracle ESB synchronous call returns. Negative cases where Oracle ESB fails to invoke another service result in the JTA transaction being marked for rollback. This action causes Oracle BPEL Process

Manager to fail when attempting to dehydrate the instance and audit trail (see "How Do I Force a Rollback in a BPEL Flow?" on page 1-5).

Figure 1–7 provides an overview.

Figure 1–7 Transaction Semantics



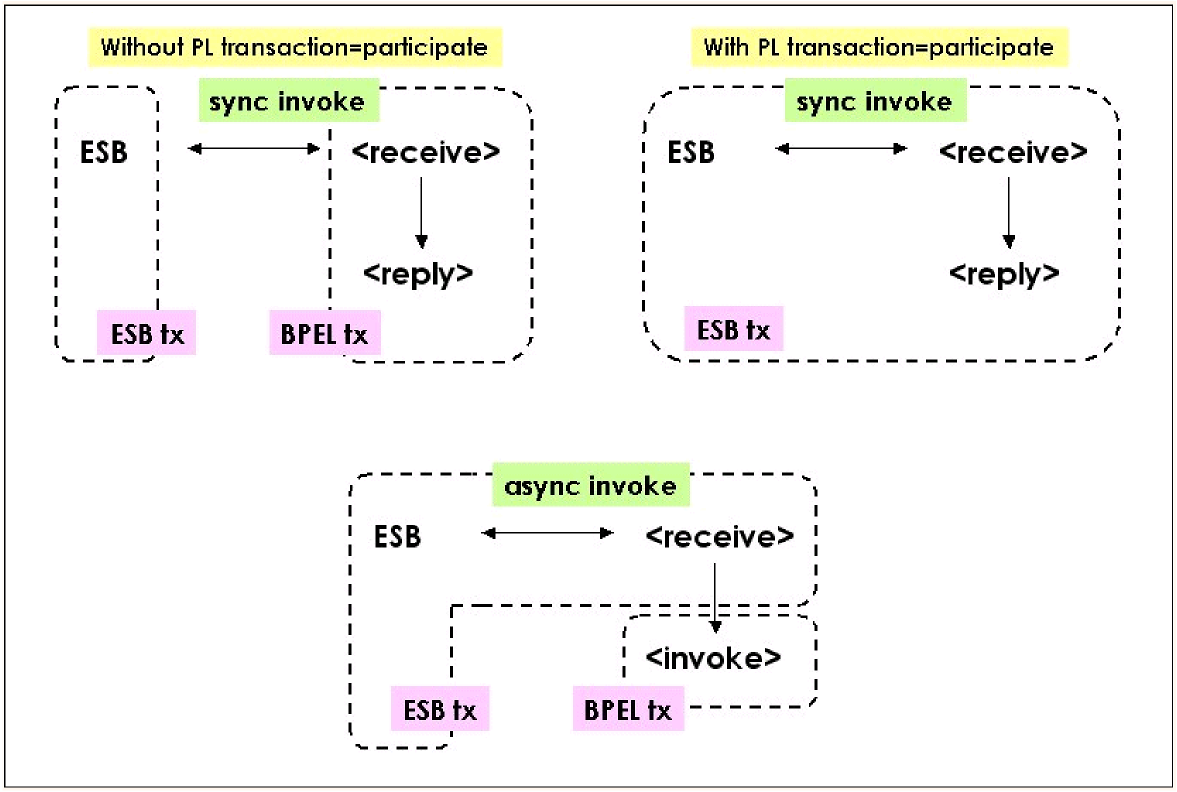
When Oracle ESB is invoking Oracle BPEL Process Manager through the local Java route (again, as opposed to SOAP), the behavior is similar to the scenario described in "What is the Impact of the transaction=participate Property on BPEL Transactionality?" on page 1-6. If Oracle ESB is calling a synchronous Oracle BPEL Process Manager process, in order to enlist the Oracle BPEL Process Manager invocation into the Oracle ESB JTA transaction, the `transaction=participate` property is passed in the `NormalizedMessage` used to invoke the process.

```
Locator l = new Locator( "default", "bpel", props );
IDeliveryService ds = (IDeliveryService) l.lookupService(
 IDeliveryService.SERVICE_NAME );

NormalizedMessage nm = new NormalizedMessage();
nm.addPart( "payload", xml );
nm.setProperty( "transaction", "participate" );
NormalizedMessage res = ds.request( "CreditRatingService", "process", nm );
```

If Oracle ESB is invoking an asynchronous Oracle BPEL Process Manager process, the persistence of the invocation message in the Oracle BPEL Process Manager `invoke_message` table is done within the Oracle ESB JTA transaction (see "What are the Transaction Boundaries in a Flow?" on page 1-4). The processing of the invoke message is done asynchronously by a background thread in a completely separate JTA transaction.

Figure 1–8 provides an overview.

Figure 1–8 Transaction Semantics

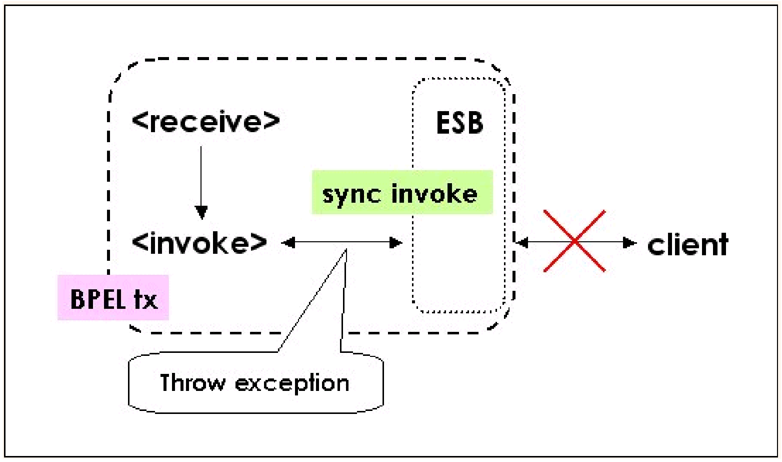


Behavior in 10.1.3.3.0

In 10.1.3.3.0, Oracle ESB no longer marks the JTA transaction for rollback if the transaction was initiated by Oracle BPEL Process Manager. Instead, Oracle ESB throws an exception back to Oracle BPEL Process Manager. Oracle BPEL Process Manager converts the exception into a fault thrown within the Oracle BPEL Process Manager flow. If the fault is handled within the Oracle BPEL Process Manager flow (either through a catch or catchAll branch), Oracle BPEL Process Manager considers the fault handled and subsequent activities following the catch block are executed. If the fault is not handled within the Oracle BPEL Process Manager flow, whether or not the fault is thrown to the caller depends on whether the `transaction=participate` property is specified for the process (see "What is the Impact of the transaction=participate Property on BPEL Transactionality?" on page 1-6).

Figure 1–9 provides an overview.

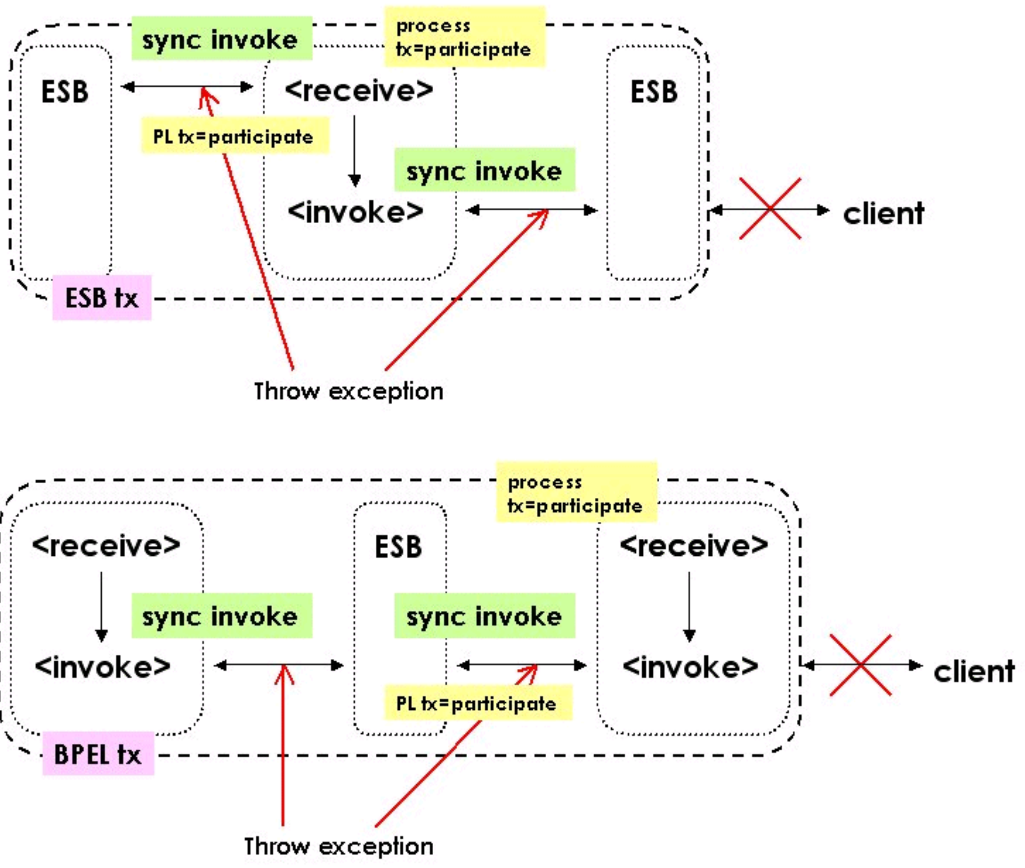
Figure 1–9 Transaction Semantics



More complicated invocation patterns are typically used, such as Oracle BPEL Process Manager — Oracle ESB — Oracle BPEL Process Manager or Oracle ESB — Oracle BPEL Process Manager — Oracle ESB. The same settings described above apply to these cases as well.

Figure 1–10 provides an overview.

Figure 1–10 Transaction Semantics



See Also: "How Do I Force a Rollback in a BPEL Flow?" on page 1-5 for details about how to force a rollback of a BPEL JTA transaction with `bplex:rollback`

BPEL Process Configuration

This section provides answers to frequently asked questions about BPEL process configuration.

This section contains the following topics:

- How Does syncMaxWaitTime Work and Why Do I Not See My Synchronous Requests Timing Out?
- How Does a Nonblocking Invoke Work?

How Does syncMaxWaitTime Work and Why Do I Not See My Synchronous Requests Timing Out?

The `syncMaxWaitTime` property applies to durable processes called in a synchronous manner.

Assume you have a BPEL process with the following definition. The process is not durable because there are no breakpoint activities within it:

```
<receive name="receiveInput" partnerLink="client" variable="input"
 createInstance="yes" />
<assign>
...
</assign>
<reply name="replyOutput" partnerLink="client" variable="output" />
```

If a Java client or another BPEL process calls this process, the assign activity is performed and the reply activity sets the output message into a HashMap for the client (actually the delivery service) to retrieve. Since the reply is the last activity, the thread returns to the client side and tries to pick up the reply message. Since the reply message was already inserted, the client does not wait and returns with the reply.

Assume you have a BPEL process with a breakpoint activity:

```
<receive name="receiveInput" partnerLink="client" variable="input"
 createInstance="yes" />
<assign>
...
</assign>
<wait for="'PT10S'" />
<reply name="replyOutput" partnerLink="client" variable="output" />
```

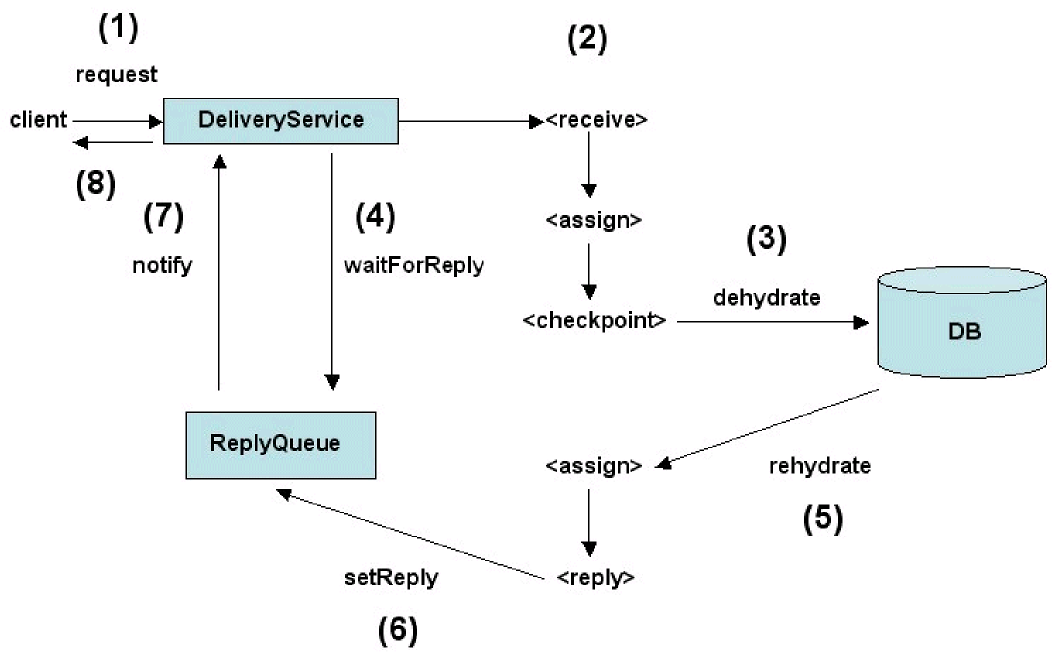
When the client (or another BPEL process) calls the process, the wait (breakpoint) activity is executed. However, since the wait is processed after some time by an asynchronous thread in the background, the executing thread returns to the client side. The client (actually the delivery service) tries to pick up the reply message, but it’s not there since the reply activity in the process has not yet executed. Therefore, the client thread waits for the `syncMaxWaitTime` seconds value. If this time is exceeded, then the client thread returns to the caller with a timeout exception.

If the wait is less than the `syncMaxWaitTime` value, the asynchronous background thread resumes at the wait and executes the reply. The reply is placed in the HashMap and the waiter (the client thread) is notified. The client thread picks up the reply message and returns.

Therefore, `syncMaxWaitTime` only applies to synchronous process invocations when the process has a breakpoint in the middle. If there is no breakpoint, the entire process is executed by the client thread and returns the reply message.

Figure 1–11 illustrates the client thread actions within the request.

Figure 1–11 Client Thread Actions in a Request



The following steps are performed:

1. The client calls a request on the delivery service (this can also be a BPEL process).
2. The synchronous flow starts. In this case, the first activity is a receive.
3. The flow executes some activities, then encounters a checkpoint, which is a breakpoint activity (other breakpoint activities are receive, onMessage, wait, and onAlarm). Once the flow encounters this activity, the instance must be dehydrated. The assign following the checkpoint is processed by another thread in the background.
4. The thread (the one from the client) goes back to the delivery service and waits for the reply from the reply queue. This waiting is subject to the `syncMaxWaitTime` value.
5. The background asynchronous thread picks up where the other thread left off. It rehydrates the instance from the database, and executes the remaining activities.
6. When the reply activity is executed, it puts the reply message in the reply queue.
7. The reply queue now notifies any waiters that the reply message is available. If the waiter has not timed out, it gets the message. If it has timed out, the message is not picked up (note that the flow has completed normally).
8. If the waiter has not timed out, it now has the reply message and returns to the client with the message.

If the flow did not have a checkpoint, then step 2 processes all the activities, including the reply (which added the reply message to the queue). When the thread returns to the delivery service, it waits for the reply. However, since the reply is already there, it returns immediately and is not subject to the `syncMaxWaitTime` value.

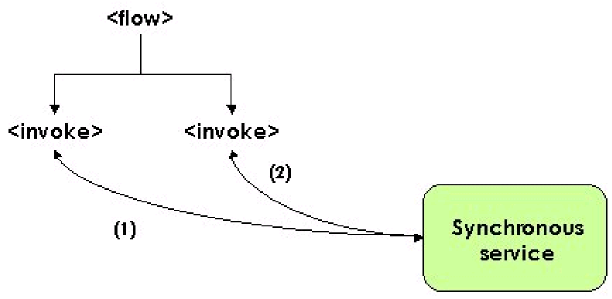
In addition, the synchronous process attempts to save itself when it is finished, and the total processing time has exceeded the transaction timeout, the JDBC save operation causes an exception and the JTA transaction rolls back.

How Does a Nonblocking Invoke Work?

Normally when executing a synchronous (two-way) invoke activity, Oracle BPEL Server waits for the response from the endpoint before executing the following activity. This behavior can present problems when synchronous invoke activities are placed inside a flow activity, because Oracle BPEL Server executes the flow using pseudo-parallelism.

Figure 1–12 provides an overview of this process.

Figure 1–12 Synchronous (Two-Way) Invoke Activity Execution



In Figure 1–12, the second invoke activity is not executed until the response from the first invoke activity is received. If the number of branches in the flow is large, the delay before the final invoke activity is executed is the sum of all the preceding synchronous invoke activities. With a nonblocking invoke, the execution of the synchronous invoke activity is scheduled to be performed by a separate thread in the background. With this change, the initial execution time of the invoke activities is minimal and the overall delay faced by subsequent invoke activities in the flow is reduced.

Figure 1–13 and Figure 1–14 provide an overview of the sequence of events for a nonblocking invoke call.

Figure 1–13 Sequence of Events for a Nonblocking Invoke Call (Part 1)

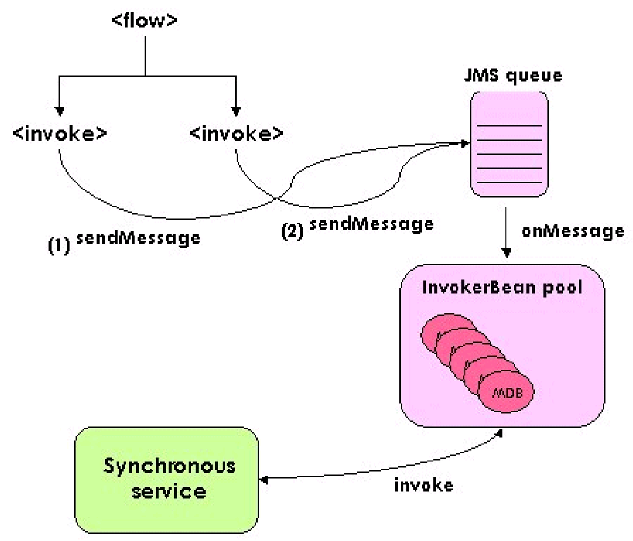
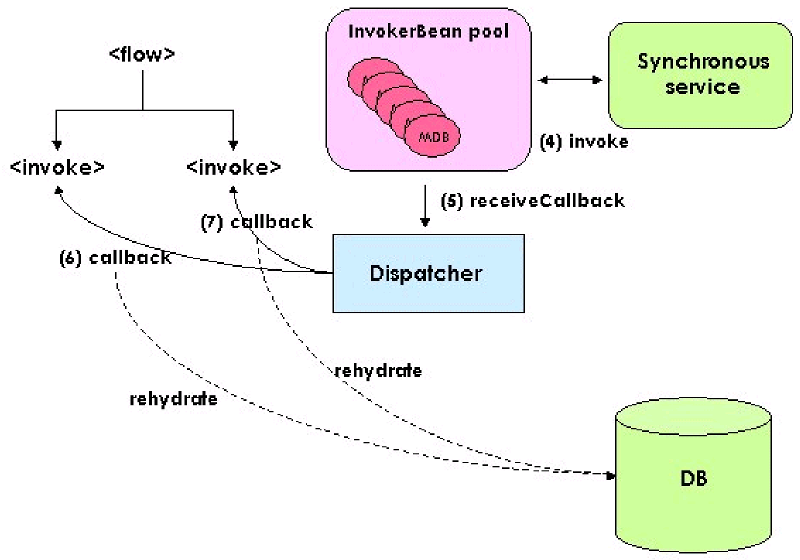


Figure 1–14 Sequence of Events for a Nonblocking Invoke Call (Part 2)



The sequence of events for a nonblocking invoke call is as follows:

1. The first synchronous invoke activity sends a message to a JMS queue. This activity now waits for the asynchronous response and relinquishes control to the next activity.
2. The second synchronous invoke activity sends a message to a JMS queue.
3. Since there are no additional activities to execute, the instance now dehydrates. The invoke activities cannot complete until they receive their callbacks.
4. Two InvokerBean instances pick up the messages in the JMS queue and execute the synchronous invoke against the partner endpoint.

5. The response from the endpoint is cached and a callback message is scheduled with the dispatcher module.
6. The first invoke activity receives its callback. The instance is rehydrated during this step.
7. The second invoke activity receives its callback. The instance is rehydrated during this step.

The sequence of callbacks can vary depending upon the order in which the endpoint response is received. Figure 1–14 shows the first and second invoke callbacks being received in that order, but it can just as easily have been reversed. Other interesting points to note are as follows:

- Use of a nonblocking invoke within a synchronous transient process causes the flow to behave as if it were synchronous durable due to the dehydrate step following the nonblocking invoke. The invoke activities must wait for the callback from the InvokerBean.
- The nonblocking scheduling in steps 1 and 2 is done immediately. There is no synchronization with the existing BPEL JTA transaction. However, the callbacks must go through the dispatcher module and are serialized with respect to access to the instance data.
- Nonblocking invocations are normally used with synchronous (two-way) invocations. However, asynchronous (one-way) invocations can also be treated in the same manner (there is no callback in this case).

Recovery and High Availability

This section provides answers to frequently asked questions about recovery and high availability.

This section contains the following topics:

- How Does an Invoke Message and Activity Display in the Manual Recovery Page?
- How Do I Implement an Automated Recovery Mechanism (10.1.x)?
- How Does Oracle BPEL Server Handle a RAC Node Failover?

How Does an Invoke Message and Activity Display in the Manual Recovery Page?

The manual recovery page in Oracle BPEL Control is simply a query against the `invoke_message` and `work_item` tables for invoke messages and activities, respectively. For invoke messages, the query picks up all messages that have state `0` (undelivered) and have not been modified within a specific threshold (for example, `10` minutes). The threshold prevents messages waiting for processing or currently processing from being selected by the query.

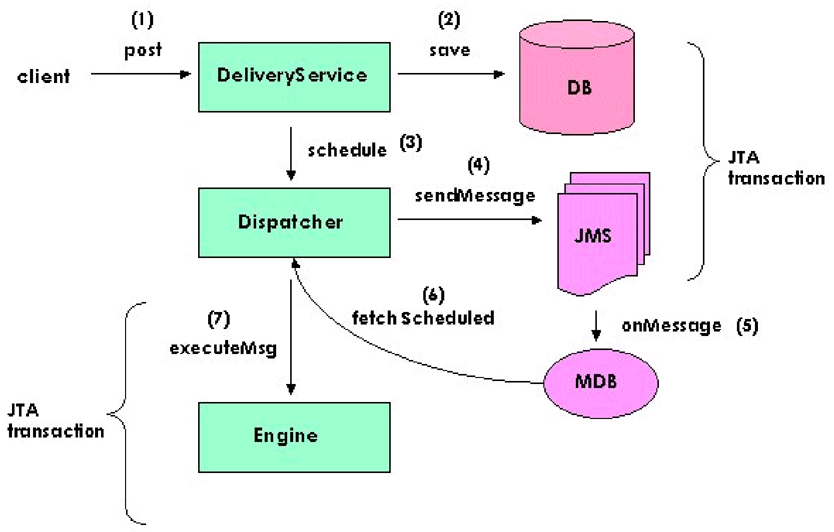
The sequence of events involved in the delivery of invoke messages is as follows:

1. The client posts the message to the delivery service.
2. The delivery service saves the invocation message to the `invoke_message` table. The initial state of the message is `0` (undelivered).
3. The delivery service schedules a dispatcher message to process the invocation message asynchronously. The JTA transaction commits, persisting the invocation message to the database.

4. The dispatcher message is delivered to the dispatcher through the `afterCompletion()` call. Therefore, the message is not delivered if the JTA transaction fails.
5. The dispatcher sends the JMS message to the queue. The message is picked up by MDB.
6. MDB fetches the invocation message from the dispatcher.
7. MDB passes the invocation message to Oracle BPEL Server, which updates the invocation message state to `1` (delivered), creates the instance, and executes the activities in the flow until a breakpoint activity is reached.

Figure 1–15 provides an overview of this process.

Figure 1–15 Sequence of Events for Delivery of Invoke Messages



The following actions typically occur:

- An instance is started from the invoke message.
- The state is set to `1` (handled).
- The activities following the initial receive are executed.
- The instance is dehydrated.

If the instance successfully dehydrates (that is, the second JTA transaction successfully commits), then the invoke message state update is committed. If the transaction is rolled back, then the invoke message state update fails. Therefore, you get an invoke message with state `0` in the manual recovery query. The JTA transaction can be rolled back for several reasons:

- The transaction timeout takes too long to execute. This is normally the result of partner links taking too long to execute.
- A database error (tablespace is full, database connection goes down, and so on).
- The middle tier node goes down.

The underlying error can normally be determined by enabling all the Oracle BPEL Process Manager loggers for the domain to debug and retrying the initial `post()`. If a single thread is unable to reproduce the issue, the problem may be an insufficient transaction timeout setting for the request load. The server transaction setting is specified in the `orion-ejb-jar.xml` file for the `ejb_ob_engine.jar` EJB in the `orabpel` application. Change the setting through either of the following methods:

- Change all the values in `orion-ejb-jar.xml`.
- Remove all the values from the file and specify a single transaction timeout for the entire server in `server.xml`.

As long as the client can send the invoke message to the server, Oracle BPEL Server does not lose any data. Unfortunately, there is no automated recovery mechanism, and you must implement your own recovery daemon using the Oracle BPEL Process Manager APIs (the same APIs that the manual recovery JSP page uses).

See Also: "How Do I Modify Transaction Timeout Settings for Oracle BPEL Server?" on page 1-9 for details about accessing the `orion-ejb-jar.xml` file.

How Do I Implement an Automated Recovery Mechanism (10.1.*x*)?

In 10.1.3.*x*, the only means for recovering failed or stranded invoke messages and activities is from the manual recovery page in Oracle BPEL Control. However, it is possible to implement an automated agent that periodically scans the Oracle BPEL Process Manager dehydration store for failed and stranded objects and resubmits them for processing. The BPEL Locator API already provides standard queries to select candidate invoke messages and activities for recovery and methods to submit these objects to Oracle BPEL Server. Here is a simple Java class that connects to the server using the Locator API and queries invoke messages that must be resubmitted.

```
import java.io.*;
import java.net.*;
import java.util.*;

import com.oracle.bpel.client.*;
import com.oracle.bpel.client.util.*;

public class RecoveryTest
{
    public static void main( String[] args )
        throws Exception
    {
        // Need to load the JNDI context properties to connect to the server
        // via RMI.  The properties are assumed to be located in the local
        // file context.properties.
        //
        // The file contents will resemble:
        // orabpel.platform=oc4j_10g
        //
 java.naming.factory.initial=com.evermind.server.rmi.RMIInitialContextFactory
        // java.naming.provider.url=ormi://localhost/orabpel
        // java.naming.security.principal=admin
        // java.naming.security.credentials=welcome
        //
        Properties props = new Properties();
        URL url = new File( "context.properties" ).toURL();
        props.load( url.openStream() );

        // Assume that arguments passed to the class are: domain_id, domain_
        // password
        Locator l = new Locator( args[ 0 ], args[ 1 ], props );

        WhereCondition wc = WhereConditionHelper.whereInvokeMessagesRecoverable();
        IInvokeMetaData[] imd = l.listInvokeMessages( wc );

        if( imd != null )
        {
            String[] guids = new String[ imd.length ];
            for( int i = 0; i < imd.length; i++ )
            {
                guids[ i ] = imd.getMessageGUID();
            }

            IBPELDomainHandle domain = l.lookupDomain();
            domain.recoverInvokeMessages( guids );
        }
    }
}
```

This class can now be invoked from a `cron` job, or by a Quartz scheduler. Logic can also be added to limit the number of messages to submit at any one time (that is, throttling).

The `where` condition that is used in the sample Java class is as follows:

```
where state = 0
```

However, applying this condition to the query may pick up some messages that are currently being processed by a thread, but whose state change has not been committed to the database. To prevent this from happening, you can choose to select invoke messages that have remained unchanged for a certain time threshold.

```
WhereCondition wc = new WhereCondition( SQLDefs.IM_state + " = " +
                                        IDeliveryConstants.STATE_UNRESOLVED );
wc.append( SQLDefs.IM_receive_date + " > ? " );

// Only pick up invoke messages that are older than 1 hour (3600 seconds)
//
Date receiveDate = new Date( System.currentTimeMillis() - 3600 * 1000 );
wc.setTimestamp( 1, receiveDate );
```

Similar patterns can be applied for callback messages and activities. See the Locator API for details.

How Does Oracle BPEL Server Handle a RAC Node Failover?

When a nonfatal exception occurs (for example, a RAC failover is caught), the transaction is retried against the database. The number of times that this retry is attempted before failing is based on the value specified with the `nonFatalConnectionMaxRetry` property. The `nonFatalConnectionMaxRetry` property is defined in the **Configuration** tab of Oracle BPEL Admin Console. Nonfatal connection errors may be thrown when the dehydration store is a RAC installation and the node to which the application server is pointing is shut down. Oracle BPEL Server resubmits messages that failed as a result of the connection error. The default value for this property is `2`.

Deployment

This section provides answers to frequently asked questions about deployment.

How Do I Resolve a Process Lock Timeout Error?

In 10.1.3.*x*, all interactions with the in-memory process object must obtain a lock before proceeding. For threads that simply must read the process definition, a read or shared lock must be obtained. For threads that must update the process definition, a write or exclusive lock must be obtained. The read lock may be held simultaneously by multiple reader threads, as long as there are no writers. A write lock is exclusive. Operations that require a write lock include:

- Deployment
- Redeployment
- Undeployment
- Change in process descriptor (`bpel.xml`)
- Change in process state, lifecycle, or both

The following example shows a timeout error:

```
<2007-04-30 15:12:23,912> <ERROR> <interfaces.collaxa.cube>
 <BaseCubeSessionBean::logError>
 Error while invoking bean "process manager": Timed out waiting for process load
 lock.
Failed to obtain load lock for process "test_debmas01-1.0"; timed out after
 150,000 seconds.

ORABPEL-05244

Timed out waiting for process load lock.
Failed to obtain load lock for process "test_debmas01-1.0"; timed out after
 150,000 seconds.

at com.collaxa.cube.engine.deployment.LockManager.acquire(LockManager.java:89)
at com.collaxa.cube.engine.deployment.LockManager.acquire(LockManager.java:62)
at

com.collaxa.cube.engine.deployment.DeploymentManager.getProcess(DeploymentManager.
java:351)
at

com.collaxa.cube.engine.deployment.DeploymentHelper.lookupProcess(DeploymentHelper
.java:247)
at

com.collaxa.cube.ejb.impl.BPELProcessManagerBean.getMetaData(BPELProcessManagerBea
n.java:224)
```

To simulate a lock starvation situation, you must have at least two threads attempting to acquire a process lock. One of the threads is attempting to acquire a write lock, while the other is attempting to acquire a read lock. If the writer acquires the lock first and does not release the lock, the reader never acquires the lock, and eventually times out. Normally if the lock timeout duration is sufficient to encompass a lengthy process redeployment, this situation never happens. However, as was noticed at a customer site, redeployment of a process with an SAP adapter may either take an inordinate amount of time, or may never return at all. In this case, the writer never completes, while the list of readers grows until all time out.

```
"AJPRequestHandler-HTTPThreadGroup-54" prio=10 tid=0x002bc6b8 nid=0xe8 in
 Object.wait() [0xacd7c000..0xacd81788]
    at java.lang.Object.wait(Native Method)
    - waiting on <0xf0581ca0> (a
 com.ibi.sap.inbound.SapInboundAdapter$MasterThread)
    at java.lang.Object.wait(Object.java:474)
    at com.ibi.sap.inbound.SapInboundAdapter.activate(SapInboundAdapter.java:223)
    - locked <0xf0581ca0> (a com.ibi.sap.inbound.SapInboundAdapter$MasterThread)
    at com.iwaysoftware.af.container.channel.Channel.start(Channel.java:175)
    at
 com.iwaysoftware.afjca15.inflow.EndpointConsumer.start(EndpointConsumer.java:101)
    at
com.iwaysoftware.afjca15.AbstractResourceAdapter.endpointActivation(AbstractResour
ceAdapter.java:183)
    - locked <0xefe26138> (a java.util.HashMap)
    at
com.iwaysoftware.afjca15.IWAFOracleResourceAdapter.endpointActivation(IWAFOracleRe
sourceAdapter.java:287)
    at
oracle.tip.adapter.fw.jca.AdapterFrameworkImpl.endpointActivation(AdapterFramework
Impl.java:541)
    at
oracle.tip.adapter.fw.agent.jca.JCAActivationAgent.performEndpointActivation(JCAAc
tivationAgent.java:1086)
    at
oracle.tip.adapter.fw.agent.jca.JCAActivationAgent.activateInboundJcaEndpoint(JCAA
ctivationAgent.java:1069)
    - locked <0xb92d3b38> (a oracle.tip.adapter.fw.agent.jca.JCAActivationAgent)
    at
oracle.tip.adapter.fw.agent.jca.JCAActivationAgent.initiateInboundJcaEndpoint(JCAA
ctivationAgent.java:974)
    at
oracle.tip.adapter.fw.agent.jca.JCAActivationAgent.onLifeCycleChanged(JCAActivatio
nAgent.java:735)
    at
com.collaxa.cube.engine.core.BaseCubeProcess$ActivationObserver.update(BaseCubePro
cess.java:1002)
    at
com.collaxa.cube.engine.observer.DomainObserverRegistry.notify(DomainObserverRegis
try.java:316)
    at
com.collaxa.cube.engine.observer.DomainObserverRegistry.notify(DomainObserverRegis
try.java:85)
    at
com.collaxa.cube.engine.observer.DomainObserverHelper.onProcessLifecycleChange(Dom
ainObserverHelper.java:109)
    at
com.collaxa.cube.engine.deployment.DeploymentManager.updateMetaData(DeploymentMana
ger.java:1646)  (LOCK!)
    at
com.collaxa.cube.ejb.impl.BPELProcessManagerBean.updateMetaData(BPELProcessManager
Bean.java:255)
```

Resolve a lock timeout error as follows:

- Increase the lock timeout value through the Java system property `orabpel.process.lock.timeout`. This property can be set from the OPMN start command by specifying `-Dorabpel.process.lock.timeout=300` (the unit is seconds).
- If you still receive a lock timeout even with extremely large timeout values, a writer thread is most likely stuck. To determine what the writer thread is doing,

try to obtain several thread dumps while the readers are blocking attempts to acquire the lock (perform Ctrl-Break on Windows or `kill -3` on UNIX).

Note that with 10.1.3.1.0, the algorithm to select the next receiver of the lock was not fair, so the possibility of lock starvation even under normal conditions was still present. With 10.1.3.3.0, the selection algorithm uses a queue to select the next receiver of the lock.

Scheduling and Activity Expiration

This section provides answers to frequently asked questions about scheduling and activity expiration.

This section contains the following topics:

- How Does Oracle BPEL Server Use Quartz?
- How Do I Tune the Performance of Quartz?
- Why Does My Wait Activity and onAlarm Branch of a Pick Activity Not Expire?

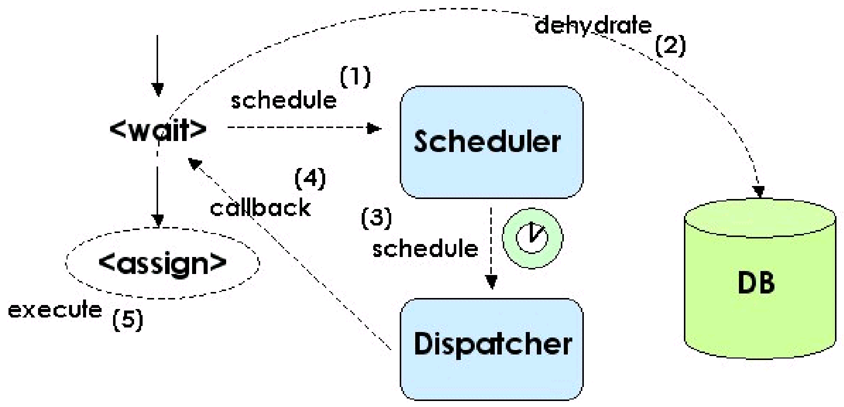
How Does Oracle BPEL Server Use Quartz?

Oracle BPEL Process Manager uses Quartz to schedule expiration events for wait and onAlarm activities. In release 10.1.*x*, all scheduled jobs are stored using an in-memory table. The benefit is that the scheduling action is very quick; it involves only an insertion into a HashMap. The disadvantages are:

- You must reschedule work items for expiration if the node that contained the in-memory schedule table goes down. The jobs are not persisted to a file or a database table. Therefore, the schedule table must be rebuilt when the server restarts.
- There is no automatic failover of scheduled jobs to another node in a clustered situation. If the scheduled jobs were stored in a shared resource, the other nodes have knowledge of jobs that are scheduled and should be executed, regardless of which node initially scheduled them.
- To manually recover the timer after a node failure in an Oracle BPEL Process Manager cluster, you must go to Oracle BPEL Control on another node, click the **Processes** tab, and click the **refresh the alarm table** link in the lower left portion of the page.

  Figure 1–16 provides an overview of scheduling a wait activity.

Figure 1–16 Overview of Scheduling a Wait Activity



The steps involved in scheduling a wait activity or the onAlarm branch of a pick activity are as follows:

1. Schedule the job with the Quartz scheduler. In this case, the job contains the work item key for the wait activity that is completed when the scheduler job calls back. The wait activity is now in state `open.pending.complete`.
2. Dehydrate the instance state (if there are no other activities to perform, as there are if you are in a flow activity).
3. When the scheduler job starts after the specified time period, a message is scheduled with the dispatcher to complete the wait activity.
4. The dispatcher passes the message to a WorkerBean MDB thread that performs the callback on the wait activity. After this step, the wait activity's state is `closed.completed`. The next activity following the wait can now be executed.
5. The assign activity following the wait is executed.

How Do I Tune the Performance of Quartz?

In 10.1.*x* releases of Oracle BPEL Process Manager, the number of Quartz threads can be tuned by changing the `com.oracle.bpel..threadCount` properties in the *SOA_ORACLE_HOME*/`bpel/domain/`*DOMAIN_NAME*/`config/resources-quartz.properties` file.

```
# BPEL agent thread pool configuration =====================================
#
# org.quartz.threadPool.threadCount will be used as the default value for
# each agent's thread pool.  More fine-grained tuning on a per-agent basis
# can be achieved by setting the threadCount values below.
#
# com.oracle.bpel.expirationAgent - schedules wait/onAlarm activities;
# this is the most heavily used agent in the system.  If your flows
# typically have many wait/onAlarm activities and the total number of
# instances created is high, performance may suffer if the expiration agent
# thread count is not sufficient to handle the load.  In worst-case not
# having enough threads to handle high incoming request counts may result
# in OutOfMemory errors as the number of scheduled tasks piles up.
#
com.oracle.bpel.expirationAgent.threadCount = 10

# com.oracle.bpel.schedulerAgent - schedules activities that will be
# performed in the future.  Currently, only mail receive/onMessage activities
# utilize this scheduler.
#
com.oracle.bpel.schedulerAgent.threadCount = 1

# com.oracle.bpel.dispatchAgent - periodically "pings" the dispatcher to
# introduce an additional thread to help process messages that may be
# trapped in the queue.
#
com.oracle.bpel.dispatchAgent.threadCount = 1

# com.oracle.bpel.activationAgent - schedules activation agents for
# processes.  You may wish to adjust the number of threads for this agent
# if your processes have implemented their own activation agent listeners
# (as opposed to use adapters).

#
com.oracle.bpel.activationAgent.threadCount = 1

# com.oracle.bpel.testAgent - periodically cleans up activities that have
# not been performed (or have failed to perform) due to a failure in the
# JMS layer.
#
com.oracle.bpel.testAgent.threadCount = 1
```

Oracle BPEL Server heavily uses Quartz for expiration of wait and onAlarm activities. In addition, Quartz is used for the following actions:

- Sending periodic `ping` commands to the dispatcher to flush out stranded messages
- Enabling polling for custom activation agents
- Scheduling execution of activities that must be performed in the future (for example, mail receive activities)

Each of these use cases can be tuned by increasing or decreasing the thread counts for the individual thread pools. Because the expiration agent is the primary consumer of Quartz scheduled jobs, the default size of the `expirationAgent` thread pool is much higher than the others. As with all thread and connection pool tuning, you must strike a balance between increasing throughput and system responsiveness and the increased overhead caused by larger thread counts.

Why Does My Wait Activity and onAlarm Branch of a Pick Activity Not Expire?

As long as the wait and onAlarm branch of a pick activity have been dehydrated without incident, the reasons why the activity can miss its expiration callback are as follows:

- The Oracle BPEL Process Manager node on which the activity expiration was scheduled goes down suddenly (as described in "How Does Oracle BPEL Server Use Quartz?" on page 1-24).
- The expiration callback is received before the instance and activity have finished dehydrating (race condition). This is normally hidden by an automatic retry mechanism in the handle expiration callback method. The `domain.xml` property `expirationMaxRetry` governs the maximum number of retries before giving up.

As described in section "How Does Oracle BPEL Server Use Quartz?" on page 1-24, the solution for both of these cases is to manually reschedule the missing expiration agents.

Performance

This section provides answers to frequently asked questions about performance.

Is Much Overhead Incurred when Calling a BPEL Subprocess?

Overhead is substantial on BPEL subprocess calls in the following areas:

- XML serialization of outbound messages if the subprocess is being invoked over SOAP
- Duplication (deep copy) of the outbound XML message if the subprocess is co-located and being invoked through the local shortcut
- Persistence of the XML message into the `invoke_message` table if the subprocess is called asynchronously

SOAP serialization of outbound messages cannot be optimized and is the price for accessing services over the network.

Calling a BPEL subprocess that is co-located in the same JVM can be optimized so that the deep copy operation is eliminated. For XML messages larger than 100k, the effect of duplicating the document object model (DOM) is quite substantial. For messages around 1-2k, the effects are typically negligible. To disable the deep copy operation, add the `skipDeepCopy` partner link property for the invoke activity.

```
<BPELSuitcase>
<BPELProcess id="BankTransferFlow" src="BankTransferFlow.bpel">
<partnerLinkBindings>
<partnerLinkBinding name="client">
<property name="skipDeepCopy">true</property>
</partnerLinkBinding>
</partnerLinkBindings>
...
</BPELProcess>
</BPELSuitcase>
```

This optimization may be applied to both asynchronous and synchronous invocations. This setting exposes a side effect when used with a synchronous invoke: as a reference to the XML document is passed, any changes made to the document in the subprocess are visible to the parent process. Normally, the deep copy operation shields you from this behavior because subprocess modifications are made to a local copy, not a shared reference.

Asynchronous invocations are handled differently than synchronous invocations. Other than the obvious blocking nature of the latter call, asynchronous subprocess invocations first store their message payloads in the `invoke_message` table, then schedule the message to be delivered through a background thread. The persistence of the invocation message allows for the following:

- The instance to be created even if the receiving node goes down
- The instance creation task to be delegated to another Oracle BPEL Process Manager node in the cluster

If the asynchronous invocation overhead is deemed unnecessary, it is possible to disable it. This effectively turns the asynchronous invoke into a synchronous one.

```
<BPELSuitcase>
<BPELProcess id="BankTransferFlow" src="BankTransferFlow.bpel">
...
<configurations>
<property name="deliveryPersistPolicy">off.immediate</property>
</configurations>
</BPELProcess>
</BPELSuitcase>
```

By specifying `deliveryPersistPolicy=off.immediate`, a `post()` call is delivered immediately to Oracle BPEL Server for execution and is not returned until the process hits a breakpoint activity (receive, onMessage, wait, or onAlarm). However, do not use this setting as the default invocation behavior for one-way invocations. This property is typically used when a synchronous process is calling an asynchronous process. Except in this case, if the asynchronous process involves many steps, the overall time to execute the calling synchronous process is extended, which may impact either the transaction timeout for the thread or overall throughput if few HTTP servlet threads are available (assuming that HTTP servlet threads are being used to execute synchronous BPEL processes).

Flow Logic

This section provides answers to frequently asked questions about flow logic.

- Which Error Conditions Cannot be Handled with a Fault Handler?
- Why Do I Receive Empty and Zero Node Selection Failure Faults?

Which Error Conditions Cannot be Handled with a Fault Handler?

A fault handler is used to catch Oracle BPEL Process Manager execution exceptions. With the catch and catchAll branches, Oracle BPEL Process Manager can catch and handle almost all business and runtime exceptions. However, there are conditions that the fault handler cannot handle:

- Some Oracle BPEL Server internal errors. When these occur, Oracle BPEL Server cannot ensure the execution of the Oracle BPEL Process Manager instance. For example, a database access exception occurs. If this occurs, Oracle BPEL Server sends this exception to the top. This prevents the fault handler from catching and handling it.
- Lower level JVM error (for example, `OutOfMemoryException`)
- If the transaction is set to roll back only or the transaction is rolled back at commit time because of a timeout. In this case, even if the fault handler catches the fault, any steps it takes are discarded.

Note: Oracle BPEL Process Manager internal errors are modeled using the class `CubeException`. The faults handled by the fault handler are modeled by the class `BPELFault`.

Why Do I Receive Empty and Zero Node Selection Failure Faults?

Many users experience zero node selection failure faults. This means the variable does not have valid XML data for the XPath query. When you deploy the process, Oracle BPEL Server tries to automatically initialize the variable using static analysis and XML schemas. However, this automatic initialization is not 100% accurate.

According to Appendix A of the *Business Process Execution Language for Web Services Specification*, if a variable is not initialized but an attempt is made to access it, Oracle BPEL Server throws a `bpel:uninitializedVariable` fault. This fault is thrown when there is an attempt to access the value of an uninitialized part in a message variable.

For ease of use in Oracle BPEL Process Manager's implementation, Oracle BPEL Server tries to initialize the variable using the `<to-spec>` queries for that variable. This usually works in 80% of the cases. For some complex cases, such as a `<choice>` model group or array usage in an XPath query, it does not initialize the variable.

Assuming you have the following assign activities for the variable, the process works as follows:

```
<process xmlns:ns1="http://sample/namespace">
...
<variable name="v1" type="messageType1"/>
...
<assign>
<copy>
<from expression="some-expression-returns-c node"/>
<to variable="v1" part="payload" query="/ns1:a/ns1:b/ns1:c"/>
</copy>
</assign>
...
<assign>
<copy>
<from expression="some-expression-returns-f node"/>
<to variable="v1" part="payload" query="/ns1:a/ns1:b/ns1:c/ns1:d/ns1:e/ns1:f"/>
</copy>
</assign>
...
</process>
```

When you deploy the above process, Oracle BPEL Server tries to initialize the variable `v1` with empty values (skeleton) by using the `<to-spec>` queries `"/ns1:a/ns1:b/ns1:c"` and `"/ns1:a/ns1:b/ns1:c/ns1:d/ns1:e/ns1:f"`, and using the XML schema. Because of automatic initialization, the XML looks as follows:

```
<a xmlns="http://sample/namespace">
<b> <c> <d> <e>
<f/>
</e> </d> </c> </b>
</a>
```

If the XPath query or XML schema is complicated, then it does not initialize the variable completely.

Usually the automatically initialized XML skeleton is not valid against the XML schema. You must make it valid using an assign activity or transformation. Also during development, you can use `bpelx:assign` or validate the variable against the XML schema. It is a good practice to validate your XML against the XML schema.

Fixing a Zero Node/Empty Node Selection Failure Fault

The *Business Process Execution Language for Web Services Specification* has an assign construct called a literal assign. You use the literal assign to initialize the variable before using the variable in the `<to-spec>` query. The syntax is as follows:

```
<assign>
```

```
<copy>
<from>
<a xmlns="http://sample/namespace">
<b> <c> <d> <e>
<f/>
</e> </d> </c> </b>
</a>
<to variable="v1" part="payload"/>
</copy>
</assign>
```

Methodology

This section provides answers to frequently asked questions about methodology.

This section contains the following topics:

- Should I Implement a Daemon using Oracle BPEL Process Manager?
- How Do I Handle Large XML Documents within BPEL (10.1.3.3)?
- How Do I Model a Two Stage Real-Time Response Process and Continue Using Oracle BPEL Server Scalability?
- When Do I Use Embedded Java within BPEL?

Should I Implement a Daemon using Oracle BPEL Process Manager?

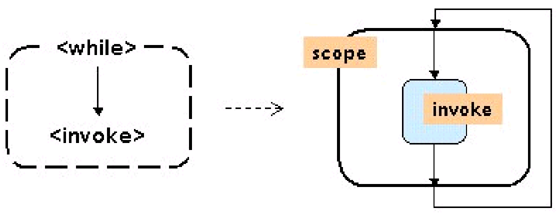
The short answer is no. This is because implementing a daemon process invariably involves a while loop in the flow. Inside the while loop, there is either a scope or a sequence compound activity. Each time the scope is entered, an object scope is created by the instance to store internal properties and variables declared within the scope. Essentially, each loop of the while loop generates a new scope in the scope tree for the instance. This causes the following elements to grow:

- Objects persisted in the database
- In-memory size of the instance's scope tree

Even if a scope or sequence is not included in the while loop, Oracle BPEL Server does so anyway to help distinguish work items created from one loop to the next.

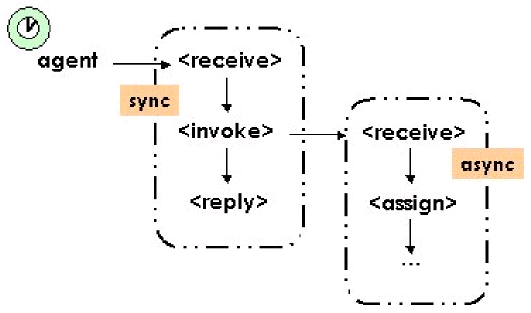
Figure 1–17 provides an overview.

Figure 1–17 While Loop Generating a Scope Activity



There is nothing in the BPEL language that prevents this pattern from being implemented. However, based on Oracle BPEL Process Manager’s implementation, if you have a long-running process that constantly persists state over time, this scenario does not fit the behavior of asynchronous durable flows. UNIX daemons are typically dedicated processes that persist little and keep a low-memory footprint over time (as they must run for days, weeks, or longer). The daemon pattern is best implemented by having the BPEL process service a single loop and have the constant looping and polling mechanism implemented using an activation agent or adapter. The adapter layer has dedicated thread pools and polling strategies that allow for more sophisticated behavior such as throttling and fan-out. Implementing a custom activation agent is another solution that allows the user to perform some validation before instantiating a BPEL process. In both cases, the suggested BPEL process type is synchronous and transient in order to minimize the database activity the polling process incurs and to minimize the number of completed polling instances created. If tracking is required, the synchronous polling process can spawn an asynchronous child that performs the actual work. Figure 1–18 provides an overview.

Figure 1–18 Synchronous Polling Process Spawning an Asynchronous Child



The following simple activation agent demonstrates how Oracle BPEL Server can initiate a BPEL process instance on a fixed interval (for example, every `3` seconds). The `heartBeatInterval` property (the value is in seconds) is an activation agent property specified in the `bpel.xml` file.

```
<activationAgents>
        <!-- value of heartBeatInterval is in seconds. the bpel engine will create
              a new instance of this process for every 3 seconds, class name is
             the activiation agent's class, in our case its SimpleActivationAgent.
              In this case, i want the engine to kick off a new instance every 3
              seconds, so i specified value "3" for the heartBeatInterval
              attribute -->
       <activationAgent
 className="com.oracle.bpel.activation.SimpleActivationAgent"
 heartBeatInterval="3">
          <!-- Specify the input data for this process, so every time when it
 creates an instance it will use this static xml as input -->
          <property name="inputData"><![CDATA[<name
 xmlns="http://samples.otn.com/helloworld">boo</name>]]></property>
          <property name="partName">payload</property>
       </activationAgent>
   </activationAgents>
```

The sample activation agent Java source is as follows:

```
package com.oracle.bpel.activation;
import java.util.Map;

import com.collaxa.cube.activation.ActivationLogger;
import com.collaxa.cube.activation.BaseActivationAgent;
import com.collaxa.cube.engine.ICubeContext;
import com.oracle.bpel.client.IBPELProcessConstants;
import com.oracle.bpel.client.Locator;
import com.oracle.bpel.client.NormalizedMessage;
import com.oracle.bpel.client.auth.DomainAuth;
import com.oracle.bpel.client.delivery.IDeliveryService;

/**
 * Activation agent to initiate/invoke a BPEL process based on the
 heartBeatInterval.
 * This activation agent can be bound to a process.It needs
 * the following properties to be specified in the bpel deployment descriptor.
 *
 *
 *
 * <activationAgents>
 * <activationAgent
 * className="com.collaxa.cube.activation.SimpleActivationAgent"
 * heartBeatInterval="60">
 * <property name="inputData" <![CDATA[<name
 xmlns="http://samples.otn.com/helloworld">boo</name>]]>
 * </property>
 * </activationAgent>
 * </activationAgents>
 */
public class SimpleActivationAgent extends BaseActivationAgent implements
    IHeartBeatAware
{
    public SimpleActivationAgent( )
    {
    }

    public void load(Map props, ICubeContext ctx)
        throws Exception
    {
        super.load(props, ctx);
    }

    public String getName( )
    {
        return "SimpleActivationAgent";
    }

    /**
     * This method gets called whenever a process state changes from ON to OFF
     * or OFF to ON. In case of OFF, it will close the scheduler listeners so that
     * no new instances will be created
     *
     * @param state
     *            the state of the process
     * @param ctx
     *            cube context
     */
    public void onStateChanged( int state, ICubeContext ctx ) throws Exception
    {
        ActivationLogger.debug( "SimpleActivationAgent", "onStateChanged",
            "State is changed for process '",
            getBPELProcessId().getProcessId(), "', state=", String
                .valueOf( state ) );
        if( state == IBPELProcessConstants.STATE_OFF )
        {
            // when the process goes to OFF state, we will close the
            // scheduler
            //
            closeSchedulerAgent( ctx );
        }
        else if( state == IBPELProcessConstants.STATE_ON )
        {
            // goes to ON state..we will restart the scheduler agent for
            // this agent
            //
            startSchedulerAgent( ctx );
        }
    }
    /**
     * This method gets called whenever a process lifecycle changes from ACTIVE
     * to RETIRED or RETIRED to ACTIVE. In case of RETIRED, it will close the
     * scheduler so that no new instances will be created
     *
     * @param mode
     *            the life cycle mode of the process
     * @param ctx
     *            cube context
     */
    public void onLifeCycleChanged( int lifecycle, ICubeContext ctx )
        throws Exception
    {
        ActivationLogger.debug( "SimpleActivationAgent", "onLifeCycleChanged",
            "Lifecycle is changed for process '", getBPELProcessId()
                .getProcessId(), "', mode=", String.valueOf( lifecycle ) );
        if( lifecycle == IBPELProcessConstants.LIFECYCLE_RETIRED )
        {
            closeSchedulerAgent( ctx );
        }
        else if( lifecycle == IBPELProcessConstants.LIFECYCLE_ACTIVE )
        {
            startSchedulerAgent( ctx );
        }
    }

    /**
     * This method gets called whenever a process get undeployed. In this case
     * it will close the scheduler.
     *
     * @param ctx
     *            cube context
     */
    public void onUndeployed( ICubeContext ctx ) throws Exception
    {
        ActivationLogger.debug( "SimpleActivationAgent", "onUndeployed",
            "Undeploying process '", getBPELProcessId().getProcessId(),
            "' shutdown the scheduler for now." );

        closeSchedulerAgent( ctx );
    }

    /**
     * This method gets called by the engine periodically (heartBeatInterval
     * period).
     */
    public void onHeartBeat( ICubeContext ctx ) throws Exception
    {
        DomainAuth auth = getDomainAuth( getBPELProcessId().getDomainId() );
        // get the input message if any
        //
        String inputMessage  = (String)
 getActivationProperties().get("inputData");
        String partName = (String) getActivationProperties().get("partName");
        NormalizedMessage nm = new NormalizedMessage();
        if( inputMessage == null )
        {
            inputMessage = "<empty/>"; // send some empty message if inputData is
 not specified

        }

        if (partName != null)
        {
            nm.addPart( partName, inputMessage);
        }
        else
            nm.addPart( "payload", inputMessage);

        String[] operationNames = super.getInitiateOperationNames();

        // use the delivery service to post the message to the engine

        //
        Locator locator = new Locator(auth );
        IDeliveryService deliveryService =
           (IDeliveryService) locator.lookupService(IDeliveryService.SERVICE_
NAME);

        // if it is a oneway operation, use the post method, if it is two way
        // operation use the request method.
        //
        deliveryService.post( getBPELProcessId().getProcessId(),
                              getBPELProcessId().getRevisionTag(),
                              operationNames[ 0 ],
                              nm );

    }
}
```

Place the compiled activation agent class under `SOA_ORACLE_HOME/bpel/system/classes/com/oracle/bpel/activation`.

How Do I Handle Large XML Documents within BPEL (10.1.3.3)?

Oracle BPEL Process Manager version 10.1.3.3 provides two new features for transferring large payloads with the file/FTP adapter:

- Attachment support — handles nondebatchable files larger then 10 MB
- XML debatching — debatches large XML files and Oracle ESB native files.

Visit MetaLink for details about downloading the 10.1.3.3 patch set.

`https://metalink.oracle.com/`

See *Oracle SOA Suite New Features* for information about these two new features:

`http://www.oracle.com/technology/products/ias/bpel/pdf/10133technotes.pdf`

How Do I Model a Two Stage Real-Time Response Process and Continue Using Oracle BPEL Server Scalability?

A common requirement when designing BPEL processes is that upon successful start, the process should send an acknowledgement back to the service consumer, and then continue processing. An asynchronous process is sent to Oracle BPEL Server for processing at a given point. Immediate execution is sometimes needed without causing Oracle BPEL Server to lock under the load (because the server is running out of threads).

Since the client expects a blocking reply, this is a use case for a synchronous process. On the other side, after consideration of the above topics, synchronous processes must be as short-lived as possible.

The solution for this problem lies in the partial processing pattern, as shown below:

```
<receive name="receiveInput"/>
[..]
<assign/>
[..]
<reply name="replyAcknowledgement">
<!-- from here the process is executed durable, and by another thread -->
<assign/>
[..]
<invoke name="callBackWithResult"/>
```

Using this pattern, you can design a highly scalable and performance-oriented system that can reply back to the client, and then continue long-lived activities (for example, wait, human workflow, and so on)

The following use case led to the above pattern: a process calls a (legacy) service synchronously. In 20% of the cases, this service is returning a fault. For business reasons, there must be a sophisticated retry with human workflow and other measures. Because the process must be as real time as possible in the best case, the first invocation of the service happens in a `<receive>` and `<reply>` block. A response is returned in any case (either a fault or the response from the service). If a fault occurs, all the logic handling it (for example, the custom retry) is performed after the reply.

Once the invocation is successful, a final response is sent to the client through the invoke. A portion of the process is shown below:

```
<sequence name="main">
        <!-- Receive input from requestor. -->
        <receive name="receiveInput" partnerLink="client"
                 portType="client:PartialProcessing" operation="initiate"
                 variable="inputVariable" createInstance="yes"/>
        <!-- invoke the external service -->
        <scope name="InvokeExternalSvc">
            <faultHandlers>
                <!-- catch the fault from the external service call -->
                <catch faultName="ns1:InvokeFault">
                    <reply name="Reply_Fault" partnerLink="client"
                           portType="client:PartialProcessing"
                           operation="initiate"
                           faultName="client:ProcessingFault"/>
                </catch>
            </faultHandlers>
            <sequence name="sequence">
                <!-- invoke the faulty svc -->
                <invoke name="Invoke_Svc"/>
                <!-- reply the response -->
                <reply name="Reply_Success" partnerLink="client"
                       portType="client:PartialProcessing"
                       operation="initiate"/>
            </sequence>
        </scope>
        <!-- send the response async'ly -->
        <invoke name="callbackClient" partnerLink="client"
                portType="client:PartialProcessingCallback" operation="onResult"
                inputVariable="outputVariable"/>
    </sequence>
```

When Do I Use Embedded Java within BPEL?

Embedded Java within BPEL is similar to the embedded assembly language in C. That is, you are embedding a lower level language in a higher level language.

Embedded Java allows you to call a lower API that has not yet been properly exposed to a BPEL native construct and activity. When a behavior is properly represented by a BPEL construct and activity, use BPEL directly.

Java Embedding Example 1

Assuming you do not have an adapter for the Simple Network Management Protocol (SNMP), a part of the process wants to get status through SNMP, and there is an SNMP-related Java library already available. In this case, you can use embedded Java to call that SNMP Java library and put the result back into a BPEL variable.

Java Embedding Example 2

The typing system in XPath 1.0 is limited. It does not natively support `xsd:decimal` or `xsd:dateTime`. You end up using a string to present those data types and providing an emulation XPath function based on string data. Given that string-based computations may not be most efficient or some `dateTime` computation has not been represented in BPEL's native activity and function, you can use Java to directly perform `dateTime` and decimal-based computation.

Since Oracle BPEL Server is running within an EJB and MDB environment, new BPEL developers should not generally do things that are dangerous or inappropriate within this environment (for example, trying to interfere with the transaction directly, creating a new thread, or trying to create a network endpoint listener).

Deployment Descriptor Properties

This section lists the deployment descriptor `configurations` and `partnerlinkbinding` properties.

Table 1–1 describes the property names of the `configurations` deployment descriptor.

Table 1–1 Configuration Properties for the configurations Deployment Descriptor

Property Name	Description	Current State
`completionPersistLevel`	Sets the portion of the instance information that you want to save after the instance is completed. Possible values are: ■ `all` (default) — the instance is saved in both the `cube_instance` and `cube_scope` tables. ■ `instanceHeader` — only the metadata of the instances are saved in the `cube_instance` table. Note that this property can only be set if the `inMemoryOptimization` property is set to `true`. **See Also:** *Oracle Application Server Performance Guide* for additional details about the `inMemoryOptimization` and `completionPersistLevel` properties	Active
`completionPersistPolicy`	Configures how the instance data is saved. Possible values are: ■ `on` (default) — the completed instance is saved normally ■ `deferred` — the completed instance is saved, but with a different thread and in another transaction ■ `faulted` — only the faulted instances are saved ■ `off` — no instances of this process are saved **See Also:** *Oracle Application Server Performance Guide* for additional details about the `completionPersistPolicy` property	Active
`defaultInput`	The XML document that you want to use as input to test the process from Oracle BPEL Control.	Active
`deliveryPersistPolicy`	The setting of persist policy of this process in the delivery layer. This setting overrides the same value in `domain.xml`. Possible values are: ■ `on` — message sent into the system is saved in the delivery store before being picked up by Oracle BPEL Server ■ `off` — message sent into the system is saved in memory before being picked up by Oracle BPEL Server ■ `off.immediate` — the instance-initiating message is not temporarily saved in the delivery layer. Oracle BPEL Server uses the save thread for initialization.	Active
`initializeVariables`	Indicates whether Oracle BPEL Server sets the default value to variables. This eliminates the need for you to assign values to variables before doing XPath manipulation. Possible values are: ■ `true` (default) — Oracle BPEL Server sets the default value to a variable based on `to-spec` queries ■ `false` — the compiler does not initialize the variables based on `to-spec` queries	Active

Table 1–1 (Cont.) Configuration Properties for the configurations Deployment Descriptor

Property Name	Description	Current State
`inMemoryOptimization`	Possible values are: ■ `false` — (default) ■ `true` — Oracle BPEL Server tries to do in-memory optimization on the instances of this process on `to-spec` queries. This property can only be set to `true` if it does not have dehydration points. Activities like wait, receive, onMessage, and onAlarm create dehydration points in the process. **See Also:** *Oracle Application Server Performance Guide* for additional details about the `inMemoryOptimization` property	Active
`keepGlobalVariables`	When the instance completes, this setting indicates if Oracle BPEL Server keeps the global variable values in the instance store. Possible values are: ■ `false` (default) — global variable values are deleted when the instance completes ■ `true` — global variable values are kept in the instance store	Active
`loadSchema`	Possible values are: ■ `true` (default) ■ `false` — XML schemas are not loaded and Oracle BPEL Process Manager becomes typeless	Active
`noAlterWSDL`	Possible values are: ■ `false` (default) ■ `true` — the compiler does not try to modify the process WSDL to add binding and service information	Active
`optimizeVariableCopy`	Possible values are: ■ `true` (default) ■ `false` — Oracle BPEL Server does not enable the copy-on-write feature for an assign copy	Active
`relaxTypeChecking`	Possible values are: ■ `false` (default) ■ `true` — the compiler does not check type compatibility with an assign activity	Active
`relaxXPathQName`	Possible values are: ■ `false` (default) ■ `true` — the compiler does not complain about unqualified steps in the query. For example, where the correct form must be: `query="/ns1:payload/ns1:name"`, the following form passes compilation, if this flag is turned on: `query="/payload/name"`.	Active
`sensorActionLocation`	Location of the sensor action XML file used by Oracle BPEL Process Manager. The sensor action XML file configures the action rule for the events.	Active
`sensorLocation`	Location of the sensor XML file. The sensor XML file defines the list of sensors into which Oracle BPEL Server logs events.	Active
`testIntroduction`	Introduction text that appears in the test console.	Active

Table 1–1 (Cont.) Configuration Properties for the configurations Deployment Descriptor

Property Name	Description	Current State
`transaction`	When set to `participate`, the process produces a fault that is not handled by fault handlers, which calls the transaction to be rolled back.	Active
`SLACompletionTime`	Service Level Agreement (Completion Time) — Threshold for a commitment within which a process is completed for a specified time period. The value is an XML duration.	Active
`xpathValidation`	Possible values are: ■ `true` (default) ■ `false` — the compiler does not validate the XPath queries	Active
`user`	The username a calling user must provide (given that domain level security is on).	--
`pw`	The password a calling user must provide (given that domain level security is on).	--
`role`	The role a calling user must belong to in the identity management (given that domain level security is on).	--

Table 1–2 describes the configuration properties of sections of the `partnerLinkBinding` deployment descriptor.

Table 1–2 Configuration Properties for the partnerLinkBinding Deployment Descriptor

Property Name	Description	State
`basicHeaders`	Creates HTTP basic authentication. The following values are supported: ■ `propagate` — If the process has been invoked securely, these credentials are also used for the outbound direction ■ `credentials` — Passes credentials from the descriptor	Active
`basicUsername`	The username (passed to basic authentication)	Active
`basicPassword`	The password credential (passed to basic authentication)	Active
`callbackBindings`	List of bindings that the compiler generates for the callback `portType`. The default value is `soap`. You set multiple bindings separated by commas (for example, `jms, soap`). The first item is used as the preferred binding when calling back.	Active
`correlation`	Possible values are: ■ `wsAddressing` (default) ■ `correlationSet` — this partner link is using the BPEL correlation set	Deprecated
`contentType`	Sets the special HTTP `contentType`. Example: `text/xml`.	Deprecated since 10.1.3
`fullWSAddressing`	Possible values are: ■ `false` (default) ■ `true` — WSDL is generated to include full WSA headers in binding: `From`, `Action`, `To`, and `FaultTo`	Active

Table 1–2 (Cont.) Configuration Properties for the partnerLinkBinding Deployment Descriptor

Property Name	Description	State
`httpAccept`	Overwrites the HTTP `accept` header that Oracle BPEL Server sends to the remote SOAP service.	Axis only
`httpContentType`	Overwrites the HTTP `content-type` header that Oracle BPEL Server sends to the remote SOAP service.	Axis only
`httpKeepAlive`	If the server permits `keepAlive` connections, this Boolean property can be turned on to take advantage of it. Therefore, connections to the same server are shared between invocations. This property was previously named `keepAlive`.	Axis only
`httpPassword`	For WSIF HTTP username and password authentication	Active
`httpUsername`	For WSIF HTTP username and password authentication	Active
`location`	URL that overrides the location defined in the WSDL. For SOAP over HTTP binding, this value overrides the SOAP address.	Active
`nonBlockingInvoke`	Possible values are: ■ `false` (default) ■ `true` — Oracle BPEL Server spawns a separate thread to perform the invocation to prevent the invoke activity from blocking the instance. **See Also:** *Oracle Application Server Performance Guide* for additional details about the `nonBlockingInvoke` property	Active
`preferredPort`	The preferred port to use in case multiple WSDL ports are available. The value is the `NCName` of the WSDL port.	Active
`retryInterval`	Number of seconds that Oracle BPEL Server waits between retries.	Deprecated by fault policy in 10.1.3.3
`retryMaxCount`	Number of retries that Oracle BPEL Server attempts, if an invoke fails because of network problems.	Deprecated by fault policy in 10.1.3.3
`sendXSIType`	Some legacy RPC-style Web services require the `xsi:type` to be set with every element in the input message. If this value is set to `true`, Oracle BPEL Process Manager populates the `xsi:type` of all the elements.	Active
`serviceProperties`	XML element that is passed to the WSIF provider.	Active
`timeout`	Number of seconds in which a SOAP call times out. A remote fault is thrown if this happens.	Axis only

Table 1–2 (Cont.) Configuration Properties for the partnerLinkBinding Deployment Descriptor

Property Name	Description	State

`validateXML`	Enables message boundary validation. When set to `true`, Oracle BPEL Server validates the XML message against the XML schema during a receive activity and an invoke activity for this partner link. If the XML message is invalid, then a `bpelx:invalidVariables` runtime fault is thrown. This overrides the domain level `validateXML` property. The following example enables validation for only the `StarLoanService` partner:
	`<partnerLinkBinding name="StarLoanService">`
	`<property name="wsdlLocation">`
	`http://<hostname>:9700/orabpel/default/StarLoan/Sta`
	`rLoan?wsdl</property>`
	`<property name="validateXML">true</property>`
	`</partnerLinkBinding>`
	See Also: *Oracle Application Server Performance Guide* for additional details about the `validateXML` property
		Active

`wsdlLocation`	URL of the WSDL file that defines this partner link. This property must be present. The BPEL compiler needs this to validate the BPEL source. This can be an abstract WSDL in that only the `portTypes` and their dependencies need to be defined in the WSDL.	Active
`wsdlRuntimeLocation`	URL to the partner link WSDL. It is used on Oracle BPEL Server, which means that the concrete WSDL with all the service, port, and binding definitions is needed. This property is optional and defaults to the `wsdlLocation` property. This property also enables multiple URLs separated by blanks (spaces, new lines, and tabs). Therefore, Oracle BPEL Server tries sequentially if any URLs are not available.	Active
`wsseHeaders`	Creates a WS-Security username token. Possible values are: ■ `propagate` — If the process has been invoked securely, these credentials are also used for the outbound direction ■ `credentials` — Passes credentials from the descriptor **See Also:** *Oracle BPEL Process Manager Administrator's Guide* for additional details about the `wsseHeaders` property	Active
`wsseUsername`	The username for the token (required) **See Also:** *Oracle BPEL Process Manager Administrator's Guide* for additional details about the `wsseUsername` property	Active
`wssePassword`	The password for the token (optional) **See Also:** *Oracle BPEL Process Manager Administrator's Guide* for additional details about the `wssePassword` property	Active

See Also: *Oracle BPEL Process Manager Developer's Guide* for additional details about setting these properties during design-time in Oracle JDeveloper and during runtime in Oracle BPEL Control

2 Oracle Enterprise Service Bus

This chapter provides answers to frequently asked questions about Oracle Enterprise Service Bus (ESB).

This chapter contains the following topics:

- Third-Party Compatibility and Certifications
- Core Functionality
- Performance
- Project Lifecycle
- Troubleshooting
- Installation
- Transaction Semantics In Oracle ESB
- Use Case for Using Pure SQL with Dynamic Routing in Oracle ESB
- Setting Up the Use Case for Using Pure SQL with Dynamic Routing in Oracle ESB

Third-Party Compatibility and Certifications

This section provides answers to frequently asked questions about third-party compatibility and certifications.

This section contains the following topics:

- Can Oracle ESB Run in a Third-Party J2EE Container (JBoss, WebSphere, WebLogic)?
- Can I Use a Third-Party JMS Provider (TIBCO, WebSphereMQ, SonicMQ) as the Internal Oracle ESB Transport?
- Can Oracle ESB Send to or Receive from a Third-party JMS Provider (TIBCO EMS, WebSphere JMS)?
- What Databases are Supported as a Storage Backend for the ESB Server?
- On Which Operating Systems Does Oracle ESB Run?
- How Can I Use an Oracle 10.1.2 Database Server on a Windows Client Workstation with Oracle ESB and Oracle SOA Suite?
- Can I Consume Oracle ESB Services from .NET Clients?
- Which Adapters are Available with Oracle ESB?

- How Do I Connect to a Protocol or Application for which Oracle Does Not Have an Adapter?
- Can You Create an ESB-to-ESB Bridge Across a Firewall?
- How Do I Call an Oracle ESB Routing Service from ADF?
- Can Oracle ESB Handle Inbound HTTP Posts?
- Does Oracle ESB Support WSIF and are Samples and Additional Information Available?
- Does Oracle ESB Support RPC Style SOAP Services?

Can Oracle ESB Run in a Third-Party J2EE Container (JBoss, WebSphere, WebLogic)?

Oracle ESB is certified to run on WebSphere (as is Oracle BPEL Process Manager). Certification on WebLogic and JBoss is currently in progress. Contact `dave.berry@oracle.com` for additional details.

Can I Use a Third-Party JMS Provider (TIBCO, WebSphereMQ, SonicMQ) as the Internal Oracle ESB Transport?

In 10.1.3.1, Oracle ESB requires OEMS for its *internal* usage. However, Oracle ESB has been designed to be provider-agnostic and Oracle may certify other vendors' JMS providers based on demand (for customers who already have a JMS provider in-house and do not want to learn and manage an extra JMS provider).

Oracle already supports many other vendors' JMS providers on the edges (that is, Oracle ESB can listen to and send messages on many different JMS providers).

Can Oracle ESB Send to or Receive from a Third-party JMS Provider (TIBCO EMS, WebSphere JMS)?

Yes. Oracle ESB, or more precisely the JMS adapter, supports a large number of third-party JMS providers. The following file shows how to send and receive against TIBCO EMS:

```
j2ee/SOA_CONTAINER/application-deployments/default/JMSAdapter/oc4j-ra.xml
```

What Databases are Supported as a Storage Backend for the ESB Server?

At this point, only Oracle databases are supported as a storage backend for the ESB Server. Oracle Lite is installed as part of the basic developer installation on Windows. Use Oracle Database 10*g* for production hosts.

On Which Operating Systems Does Oracle ESB Run?

Oracle ESB is part of Oracle Application Server, and runs on the same operating systems as that. See the certification matrix for the most up-to-date information:

`http://www.oracle.com/technology/software/products/ias/files/oracle_soa_certification_101310.html`

How Can I Use an Oracle 10.1.2 Database Server on a Windows Client Workstation with Oracle ESB and Oracle SOA Suite?

Normally, the Oracle Database server can only be installed on a Windows Server, such as 2000 or 2003 Server.

For development purposes, Oracle SOA Suite bundles Oracle Lite as part of the basic installation to minimize the installation footprint of using a full Oracle database. You can also use the advanced installation option and configure it to run against Oracle XE, which works fine on Windows XP.

Can I Consume Oracle ESB Services from .NET Clients?

It is completely possible. Be sure to install and develop the .NET clients with VisualStudio SDK 3.0 and use VisualStudio 2005 Service Pack 1. There are problems if the .NET clients are *not* developed with these components.

Which Adapters are Available with Oracle ESB?

All adapters supported by Oracle BPEL Process Manager are available in Oracle ESB:

- Technology adapters (file, FTP, MQSeries, database, AQ, and JMS)
- OracleAS Adapter for Oracle Applications
- Third-party adapters:
  - Applications (J.D. Edwards OneWorld, SAP, PeopleSoft, and Siebel)
  - Legacy (CICS, IMS/DB, IMS/TM, Tuxedo, and VSAM)

How Do I Connect to a Protocol or Application for which Oracle Does Not Have an Adapter?

You can always write Java code and integrate either using WSIF Java binding or by building a custom JCA adapter.

Oracle provides technology, application, and mainframe adapters. Others may be available through our partners in a co-sell model (that is, sold on partner paper and supported by the partner). Current Oracle SOA Suite adapter partners are iWay, Pervasive (specialize in small-to-medium size business applications), Attunity (specialize in mainframe adapters), GT Software (specialize in adapters for mainframe platforms z/OS, MVS, and VSE), NetManage, and Ericom (specialize in screen scraper adapters). If you still do not find an adapter for your application, you likely need to understand the architecture for the target system and application and look at using our available technology adapters.

Can You Create an ESB-to-ESB Bridge Across a Firewall?

Oracle ESB can connect through a firewall on the edges using traditional MOM technologies such as the FTP adapter and SOAP/HTTP Web services with proxy. The 10.1.3 OC4J JMS Router component can also send JMS messages across a firewall over HTTP(S). Oracle ESB does not internally support any node-to-node communications through a firewall for asynchronous routing rules or any internal Oracle ESB JMS topics.

How Do I Call an Oracle ESB Routing Service from ADF?

You create a Web service proxy, as shown in section 10.3 of the *Oracle SOA Suite Tutorial*. This question came up on the SOA Oracle Technology Network forum:

```
http://forums.oracle.com/forums/thread.jspa?threadID=451847
```

Can Oracle ESB Handle Inbound HTTP Posts?

Yes, by writing a J2EE servlet that receives the HTTP POST or GET and invokes an Oracle ESB service. The HTTP WSIF provider can generate outbound HTTP requests. Additionally, Oracle is considering building an HTTP adapter if customer demand is sufficient. Send customer requests to `vikas.anand@oracle.com`.

Does Oracle ESB Support WSIF and are Samples and Additional Information Available?

Yes, Oracle ESB supports WSIF. There are samples available on the Oracle ESB Oracle Technology Network page (`http://www.oracle.com/technology/goto/esb`). There is an article on WSIF entitled "Using WSIF for Integration" that is recommended for anyone looking to understand and make use of the benefits of WSIF. Additionally, Oracle ESB relies more heavily on WSIF for Java integration because Oracle ESB does not include the `EXEC` Java functionality that is included in Oracle BPEL Process Manager.

Does Oracle ESB Support RPC Style SOAP Services?

No, Oracle ESB 10.1.3 does not support RPC style SOAP services, binary attachments, or multipart WSDLs.

Core Functionality

This section provides answers to frequently asked questions about core functionality.

What Type of Transaction Support Does Oracle ESB Have?

The following information is from the ESB Advanced Presentation available at the following location:

```
http://www.oracle.com/technology/products/integration/esb/pdf/esb-advanced-archite
cture-presentation.pdf
```

Oracle ESB provides full support for participating in and initiating JTA/XA transactions across any service interactions that use Java, and which are from the following destinations:

- JCA adapters such as database, JMS, AQ, and MQ
- Java programs invoked using the Oracle ESB invocation API
- BPEL processes using the internal Java service binding

Oracle ESB inherits inbound global transactions:

- An asynchronous routing rule ends the scope of the current transaction
- New Oracle ESB-initiated transactions are grouped by the ESB system

Transaction exception handling and rollback:

- Errors on existing inbound transactions are rolled back to the initiator
- Errors on Oracle ESB-initiated transactions can be resubmitted from Oracle ESB Control
- The end-to-end message flow terminates on the first failed service regardless of transaction state or owner

More Transaction Use Case Questions and Answers

Question

Is a global transaction supported across Oracle BPEL Process Manager and Oracle ESB when they are in the same OC4J?

Answer

Yes, but you must force Java between Oracle BPEL Process Manager and Oracle ESB, not SOAP HTTP. See the transaction lesson at the following location: `http://www.oracle.com/technology/goto/esb`.

Question

Oracle BPEL Process Manager (containing logic) calls a local ESB (through a synchronous routing service in Oracle ESB) on a global transaction. When the local ESB goes down, it fails. Does the transaction get rolled back?

Answer

Yes.

Question

When Oracle BPEL Process Manager calls a local ESB in a global transaction and succeeds, the global transaction ends here. The local ESB takes control and routes the update or insert message to a remote ESB through an asynchronous SOAP service. When the remote ESB goes down, it fails and the message persists in the hospital queue in the local ESB and waits for resubmission in the order in which the message was received. Is this possible?

Answer

Yes, this is possible if Oracle BPEL Process Manager calls Oracle ESB over SOAP or you use an asynchronous routing rule in the local Oracle ESB to terminate the inbound Oracle BPEL Process Manager transaction.

Question

BPEL calls a local ESB in a global transaction. When it succeeds, the global transaction ends there. The local ESB takes over and routes the insert or update message to the remote ESB through a synchronous SOAP service. When it also succeeds, the remote ESB receives the message as a routine service and internally routes the message to an outbound ESB adapter. When the remote ESB adapter fails (for example, the EBS database may be down), the message persists in the remote ESB and waits for resubmission. Is this possible?

Answer

Yes, same as the previous answer. It is recommended that you go through the transaction slides on `http://www.oracle.com/technology/goto/esb`. You have the flexibility to do this, but there are rules you must follow and understand to achieve the behavior you want.

Is a Database Installation Required for Oracle ESB Only If You Do Not Use JMS/AQ?

Yes, for the Oracle ESB service metadata.

Does Oracle ESB Support Multipart Messages or Binary Attachments?

The only workaround for this in 10.1.3 is to place an OC4J Web service, Oracle Web Services Manager, or Oracle BPEL Process Manager in front of Oracle ESB services.

Besides Being a Logical Grouping of ESB components, What Else Does an ESB System Provide?

The ESB system is often referred to as the unit of scalability. Any set of services defined within an ESB system share the same basic configuration data as defined by the following features:

- Runtime server cluster isolation for high availability and scalability
- Discrete internal asynchronous topic for service distribution and scalability
- Discrete internal error topic for increased exception handling granularity
- Discrete exception notification configuration
- External load balancing virtual host port and host names
- Life cycle management using import and export system identifier

See the ESB advanced architecture documentation for more information.

http://www.oracle.com/technology/products/integration/esb/pdf/esb-advanced-architecture-presentation.pdf

How Can I Send an E-mail Notification from Oracle ESB?

Out of the box, you can configure notification information in the Oracle ESB Control System Configuration page to send notifications by e-mail, phone, fax, and other channels when exceptions occur. For custom notifications, you must define an outbound WSIF or OC4J J2EE Web service with custom coding to send notifications. The following Oracle Wireless Messaging sample provides a code example that can be repurposed to build an OC4J Web service to call from Oracle ESB:

http://www.oracle.com/technology/obe/obe_as_10g/wireless/wirelessmessaging/mesg.htm

Additional information can be found on this Oracle Technology Network Forum thread:

http://forums.oracle.com/forums/thread.jspa?messageID=1531686

If a Routing Service Has Two Asynchronous Services, What Happens If Both Fail? Do One or Two Error Hospital Messages Get Generated?

Only the first service generates an error based on the transaction rule described above where the message flow terminates on the first failed service. See the ESB advanced architecture documentation for more information:

http://www.oracle.com/technology/products/integration/esb/pdf/esb-advanced-architecture-presentation.pdf

Does Oracle ESB Support a Management API like Oracle BPEL Process Manager Provides?

Oracle ESB provides some support. The standard interfaces for Oracle ESB are the user interfaces in Oracle Enterprise Manager 10*g* Grid Control Console and Oracle ESB Control. Oracle ESB also includes an invocation API that can invoke a service from Java.

The 10.1.3.3 patch release includes an ESB Client API that has been used at sites to track and resubmit error instances. A patch of this API for 10.1.3.1 is also available.

Is Service Communication in the Same Java Container Optimized to Avoid the Overhead of TCP Connections and HTTP Request Formatting?

Yes, if the services are in the same container, the ESB fabric uses Java to communicate for synchronous routing rules and internal JMS topics for asynchronous routing rules.

What Are the Best Practices for Oracle ESB User Accounts And Roles?

Oracle ESB uses Oracle Application Server authentication. All Oracle ESB users can delete instance data, but only users assigned the `ascontrol_admin` role can update Oracle ESB metadata. In a production environment, it is suggested that only Oracle ESB users with a specific requirement to update service metadata be given the `ascontrol_admin` role. The following roles are exposed in 10.1.3.1 Application Server Control Console with Oracle ESB support.

- Role `ascontrol_admin` enables the following:
  - Oracle ESB metadata read and write access
  - Oracle ESB instance read and write access
- Role `ascontrol_appadmin` enables the following:
  - Oracle ESB metadata read access
  - Oracle ESB instance data read and write access
- Role `ascontrol_monitor` enables the following:
  - Oracle ESB metadata read access
  - Oracle ESB instance data read and write access

Does Oracle ESB Support Oracle BAM?

Oracle ESB does not support Oracle Business Activity Monitoring (BAM) design-time sensors like Oracle BPEL Process Manager, but there is an Oracle ESB — Oracle BAM dashboard that is ready for use, out of the box. You can install the dashboard, and change the host name from the default of `localhost` in the demonstration.

A generic BAM WSDL with its operations can be used to insert directly through Web services, or you can have a JMS listener on the monitor topic and receive events from there.

How Do I Manage the Timeout Duration for a Synchronous Request-Response from a SOAP Invocation Service?

Service timeout durations can be managed from the Oracle ESB Control **Properties** tab by setting the service **RetryCount** and **RetryInterval** endpoint properties.

There are also runtime server global settings available in *SOA_ORACLE_HOME*\integration\esb\config\esb_config.ini for OutboundRetryCount, OutboundRetryInterval, and OutboundRetryEnabled.

What XML document Size Does Oracle ESB support?

The main constraints to document size are document parsing and transformations by Oracle XDK, and auditing and logging. Document sizes over 10 MB or more should be achievable by limiting the number of transformations and filter expressions and disabling instance tracking and Oracle Enterprise Manager logging.

Does Oracle ESB Support Message Ordering?

For a synchronous flow, order is guaranteed if the inbound source adapter is configured with one thread for the source endpoint and there is only one target endpoint. Ordering for asynchronous routing rules is guaranteed if the outbound endpoint is not clustered and multithreaded.

Does Oracle ESB Support the JCA Adapter rejectedMessageHandlers Property Like Oracle BPEL Process Manager?

Oracle ESB supports the file, queue, and WSIF rejection handler types. The syntax example below can be inserted in the adapter's esbsvc file after the invocation element. You must terminate Oracle JDeveloper and update this file manually with a text editor, then open the project in Oracle JDeveloper and register it. Check the file again to ensure that the update still exists after you register the project. After registering, you can view and edit the new rejectedMessageHandlers property in the adapter's Oracle ESB Control **Properties** tab. The following example shows how to change the location of the file adapter rejection handler.

```
...
   </invocation>
   <endpointProperties>
        <property name="rejectedMessageHandlers"
 value="file://c:\rejectSample\reject"/>
   </endpointProperties>
</service>
```

The following example shows how to set a rejection handler to send the message to AQ. JMS and BPEL message handlers are not supported.

```
<property name="rejectedMessageHandlers"
 value="queue://jdbc:oracle:thin:@localhost:1521:XE|soademo/soademo|AQ_RAW_IN_
ERROR_QUEUE"/>
```

How Can I Set Up Oracle ESB for High Availability?

See chapter 3 of the *Oracle Application Server Enterprise Deployment Guide*. Additionally, see the Oracle ESB Deployment presentation on the Oracle Technology Network (http://www.oracle.com/technology/goto/esb) detailing this feature.

Does Oracle ESB Support Filtering or Setting Message Headers?

Yes, you can perform header-based filtering and set standard and custom header properties for adapters (JMS, AQ, and so on) and SOAP services. There are samples showing how to do this on the Oracle ESB Oracle Technology Network pages at http://www.oracle.com/technology/goto/esb.

Can Oracle ESB Pass Security Credentials When Invoking an External Web Service?

Yes, you set SOAP security headers such as WS-Security in the transformation before the SOAP service callout. There is a SOAP headers sample showing how to do this on the Oracle ESB Oracle Technology Network page (`http://www.oracle.com/technology/goto/esb`). Additionally, you can use Oracle ESB Control to set HTTP basic authentication endpoint properties **HTTP_USERNAME** and **HTTP_PASSWORD** for an Oracle ESB SOAP service.

Can I Resubmit Multiple Error Instances at the Same Time?

Not in Oracle ESB Control, but you can do this using the Oracle ESB Management Client API, which is included in the 10.1.3.3 patch release or in a one-off 10.1.3.1 patch.

```
package oracle.tip.esb.client;

public interface ConsoleClient public static ConsoleClient public class
ConsoleClientFactory

API action=GetFailedInstances action=ResubmitInstancesByIds
action=ResubmitInstanceById
```

What Is the Oracle ESB Number of Listeners Feature?

This feature controls the number of listeners the Oracle ESB Server spawns on the underlying asynchronous topic used for all asynchronous routing rules under this system. If you have two ESB systems with five listeners each, then you have 10 listener threads altogether on this container. Each listener picks up only one message off the topic. There are no duplicates because ESB always guarantees exactly-once message delivery. However, multiple listeners does not guarantee FIFO message ordering.

Can I Register an Oracle ESB Project from the Command Line with Ant Scripts?

Yes, you can download the ESB deployment package, which contains the following custom `ant` tasks:

- The `deployESBProjects` task allows Oracle JDeveloper ESB projects to be registered to a given Oracle ESB Server.
- The `undeployESBEntities` task allows ESB systems, service groups, or services to be deleted (that is, unregistered) from the identified Oracle ESB Server.
- The `extractESBDeploymentPlan` task builds an XML deployment plan of customizable properties for all of the ESB systems, service groups, and services found within a single Oracle JDeveloper ESB project.
- The `deployESBSuitcase` task supports deploying the ESB project artifacts found in the identified Oracle JDeveloper ESB project directory using the extracted and edited deployment plan.

Can Automatic Retry Be Configured in the Following Scenario?

Question

I am trying to configure a publish-subscribe scenario. I have the subscribers transacting independently (asynchronous routing rules with adapters in different systems) and I can manually resubmit failures from the error hospital. These are the properties in `esb_config.ini`:

```
#Retry
InboundRetryCount = 3
InboundRetryInterval = 5
InboundRetryEnabled = true

OutboundRetryCount = 3
OutboundRetryInterval = 5
OutboundRetryEnabled = true
```

Answer

You can configure the adapter to retry like in Oracle BPEL Process Manager with adapter endpoint properties. Edit the Oracle ESB Control Properties page **RetryCount** and **RetryInterval** values. For example, if an endpoint is down for several minutes, setting **RetryCount** = **5** and **RetryInterval** = **60** causes it to retry for **5** minutes before it gives up and sends back to the error hospital for manual resubmitting. The `esb_config.ini` values are default values for that ESB runtime server.

Additionally, if you want to perform automatic or batch resubmittal of errors from error hospital, you can utilize the ESB Management Client API.

Are There Known Issues with Oracle ESB WSDLs?

Question

Why do I receive an internal error when trying to register a concrete WSDL from Oracle ESB in the Service Registry? I do not have a problem with registering a Java Web service or BPEL process.

Answer

Strictly speaking, Oracle ESB WSDLs are 100% compliant to the specification. However, clients sometimes have problems parsing Oracle ESB WSDLs from adapter services or handling multiple bindings or nested imports in Oracle ESB-generated concrete WSDL files. In general, it is best to only expose and consume Oracle ESB routing service WSDLs and to test Oracle ESB WSDLs with tools such as SOAPUI or the Oracle Enterprise Manager Test tool to determine if there is a problem with Oracle ESB or the SOAP client.

How Can I Disable the Tracking Messages for One or More Services?

You can delete tracking fields for each individual service using Oracle ESB Control. There is a known issue in which the **Disable** check box does not work, but deleting it is a simple work around.

What Is the Correct Way to Purge ESB Instance Data?

There are at least three options:

- Use Oracle ESB Control to purge using the time-based filter criteria.
- Use the ESB client to programatically select and choose based on the service metadata. The existing resubmit sample in the client package can be adjusted to purge instead of resubmit.
- Use the database package provided below when the ESB design-time server is down.

You can purge instance data from Oracle ESB Control or use the `DBMS_SCHEDULER` as shown below to automate purging of ESB instance data. Since Oracle ESB Servers cache instance data, it is recommended that you only run this when they are not running.

```
REM Change the value of repeat_interval in DBMS_SCHEDULER.CREATE_SCHEDULE
REM If the esb schema is different than ORAESB, Replace ORAESB string with the esb
REM  schema name

GRANT CREATE JOB TO ORAESB;
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;

BEGIN
   DBMS_SCHEDULER.DROP_JOB(job_name => '"ORAESB"."PURGE_ALL_INSTANCES"');
   DBMS_SCHEDULER.DROP_SCHEDULE(schedule_name => '"ORAESB"."PERIODIC_PURGE"');
   DBMS_SCHEDULER.DROP_PROGRAM(program_name=>'ORAESB.PURGE_INSTANCES');
END;
/

BEGIN
   DBMS_SCHEDULER.CREATE_PROGRAM(
      program_name=>'ORAESB.PURGE_INSTANCES',
      program_action=>'begin
         DELETE FROM ESB_TRACKING_FIELD_VALUE;
         DELETE FROM ESB_FAULTED_INSTANCE;
         DELETE FROM ESB_ACTIVITY;
         DELETE FROM ESB_TRANSACTION_STATUS;
         DELETE FROM ESB_RELATION_XML;
         DELETE FROM ESB_SERVICE_RELATION;
         DELETE FROM ESB_INSTANCE_RELATION_XML;
         COMMIT;
         end;',
      program_type=>'PLSQL_BLOCK',
      number_of_arguments=>0,
      comments=>'Purge All Instances',
      enabled=>TRUE);

   DBMS_SCHEDULER.CREATE_SCHEDULE(
      repeat_interval => 'FREQ=DAILY; BYHOUR=23',
      start_date => SYSTIMESTAMP,
      comments => 'Schedule for periodic purge of instance data',
      schedule_name => '"ORAESB"."PERIODIC_PURGE"');

   DBMS_SCHEDULER.CREATE_JOB(
      job_name => '"ORAESB"."PURGE_ALL_INSTANCES"',
      program_name => 'ORAESB.PURGE_INSTANCES',
      schedule_name => 'ORAESB.PERIODIC_PURGE',
      job_class => 'DEFAULT_JOB_CLASS',
      comments => 'Purge instances',
      auto_drop => FALSE,
      enabled => TRUE);

END;

/
```

How Can I Propagate the File Name from Input File Adapter to Output File Adapter Using JCA Headers?

Oracle ESB supports JCA header manipulation just like Oracle BPEL Process Manager. This specific file name example is shown on the Oracle ESB Oracle Technology Network page (`www.oracle.com/technology/goto/esb`) in the **Learning More** section under the **Overview: Oracle ESB Header Support** link. Additionally, there are samples provided there that show more complex SOAP and JMS use cases.

Does Oracle ESB Support an Inbound E-mail Adapter?

Not in 10.1.3. Oracle ESB does not currently support the activation agent framework required for this. A somewhat simple workaround is to write a J2EE servlet that uses JavaMail to poll a mail server and generate a JMS message that Oracle ESB receives using the JMS adapter.

Why are BPEL Processes Displayed in Oracle ESB Control?

This provides the Oracle ESB user with increased visibility of BPEL processes when they are in the same container. When a BPEL process is deployed, Oracle ESB listens on the internal JMS topic and builds a proxy service representing that BPEL process. This service can call the BPEL process directly over Java without using an Oracle ESB SOAP service over SOAP/HTTP. For more details, see the **Overview: ESB BPEL Java FastPath** link at `http://www.oracle.com/technology/goto/esb`.

Should Every BPEL Process Display in Oracle ESB Control?

Yes, every BPEL process that is in the same OC4J container should display as an Oracle ESB service.

Are There Known Issues with Synchronization of Oracle ESB Control?

Yes, currently the BPEL service can only be added during Oracle BPEL Process Manager deployment. If the service is deleted, you must undeploy and redeploy the BPEL process for it to appear again in the Oracle ESB Control service pane.

Can I Install the Oracle ESB and BPEL Components Without the Rest of Oracle SOA Suite?

Yes, there are links on the Oracle ESB Oracle Technology Network page (`http://www.oracle.com/technology/goto/esb`) to download the standalone components. Instructions are in *Oracle Application Server Enterprise Deployment Guide*.

Is the reset.bat/reset.sh Script a Supported Command and Where Is It Documented?

The `reset ant` script in `SOA_ORACLE_HOME/integration/esb/bin` was originally designed as a way to restart Oracle ESB for internal testing scripts. This script is useful for restoring an existing Oracle ESB environment to the original postinstallation state. Out of the box, it only works on Windows and Oracle Lite, but there are environment variables to use to set the database connection parameters to work against an Oracle database. The `reset` script clears the database schema and webDAV repositories, but runs the `export` script to attempt to back up all data in case of errors. You can use it on your own. Oracle tries to fix any serious bugs they may find, but essentially it is available for you to use in an as-is state.

Environment Variables Sample

```
set DB_URL=jdbc:polite4@127.0.0.1:1531:oraesb
set DB_USER=system set DB_PASSWORD=any
set SYS_PASSWORD=any
```

How Do You Handle Multiple BPEL Versions When Using an ESB SOAP Service to Call BPEL from Oracle ESB?

This assumes that an Oracle ESB SOAP service uses the Oracle BPEL Process Manager WSDL URL that does not include the version.

When a new version of the BPEL process is deployed, Oracle ESB continues to use the old version unless:

- Oracle ESB is restarted.
- The Oracle ESB SOAP service is disabled and enabled.
- A simple update is made to the Oracle ESB SOAP service.
- The endpoint location property for the SOAP service is changed.

Therefore, the best practice is as follows:

1. Use an Oracle ESB SOAP service that uses the Oracle BPEL Process Manager WSDL URL that does not include the version.
2. In Oracle ESB Control, set the following:
   a. Change the SOAP service location property to point to the new version.
   b. Edit and apply the SOAP Service description field, such as to the Oracle BPEL Process Manager WSDL version number.

Oracle BPEL Process Manager WSDL Examples

Oracle BPEL Process Manager exposed the WSDL URLs for each process: one with a version and one without:

```
http://localhost:8888/orabpel/default/SOAOrderBooking/SOAOrderBooking?wsdl
http://localhost:8888/orabpel/default/SOAOrderBooking/1.0/SOAOrderBooking?wsdl
```

The SOAP and HTTP endpoint location always includes the version in the URL:

```
http://localhost:8888/orabpel/default/SOAOrderBooking/1.0
```

How Do I Override a WSIF Invocation of an Oracle BPEL Process Manager Service from an Oracle ESB Service and Set the Endpoint Property Location?

You cannot currently do this (as of 10.1.3.3). If it is a WSIF, it assumes the same container and ignores the location.

Is There a Property for Pointing at Alternate Locations for a WSDL of a SOAP Service Endpoint?

There is not currently a property (as of 10.1.3.3), just the endpoint location itself. The best way to protect against the WSDL not being available is to import it into the project. It is then referenced from Oracle ESB design time through a slide, which is always there, even if the real WSDL is gone. This is what is done in the manage endpoints sample available at `http://www.oracle.com/technology/goto/esb`.

What Is the Separator for Different SOAP Endpoints When Configuring the Location Endpoint Property?

Space.

Does Oracle ESB Support Functionality Similar to Oracle BPEL Process Manager Dynamic Partner Links?

Yes, this functionality is included in the 10.1.3.3 release. For 10.1.3.1 users, this functionality is available with patch 5924483.

Can You Guarantee Once-and-Once-Only Processing in Oracle ESB Assuming the Failure of the Processing Node?

Reliability of message delivery always depends on the protocol used. If you use JMS (and for example, the database adapter, Java binding, and EJB calls as well), required once-and-once-only delivery is guaranteed. You must use reliable (persistent) messaging as the JMS provider (like Oracle Advanced Queuing) to avoid losing messages in case of, for example, a system crash.

If you use SOAP over HTTP, the situation is more complicated. There is a WS-Reliability standard to achieve guaranteed delivery, no duplicates, and guaranteed message ordering of SOAP messages. However, Oracle ESB does not support this (and you must also modify your existing services).

Can I Use Custom XPath Functions in Oracle ESB?

Yes, see the String2XML sample available at:

`http://www.oracle.com/technology/products/integration/esb/files/stringtoxml.zip`

However, this sample does not describe how to make the function appear in the Oracle JDeveloper XSLT Mapper.

To make the function appear in the Oracle JDeveloper XSLT Mapper, and to use it in the XSLT Mapper test facility, you must do two things:

1. Create an XML file detailing your extension functions and import the functions in the **User Defined Extension Functions Config File** field of the XSL Maps dialog of Oracle JDeveloper (select **Tools** > **Preferences** > **XSL Maps**). This adds the functions in the **User Defined Extension Functions** section of the **Component Palette** and enables you to drag and drop them into the XSLT Mapper during transformation design time.
2. Add the class to the Oracle JDeveloper classpath (*JDEVELOPER_HOME*/`jdev/extensions`) so that you can run the transformation using the test feature with Oracle JDeveloper.

The extension functions XML file looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?> <extension-functions>
<functions sample =
"http://www.oracle.com/XSL/Transform/java/extensionfunction.MyExtensionFunctions">

<!- converts a hex string to a decimal string ->
<function name="sample:parseString1" as="string">
<param name="inputXML" as="node-set"> </function>
```

```
</functions> </extension-functions>
```

To add the class to the Oracle JDeveloper classpath involves defining an Oracle JDeveloper extension. For Oracle JDeveloper 10.1.3.2, the extension must be a valid JSR 198 extension. Documentation on this can be found in: `JDEVELOPER_HOME/jdev/doc/extension/ide-extension-packaging.html`.

Do this by defining a simple `extension.xml` file as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<extension version="1.0" esdk-version="1.0" id="xsltfunctions"
xmlns="http://jcp.org/jsr/198/extension-manifest">
<name>XSLTExensionFunctions</name> <owner>Oracle</oracle> </extension>
```

The `extension.xml` (naming convention for these files) must be in the `meta-inf` folder of the jar file containing the classes. The jar file name must be a combination of the extension id and esdk-version. In this case, it is `xsltfunctions.1.0.jar`. Copy the jar file into the `/extension` directory.

Restart Oracle JDeveloper to pick up the changes for both the file specified in the **User Defined Extension Functions Config File** field and the Oracle JDeveloper extension. This function can then be referred to in any XSLT in any project in Oracle JDeveloper.

What Is Error Hospital and Do You Have to Set It Up?

Error hospital is a term that Oracle uses for guaranteed and reliable delivery of messages described in previous sections of this chapter. Certain parts of error hospital are provided out of the box and others may require custom work to retry a failed message. The Oracle ESB Oracle Technology Network page (`http://www.oracle.com/technology/goto/esb`) has a lesson on transactions and exception handling that covers this feature.

Does Oracle ESB Support Messages with No Namespace Prefix?

Oracle ESB, as with the rest of Oracle SOA Suite, generally requires fully-formed XML documents that conform to XML schema standards. If a message payload does not use namespaces, you can do one of the following tasks:

- Manually edit the XSL source file to remove all namespace data in the outbound document.

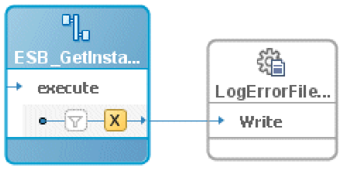
  Note: In 10.1.3, this breaks the Oracle JDeveloper XSLT Mapper user interface, but it works.

- Edit the data to remove namespace information often by using the Native Format Builder wizard as described in *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide*.

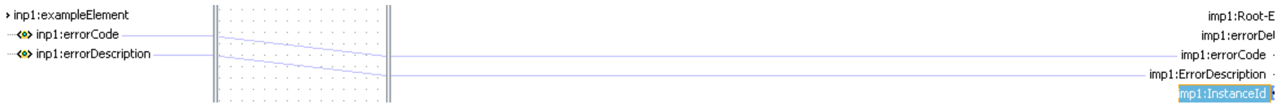
How Can I Get an Instance ID for Oracle ESB?

A new XPath function was added in 10.1.3.3 to get instance IDs. This section provides an example in which you invoke a BPEL process from an ESB process and pass the ESB instance ID to the BPEL process, along with other data.

1. Define a simple ESB project: **RS** > **Synch Routing Rule** > **File Adapter**.



2. Perform a transformation in the routing service:



   The only unmapped element in the target XML file is the **InstanceId** field element.

3. Perform the following steps to get the instance ID:

   a. Go to **Design View**.

   b. Select **Advanced Functions** in the **Component Palette**.

   c. Select **xpath-expression** and drag it to the middle column.

   d. Double-click the function icon.

   e. Enter **ehdr:getInstanceID()** in the **XPath Expression** field.

   f. Click **OK**.

4. Click **Yes** when prompted with a warning.

5. Map the function to the **InstanceID** element and save the project.

6. Return to the routing service where the XSL transformation activity is defined.

7. Double-click the routing service.

8. Expand the **Properties** section and add the following property:

   ```
   EnableAccessBusinessEvent=true
   ```

9. Save and deploy the project.

   The instance ID function should now work.

> **Note:** The XSL Design Time view will now not work. This is a limitation of the Oracle JDeveloper 10.1.3.3 product.

Can I Invoke a BPEL Process Locally (Using Local EJB References Instead of RMI) from Oracle ESB in 10.1.3.3?

Yes, you can improve Oracle ESB to Oracle BPEL Process Manager invocation performance by instructing Oracle ESB to call Oracle BPEL Process Manager using local EJB references instead of going over RMI. Perform the following configuration steps:

1. Enable the global JNDI for the OC4J container by adding `global-jndi-lookup-enabled="true"` to the `application-server` element in `server.xml`.
2. Add an endpoint property called `InvocationMode` to the Oracle ESB service that represents the BPEL process, and specify a value of `local`. Do this through the **Properties** tab on the ESB Service Definition page in Oracle ESB Control. Possible values for this property are `local` and `remote`. The default value is `remote`, which implies that by default Oracle ESB calls BPEL processes over RMI.
3. Restart the server (required for step 1 above).

Setting the global JNDI attribute to `true` flattens the JNDI tree. This means that any J2EE application can access any JNDI object in the container without providing credentials. This may increase security issues. Therefore, use this functionality carefully.

What Can I Do with Oracle ESB that I Cannot Do with Oracle BPEL Process Manager?

A short list of responses to this question are as follows:

- Optimized stateless runtime Oracle ESB Server provides higher performance messaging
- Simplified service virtualization including runtime configuration
- Domain value maps
- System cross referencing

Performance

This section provides answers to frequently asked questions about performance.

This section contains the following topics:

- How Fast Is Oracle ESB?
- Are There Any Recommendations to Follow to Maximize Oracle ESB Performance?
- Oracle ESB Sizing Questions

How Fast Is Oracle ESB?

This depends largely on the use case, but performance requirements can generally be met based on the following criteria:

- Given the stateless and minimal ESB Server, Oracle ESB is generally 2-4 times faster than Oracle BPEL Process Manager in similar use cases.
- For our internal exit criteria, well over 300 messages per second were achieved in a file adapter-to-file adapter use case.
  - See additional use cases in the *Oracle ESB Performance Guide*:

    `http://www.oracle.com/technology/products/integration/esb/files/oracleesb10-1-3performanceguide.pdf`
- For a recent JMS proof-of-concept, Oracle was able to improve performance from 10 MPS to over 160 MPS by following the OC4J JMS performance tuning recommendations described in "OC4J JMS" on page 2-18.

- For SOAP HTTP use cases, Oracle ESB can also process several hundred messages per second with linear scalability. The HTTP, SOAP, and Oracle ESB layers continue to be profiled in order to improve this.

Are There Any Recommendations to Follow to Maximize Oracle ESB Performance?

Install the performance updates available in patch 10.1.3.3 before you engage in a proof of concept.

General Performance Recommendations

Oracle ESB performance recommendations are generally the same as Oracle BPEL Process Manager, with suggestions listed below.

1. Turn off logging in Oracle Enterprise Manager.
2. Turn off instance tracking in Oracle ESB Control.
3. Since Oracle ESB is stateless, the database performance is not as important as for Oracle BPEL Process Manager if you turned off instance tracking.

- If you must leave instance tracking on, then migrate away from the default Oracle Lite database installed as part of Oracle SOA Suite.

1. Use synchronous services wherever possible to avoid costly file or database writes.
2. Minimize the use of transformations as much as possible.
3. Use domain-value mapping (DVM) instead of database lookup within XSLT transformations to minimize file input and output.

Web Services SOAP/HTTP

For Web service use cases, the out-of-box thread configuration is largely self tuning. You create a small 5-10% gain by using the OC4J built-in HTTP server instead of Oracle HTTP Server, but this is not a recommended production environment configuration.

While testing Oracle ESB services for performance through the SOAP/HTTP protocol, ensure that the SOAP client is configured for HTTP POSTs, and not GETs. GETs seem to keep Oracle ESB Server from overloading the CPU. Therefore, if you are seeing issues with CPU not topping off, verify the SOAP client configuration.

OC4J JMS

For file or in-memory JMS use cases using the OC4J JMS provider, there are several parameters to investigate for maximizing throughput and average response time. These parameters are described below.

JMS Adapter

JMS adapter activation `UseServerTimeout="true"`

- This is the most important change you can make to improve OC4J AS JMS performance.
- View the following code sample from the adapter `.wsdl` file. You must manually edit this outside of Oracle JDeveloper and reregister your service.

```
<jca:operation

ActivationSpec="oracle.tip.adapter.jms.inbound.JmsConsumeActivationSpec"
          DestinationName="jms/OSAOutQueue"
```

```
            UseMessageListener="false"
            UseServerTimeout="true"
            PayloadType="TextMessage"
            OpaqueSchema="false" >
        </jca:operation>
```

Incrementally increase the JMS adapter threads `adapter.jms.receive.threads` count.

- Do this incrementally because too many threads adversely impact performance due to thread contention.
- See the following code sample from the adapter `.esbsvc` file. You must manually edit this outside of Oracle JDeveloper and reregister your service.
- Once these actions are performed, you can edit inside the Oracle ESB Control adapter service **Properties** tab.

```
<invocation>
      <targetService guid="397496616AD211DBAF173B29CCB6C0EF"
 qname="PearsonA.OSAOutQueue_RS"/>
      <interface>
         <wsdlURL>OSAOutQueue.wsdl</wsdlURL>
         <portType
 xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/jms/OSAOutQueue/">tns:Consume_
Message_ptt</portType>
      </interface>
   </invocation>
   <endpointProperties>
      <property name="adapter.jms.receive.threads" value="2"/>
   </endpointProperties>
```

OC4J AS JMS Provider for File Persistence and In-Memory

1. Edit *SOA_ORACLE_HOME*`\j2ee\home\config\jms.xml` and restart the server with the following changes.
2. To reduce the average response time, set the `oc4j.jms.messagePoll` property to `100`.

   This is particularly important in light loads.
3. To add an additional 5-10%, you can turn off instrumentation by setting `oc4j.jms.noDms`" to `true`.

   ```
   <config-properties>
         <config-property name="oc4j.jms.noDms" value="true"/>
         <config-property name="oc4j.jms.messagePoll" value="100"/>
   </config-properties>
   ```

Oracle ESB Sizing Questions

There is some sizing information in the *ESB Performance Guide* available on the Oracle Technology Network:

http://www.oracle.com/technology/products/integration/esb/files/oracleesb10-1-3performanceguide.pdf

Use Case 1

I have a retail customer who is interested in implementing an Oracle ESB solution for real-time POS data. They are looking at creating two Oracle ESB flows:

- The first reads from a database or AQ queue, transforms the data from the original format to a common XML format, and writes it out to an AQ queue.
- The second reads from the AQ queue and writes it out to an Oracle database (there may also be a simple transformation).

The message size is small and they are looking at a volume of 10 million transactions a day through both Oracle ESB processes. This is the first Oracle ESB implementation, with more planned.

At this point, they must create a ballpark budget and are interested in what the Oracle ESB architecture may look like (number of servers, size, memory, and so on). In using the sizing documentation, the following estimation has been created:

- Throughput:

  ```
  10m txn/day ~ 416670/hr ~ 7000/min ~ 120/sec
  ```

- Based on the Oracle BPEL Process Manager and Oracle ESB performance documentation and because Oracle ESB is about twice as fast as Oracle BPEL Process Manager, a single CPU can be enough to handle the `120 txn/sec` projected volume for each Oracle ESB process. If you have two Oracle ESB processes, you should have a 2 CPU server with 4-8 GB of memory at a minimum.
- For redundancy, you should have two instances of each of the two processes: Inbound1, Inbound2, and Outbound1, Outbound2.
- Given all of this, Oracle ESB can handle this volume with two servers, each with 2 CPUs and 4-8 GB of memory.
- Source and target resources are not figured into these numbers. If the Oracle RDBMS and/or Oracle AQ are used, those resources are on different servers.

Questions

- With this limited information, are there any comments or suggestions on the suggested architecture?
- Is there a formula for estimating the required memory for the XML transformations?

Response

This is a very good Oracle ESB use case summary and sizing analysis. Your hardware proposal is very much in line with our Oracle ESB performance estimates. If they truly are small messages (< 1k) and the endpoints are on external systems with a fast pipeline, then you may be able to achieve much greater throughput (> 300 MPS). However, that margin makes it safe to assume that you can succeed with what you have proposed.

In terms of architecture, the high availability topology defined in the *Oracle Application Server Enterprise Deployment Guide* is sufficient. There is nothing specific on Oracle ESB XSL memory utilization. However, as you know, Oracle ESB uses the same Oracle XDK as Oracle BPEL Process Manager. Therefore, any best practices used there apply. One known issue is that Oracle XSD memory utilization is highly dependent on the document size and number of XSLs and filter expressions in the flow. If this is 10.1.3.1, then ensure that you apply the 10.1.3.3 patch.

Use Case 2

From your experience, can you say if a Sun Niagara server (UltraSparc T1 with 8 core) can manage 150,000 flows per day coming from an SAP system through the SAP adapter deployed on Oracle ESB 10.1.3.1? The 150,000 flows per day means an average of 6 flows a second.

How much memory is needed? Can you confirm that using the SAP adapter on Oracle ESB is more performant than using the adapter directly in the BPEL process?

Response

This is always a difficult question to answer because much depends on the actual implementation details (synchronous, asynchronous, message size, SAP systems throughput, and so on). In addition to average traffic volume (you averaged over 8 hours — a wise precaution), ask for the peak traffic because you must size your systems for bursts.

Oracle ESB is faster than Oracle BPEL Process Manager. In terms of the SAP adapter, there is no major difference in performance, whether you use it in Oracle BPEL Process Manager or SAP; it's the same adapter.

The first bottleneck you reach in applications integration is often on the ERP system and adapter side. These systems tend to be slower than the integration technology. This translates into growing queues (if using Oracle ESB asynchronous routing rules)/growing size in RAM (if synchronous RR) in Oracle ESB during peak times.

How do you size RAM? If your SAP endpoints can handle the bursts and Oracle ESB does not build a backlog, then go for minimum requirements such as (2 GB) x 2 = 4 GB. If SAP cannot handle that, and all runs are synchronous, you must perform testing to see the impact of backlog on RAM utilization.

Having said this, an 8-core machine for 6 MSG/S sounds quite appropriate. For more data points, see the other subsections of "Performance" on page 2-17 and ensure that you apply the required 10.1.3.3 patch.

Project Lifecycle

This section provides answers to frequently asked questions about project lifecycle.

This section contains the following topics:

- Can I Retire Oracle ESB Projects Like in Oracle BPEL Process Manager?
- Does Oracle ESB Support Versioning of Services Like Oracle BPEL Process Manager?

Can I Retire Oracle ESB Projects Like in Oracle BPEL Process Manager?

Yes, Oracle ESB offers a similar feature. In Oracle ESB Control, you can disable services. To retire a flow, you disable all starting endpoints.

Does Oracle ESB Support Versioning of Services Like Oracle BPEL Process Manager?

No, Oracle ESB does not support versioning in 10.1.3.1. The problem with versioning is not as acute in Oracle ESB as in Oracle BPEL Process Manager. This is because the typical lifetime of an Oracle ESB flow is many times shorter than a BPEL process. Therefore, the need to maintain multiple versions running simultaneously is lower.

Troubleshooting

This section provides answers to frequently asked questions about troubleshooting.

This section contains the following topics:

- Basic information to Provide When Reporting A Problem
- ORA-00018: maximum number of sessions exceeded (Oracle XE Running Out Of Processes)
- Helpful Troubleshooting Tools - Web Services
- How Can I Enable End-to-End Instance Tracking between Oracle ESB and Oracle BPEL Process Manager in the OrderBooking Demonstration?
- I Cannot Register Projects or Connect to Oracle ESB Control — How Do I Check that the Server is Running and On Which Port?
- I See Inconsistent Data in Oracle ESB Control and the DefaultSystem Group is Missing (How to Reset Your Slide/webDAV Cache)
- Why Does Oracle ESB Control Not Display Services in the Services Panel or Instance Tracking Information?
- I Recreated the oraesb Schema and Now the DefaultSystem is Missing. How Do I Create It?
- What Logging Facilities are Available for Oracle ESB in an Application Server Installation?
- How Do I Fix the Registration Error "system and service group that the project was using was already present in the server"?
- Why Am I Getting Error IOException ... access error ... Database is set to dirty, this may mean it is corrupt?

Basic information to Provide When Reporting A Problem

- Log files to send:
  - `SOA_ORACLE_HOME/opmn/logs/default*.log` and `opmn.log` (in the same location). Check to see if there are any Oracle ESB-related exceptions logged.
  - `SOA_ORACLE_HOME/j2ee/Instance_Name/log/*/oc4j/log.xml`. Check to see if the Oracle ESB application has started correctly.

  1. Clean up logs before starting Oracle SOA Suite as follows:

     `log-cleanup.bat` file contents:

     ```
     set AS_HOME=D:\ORACLE\OracleAS_1
     rm -R %AS_HOME%\opmn\logs\*
     rm -R %AS_HOME%\j2ee\home\log\*
     rm -R %AS_HOME%\j2ee\home\persistence\home_default_group_1\*
     pause
     ```

  2. Copy and zip the logs.

     `log-copy.bat` file contents:

     ```
     zip -r c:\tmp\logs.zip %AS_HOME%\opmn\logs\*
     zip -r c:\tmp\logs.zip %AS_HOME%\j2ee\home\log\*
     NAMEDATE /YZ:"Ymd-HMS" c:\tmp\logs.zip
     pause
     ```

- Send screenshots of failed user interface components (for example, Oracle JDeveloper, Oracle ESB Control, and so on).

- Send the Oracle JDeveloper project, an export of the problematic service metadata, or both.

You can automate the log capture process by using two simple scripts such as those mentioned above. Ensure that you adjust your `AS_HOME` and your OC4J instance name. (It may not be `home`.) These scripts assume availability of `MKS` or `CYGWIN`, and `namedate`. You can always write a DOS version of each.

ORA-00018: maximum number of sessions exceeded (Oracle XE Running Out Of Processes)

In its default configuration, Oracle XE barely has enough processes to handle Oracle SOA Suite. If you install the Orderbooking demonstration on top of it, you likely can run out of processes. Here is how to increase the number of processes:

```
sqlplus sys/system@XE as sysdba;
SQL> show parameter session;
SQL> show parameter processes;
SQL> alter system set sessions=100 scope=spfile;
SQL> alter system set processes=150 scope=spfile;
SQL> shutdown immediate
SQL> startup
SQL> show parameter session;
```

Helpful Troubleshooting Tools - Web Services

One easy way to test the error-prone JCA adapter configurations is to go against the WSDL of their routing service using one of the following SOAP testing tools.

- SOAP testing
  - Oracle Enterprise Manager — all Web services exposed by Oracle ESB are visible in Oracle Enterprise Manager. which also allows you to test them from your Web browser.
  - SOAPUI — a desktop application for inspecting, invoking, developing, and functional/load/compliance testing Web services over HTTP. Use this tool to invoke ESB services and check the responses. This tool is lightweight, requires no installer, and requires no documentation to get started. Visit `http://www.soapui.org/`.
  - Microsoft's Fiddler — allows you to inspect all HTTP traffic, set breakpoints, and debug with incoming or outgoing data. For example, you can use this tool to view SOAP headers. Visit `https://www.fiddlertool.com/fiddler/`.
- JMS testing
  - Hermes — a lightweight Swing application to test a variety of JMS providers, including Oracle. Visit `http://www.hermesjms.com/`.
- Load testing
  - Apache JMeter — a load-generation tool. Works for SOAP and JMS. Convenient for finding bottlenecks and configuring your demonstrations/proof of concepts computers. Visit `http://jakarta.apache.org/jmeter/`.

How Can I Enable End-to-End Instance Tracking between Oracle ESB and Oracle BPEL Process Manager in the OrderBooking Demonstration?

To enable end-to-end instance tracking in the OrderBooking demonstration:

1. Open `OrderBookingESB.esb` in Oracle JDeveloper and replace the SOAP service with a routing service directly invoking the Oracle BPEL Process Manager WSDL.
2. Ensure that all URLs in the chain are consistent (for example, use `localhost` and continue with it).
3. Unselect the **Can be invoked from an external service** check box under the **Definition** tab for **OrderFulfillment** (from Oracle ESB Control). You now have a single, end-to-end instance for each execution of the OrderBooking demonstration.

I Cannot Register Projects or Connect to Oracle ESB Control — How Do I Check that the Server is Running and On Which Port?

By default, the URL for Oracle ESB Control is:

```
http://127.0.0.1:8888/esb/esb/EsbConsole.html
```

There are multiple things you can check:

- Ensure the Container is Running
- Test that Oracle ESB Control is Functional
- Test Other Projects
- Identify the Port on Which the Oracle ESB Process is Listening

Ensure the Container is Running

1. Ensure that the container is running by executing the following `opmnctl` command:

```
SOA_ORACLE_HOME\opmn\bin\opmnctl status

Processes in Instance: yourhost.us.oracle.com
--------------------------------+--------------------+---------+---------
ias-component                   | process-type       |     pid | status
--------------------------------+--------------------+---------+---------
OC4JGroup:default_group         | OC4J:home          |   28568 | Alive
ASG                             | ASG                |     N/A | Down
HTTP_Server                     | HTTP_Server        |   28567 | Alive
```

Test that Oracle ESB Control is Functional

1. Go to Oracle ESB Control and see whether or not there is any content, such as the `DefaultSystem`.
2. If Oracle ESB Control is functional, test to see if you can create a new system by clicking the **Create** button. This tests to see if Oracle ESB Control is communicating with the backend database.

Test Other Projects

1. Ensure that your application server and integration server connections are working from Oracle JDeveloper.

2. Test a very basic project to determine if the problem exists with all projects or just this one.
3. Test the import function by importing the `CustomerData.zip` file as described in *Oracle Enterprise Service Bus Quick Start Guide*.
4. Ensure that there are no spaces in the path of your Oracle JDeveloper project.
5. Ensure that all external service dependencies such as a SOAP WSDL URL in the project are resolved.

Identify the Port on Which the Oracle ESB Process is Listening

The following instructions assume that you have `MKS` or `Cygwin` installed (`grep` and `ps`). If you do not, then the procedure is similar, but more manual and visual.

1. Open the Process Manager and try to find the `java.exe` file corresponding to Oracle ESB. Note the PID (`2884` for this example).

```
C:\> ps -l | grep -i java
   2884  0:32 java
```

2. Look for that PID in the list of processes with open ports on the host:

```
C:\tools&gt;netstat -ao | grep -i 2884
  TCP    dlher-us:8888      dlher-us.us.oracle.com:0  LISTENING       2884
  TCP    dlher-us:9129      dlher-us.us.oracle.com:0  LISTENING       2884
  TCP    dlher-us:23793     dlher-us.us.oracle.com:0  LISTENING       2884
  TCP    dlher-us:23945     dlher-us.us.oracle.com:0  LISTENING       2884
  TCP    dlher-us:3825      localhost,:3826        ESTABLISHED     2884
  TCP    dlher-us:3826      localhost,:3825        ESTABLISHED     2884
  TCP    dlher-us:3831      localhost,:1521        ESTABLISHED     2884
  TCP    dlher-us:3832      localhost,:1521        ESTABLISHED     2884
  UDP    dlher-us:3830      *:*
```

3. If you detect a port conflict, use the List of Open Files (`lsof`) on Unix or Linux to find which process is using a port:

```
lsof -i tcp:6003
```

I See Inconsistent Data in Oracle ESB Control and the DefaultSystem Group is Missing (How to Reset Your Slide/webDAV Cache)

If you do not have `DefaultSystem` showing in Oracle ESB Control, it usually indicates that the Oracle ESB runtime has an issue with the metadata server and underlying data stores. A few things to check are as follows:

1. Ensure Oracle Lite is running.

   Stop Oracle SOA Suite. Go to the Windows services panel and stop Oracle Lite. Restart it. Go to *`SOA_ORACLE_HOME`*`\integration\esb\olite\bin` and run `sql_olite.bat`. If you can connect, restart Oracle SOA Suite.

2. Reset the cache:

   Option 1:

   - In Windows Explorer, follow the wizard for **Add a network place** and create a network place for `http://localhost:8888/esb/slide/`. Access the wizard by selecting **My Network Places** and then deselecting the **Folders** button in the tool bar. See **Add a network place** in the left navigation bar.

When you open this location (your server must be running), you see several files that correspond to what you have already deployed to `DefaultSystem`. However, they may not correspond directly because of the problems you have encountered. Delete everything here, then shut down and restart the middle tier components. You should now see `DefaultSystem` when you open Oracle ESB Control in the browser. You can now attempt to register your Oracle ESB services.

Option 2:

- From the Developer Prompt, go to *`SOA_ORACLE_HOME`*`\integration\esb\bin` and run `reset.bat`. This clears your repository. Note that you must redeploy all Oracle BPEL Process Manager services for them to be visible to Oracle ESB Control under **DefaultSystem\BPEL**.

Why Does Oracle ESB Control Not Display Services in the Services Panel or Instance Tracking Information?

There is a fix available in 10.1.3.1. You can download and apply patch `5769394` from MetaLink. This fix is also incorporated into the 10.1.3.3 patch release.

1. Check the `opmn/logs/default*.log` and `opmn.log` (in the same location) files to see if there are any Oracle ESB-related exceptions logged.
2. Check the `j2ee/`*`Instance_Name`*`/log/*/oc4j/log.xml` to see if the Oracle ESB application has started correctly.
   - For the Oracle ESB design time installation, you see the following confirmation in the log file that the application was initialized correctly: `ESB bootstrap: Repository initialized`.
   - For the Oracle ESB runtime installation, you see the following confirmation in the log file that the application was initialized correctly: `ESB bootstrap: Runtime initialized`.
3. For an Oracle SOA Suite installation, you see both of the above logged separately.
4. Verify that the `ESB_PARAMETER` table in the `oraesb` schema has the following name-value pairs. Use SQL*Plus and enter these values into the table. Ensure the `DT_OC4J_HTTP_HOST` and `PORT` values reflect the values from your setup. Also, it is important to note that these values are applicable only for application server JMS.
   - `PROP_NAME_MONITOR_TOPIC_JNDI=OracleASjms/ESBMonitorTopic`
   - `PROP_NAME_ERROR_XATCF_JNDI=OracleASjms/MyXATCF`
   - `TrackingEnabled=Y`
   - `PROP_NAME_ERROR_RETRY_JNDI=OracleASjms/ESBErrorRetryTopic`
   - `DT_OC4J_HTTP_PORT=7777`
   - `PROP_NAME_CONTROL_TCF_JNDI=OracleASjms/MyXATCF`
   - `PROP_NAME_ERROR_TOPIC_JNDI=OracleASjms/ESBErrorTopic`
   - `PROP_NAME_MONITOR_TCF_JNDI=OracleASjms/MyTCF`
   - `PROP_NAME_ERROR_TCF_JNDI=OracleASjms/MyTCF`
   - `ACT_ID_RANGE=13650`

- `PROP_NAME_CONTROL_TOPIC_JNDI=OracleASjms/ESBControlTopic`
- `PROP_NAME_DEFERRED_TCF_JNDI=OracleASjms/MyTCF`
- `DT_OC4J_HOST=orasoadevapp101.ma.tmpw.net`
- `PROP_NAME_DEFERRED_XATCF_JNDI=OracleASjms/MyXATCF`
- `PROP_NAME_ERROR_RETRY_TCF_JNDI=OracleASjms/MyXATCF`

5. Review *Oracle Application Server Enterprise Deployment Guide* for instructions on how to set up a high availability environment where the JMS provider is Oracle AQ.
6. Check `j2ee/`*`Instance_Name`*`/log/*/oc4j/log.xml` for Oracle SOA Suite advanced installations to see if it contains `<MSG_TEXT>ORA-01000: maximum open cursors exceeded`. If this is the case, then increase the `open_curors` parameter to `1500` in the `init.ora` file and restart the database.
7. Restart your servers using `opmnctl shutdown` and `opmnctl startall`.

I Recreated the oraesb Schema and Now the DefaultSystem is Missing. How Do I Create It?

Use the *`SOA_ORACLE_HOME`*`/integration/esb/bin/import` script to import *`SOA_ORACLE_HOME`*`/integration/esb/install/config/seed/default-system.zip`.

What Logging Facilities are Available for Oracle ESB in an Application Server Installation?

Question

I tried modifying *`SOA_ORACLE_HOME`*`/integration/esb/config/esb_logger.properties`, and followed the Oracle Enterprise Manager instructions in *Oracle Enterprise Service Bus Developer's Guide*. While it appears that ESB is generating logs in *`SOA_ORACLE_HOME`*`/j2ee/`*`OC4J_HOME`*`/log/`*`OC4J_HOME`*`_default_group_1/oc4j`, I cannot correlate the entries with what I see in Oracle ESB Control. Am I missing something?

Answer

On the Oracle Technology Network Oracle ESB page (`http://www.oracle.com/technology/goto/esb`), there is a lesson showing how to configure the logger. You likely want to set `oracle.tip.esb.console` to `fine` to understand the logging information for each Oracle ESB Control request.

How Do I Fix the Registration Error "system and service group that the project was using was already present in the server"?

All Oracle ESB objects such as systems, service groups, or services have a unique global ID or GUID. You may see this error if a name collision occurs where two objects have the same GUID. The best practice for avoiding this is as follows:

- Do not redeploy with Oracle JDeveloper after deploying with `ant`.
- Always create Oracle ESB systems and service groups in Oracle JDeveloper and not in the server.

- If you receive this error, delete the services and systems from Oracle ESB Control and reregister the project.

Why Am I Getting Error IOException ... access error ... Database is set to dirty, this may mean it is corrupt?

One reason for this error is due to the ESB design time server being started with OPMN under `root` privileges. This causes all design time Oracle ESB artifacts to obtain `root` privileges, which makes it impossible for the Oracle home owner to read them. To resolve this issue, try giving back the required operating system privileges for the design time artifact files to the Oracle home owner, and restarting the server.

Installation

This section provides answers to frequently asked questions about installation.

How Do I Change Oracle ESB Schema Creation (IRCA Utility) to Use My Own Tablespace?

By default, the `irca` installation utility recreates the Oracle ESB schema in the `SYSTEM` tablespace, which is not always wanted. You can change `IRCA` script files to enable the schema to use your own custom tablespace.

1. Create the necessary tablespaces. You can also take care of the Oracle Web Services Manager schema at this point. In this sample, Oracle ESB and Oracle Web Services Manager are using their own tablespaces. Log in as database administrator and execute the following commands. Change the data file path (`/opt/oracle/oradata/asdb`) to reflect your own database installation structure:

```
CREATE SMALLFILE TABLESPACE "ORAESB" DATAFILE
 '/opt/oracle/oradata/asdb/oraesb01.dbf'

SIZE 200M AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED

LOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE SMALLFILE TABLESPACE "ORAWSM" DATAFILE
 '/opt/oracle/oradata/asdb/orawsm01.dbf'

SIZE 200M AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED
LOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

2. Navigate to the Oracle SOA Suite installation directory. In these steps, the root installation directory is called *`INSTALL_DIRECTORY`*. If the files are on a CD or DVD, copy the *`INSTALL_DIRECTORY`*`/install/soa_schemas/irca` subdirectory to your local hard drive for editing.
3. Change directories to *`INSTALL_DIR`*`ECTORY/install/soa_schemas/irca/sql/esb`.
4. Edit the `createschema.sql` file by changing line:

```
create user &oraesb_user identified by &oraesb_password;
```

to:

```
create user &oraesb_user identified by &oraesb_password default tablespace
oraesb;
```

5. Do the same thing for Oracle Web Services Manager by changing directories to *`INSTALL_DIRECTORY`*`/install/soa_schemas/irca/sql/owsm`.
6. Edit `createuser.sql` by changing line:

   ```
   create user &orawsm_user identified by &orawsm_password;
   ```

   to:

   ```
   create user &orawsm_user identified by &orawsm_password default tablespace
   orawsm;
   ```

Transaction Semantics In Oracle ESB

This section describes transaction semantics in Oracle ESB.

Same System — All Asynchronous

- If the first asynchronous routing rule fails, no other asynchronous routing rules execute.
- If any subsequent asynchronous routing rule fails, it tries to roll back all executed asynchronous rules. If the end service is nontransactional, the rollback does not happen. The flow as a whole does not roll back.

Same System — All Synchronous

- If the first synchronous routing rule fails, no other synchronous routing rules execute. The complete flow rolls back to the inbound service.
- If any subsequent synchronous routing rule fails, it tries to roll back all executed synchronous rules, and no other synchronous rules execute. If the end service is nontransactional, the rollback does not happen. The flow as a whole rolls back to the inbound service that started the flow.

Same System — Synchronous and Asynchronous

- If the first synchronous routing rule fails, no other routing rule executes, and the flow rolls back, including the inbound service.
- If any subsequent synchronous routing rule fails, it tries to roll back all executed synchronous rules, and no other synchronous and asynchronous rules execute. If the end service is nontransactional, the rollback does not happen. The flow as a whole rolls back to the inbound service that started the flow.
- If the first asynchronous routing rule fails, no other asynchronous routing rules execute, but none of the synchronous routing rolls back.
- If any subsequent asynchronous routing rule fails, it tries to roll back all executed asynchronous rules (it does not roll back synchronous routing rules). If the end service is nontransactional, the rollback does not happen. Also, the flow as a whole does not roll back.

Multisystem — All Asynchronous (Systems A and B)

- If the first asynchronous routing rule in system A fails, no other asynchronous routing rule in this system executes. The execution of routing rules in system B continues normally.

- If any subsequent asynchronous routing rule fails in system A, it tries to roll back all executed asynchronous rules in system A. If the end service is nontransactional, the rollback does not happen. The flow as a whole does not roll back. Execution of routing rules in system B continues normally.

Multisystem — All Synchronous

- If the first synchronous routing rule fails, no other synchronous routing rules (in any system) execute. The complete flow rolls back to the inbound service.
- If any subsequent synchronous routing rule fails, it tries to roll back all executed synchronous rules (both on system A and system B), and no other synchronous rules execute. If the end service is nontransactional, the rollback does not happen. The flow as a whole rolls back to the inbound service that started the flow.

Multisystem — Both Synchronous and Asynchronous

- If the first synchronous routing rule fails, no other routing rule executes (on any of the systems) and the flow rolls back, including the inbound service.
- If any subsequent synchronous routing rule fails, it tries to roll back all executed synchronous rules (on any of the systems), and no other synchronous and asynchronous rules execute across systems. If the end service is nontransactional, the rollback does not happen. The flow as a whole rolls back to the inbound service that started the flow.
- If the first asynchronous routing rule fails on system A, no other asynchronous routing rules execute on system A. None of the synchronous routing rolls back and the execution of routing rules in system B continues normally.
- If any subsequent asynchronous routing rule fails in system A, it tries to roll back all executed asynchronous rules in system A, (it does not roll back synchronous routing rules across systems). If the end service is nontransactional, the rollback does not happen. The flow as a whole does not roll back. The execution of routing rules in system B continues normally.

Notes:

- All synchronous services are always executed before asynchronous services.
- All SOAP endpoints are nontransactional.

Oracle BPEL Process Manager — Oracle ESB Pattern

This section describes Oracle BPEL Process Manager — Oracle ESB patterns.

Synchronous BPEL Process Calling Synchronous Oracle ESB Routing (Oracle ESB Native Binding)

If an Oracle ESB endpoint fails, the Oracle ESB flow rolls back all the way to the BPEL process and the BPEL process instance goes into the **Perform Manual Recovery** option in Oracle BPEL Control. If the conditions for the end point failure are removed, the recovery happens once manually resubmitted from the **Perform Manual Recovery** option. The same Oracle ESB instance that errored out now succeeds.

Synchronous BPEL Process Calling Synchronous Oracle ESB Routing (SOAP Binding)

If an Oracle ESB endpoint fails, the Oracle ESB flow rolls back all the way to the BPEL process and the BPEL process instance is faulted (in Oracle BPEL Control). The error returned is something like `Privileged Security Exception` and the real error for the Oracle ESB endpoint failure is available from an Oracle ESB Control trace.

Asynchronous BPEL Process Calling Asynchronous Oracle ESB Routing (Oracle ESB Native /SOAP binding)

If an Oracle ESB endpoint fails, the Oracle ESB instance is faulted and the instance can be submitted again for the failed endpoint. The BPEL process completes successfully.

Adapters and Transactional Behavior

Table 2–1 identifies which adapters are transactional and nontransactional.

Table 2–1 Adapters and Transactional Behavior

Adapter	Transactional	Nontransactional
Database	Yes	No
JMS	Yes	No
AQ	Yes	No
Applications	Yes	No
MQ	No	Yes
File	No	Yes
FTP	No	Yes

Transactional Behavior Configuration

- For XA:

  `oc4j-ra.xml` (sample entry) for the database adapter.

```
<connector-factory location="eis/DB/DhavalConnection" connector-name="Database
 Adapter">
<config-property name="xADataSourceName" value="jdbc/XADataSource"/>
<config-property name="dataSourceName" value=""/>
<config-property name="platformClassName"
 value="oracle.toplink.platform.database.Oracle9Platform"/>
<config-property name="usesNativeSequencing" value="true"/>
<config-property name="sequencePreallocationSize" value="50"/>
<config-property name="defaultNChar" value="false"/>
<config-property name="usesBatchWriting" value="false"/>
<connection-pooling use="none">
</connection-pooling>
<security-config use="none">
</security-config>
</connector-factory>
```

- Corresponding `data-sources.xml` entry:

```
<managed-data-source name="XADataSource" connection-pool-name="XAPool"
 jndi-name="jdbc/XADataSource"/>
<connection-pool name="XAPPool">
```

```
        <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
user="<username>"
password="<password>"
 url="jdbc:oracle:thin:@//<hostname>:<port>/<ServiceName>">
      </connection-factory>
  </connection-pool>
```

Nontransactional Behavior Configuration

- For non-XA:

  `oc4j-ra.xml` (sample entry) for the database adapter.

```
<connector-factory location="eis/DB/DhavalConnection" connector-name="Database
 Adapter">
<config-property name="xADataSourceName" value=" "/>
<config-property name="dataSourceName" value=" jdbc/NonXADataSource"/>
<config-property name="platformClassName"
 value="oracle.toplink.platform.database.Oracle9Platform"/>
<config-property name="usesNativeSequencing" value="true"/>
<config-property name="sequencePreallocationSize" value="50"/>
<config-property name="defaultNChar" value="false"/>
<config-property name="usesBatchWriting" value="false"/>
<connection-pooling use="none">
</connection-pooling>
<security-config use="none">
</security-config>
</connector-factory>
```

- Corresponding `data-sources.xml` entry:

```
<managed-data-source name="NonXADataSource" connection-pool-name="NonXAPool"
 jndi-name="jdbc/NonXADataSource"/>
<connection-pool name="NonXAPPool">

        <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
 user="<username>" password="<password>"
 url="jdbc:oracle:thin:@//<hostname>:<port>/<ServiceName>">
      </connection-factory>
  </connection-pool>
```

Use Case for Using Pure SQL with Dynamic Routing in Oracle ESB

In this use case, a client has more than one database instance for storing customer-related data. A third-party customer resource management (CRM) system is installed separately on different hosts in every geographical region.

Each CRM database has an identical schema and processes exactly the same queries in all of the applications and services. Instead of creating and maintaining six different database adapters (one for each data source) for every query, it would be useful to pass in the data source to use for a given call at runtime (for example, in a SOAP header). That way, they can create a single adapter for each query.

No data manipulation language (DML) statements are allowed for updating the CRM data using the Adapter Configuration Wizard services.

The goal of this use case is ease of maintenance in case of modifications in the query input and output parameters.

Setting Up the Use Case for Using Pure SQL with Dynamic Routing in Oracle ESB

This section describes how to set up, deploy, and test the use case.

This section contains the following topics:

- Setting Up the Database
- Setting Up the Data Sources
- Setting Up the Database Adapter
- Modeling the Use Case with ESB Designer
- Creating the Routing Service
- Deploying the Use Case
- Testing the Use Case

Setting Up the Database

1. Copy and paste the following SQL statements into a SQL*Plus session. For this use case, users named `kabel1` and `kabel2` and table `TEST1` are created. Use values appropriate to your environment.

```
connect sys/oracle10g@soa
/
create user kabel1 identified by oracle10g default tablespace users temporary
tablespace temp
/
grant connect, resource to kabel1
/
grant unlimited tablespace on users to kabel1
/
create user kabel2 identified by oracle10g default tablespace users temporary
tablespace temp
/
grant connect, resource to kabel2
/
grant unlimited tablespace on users to kabel2
/
connect kabel1/oracle10g@soa
/
CREATE TABLE "TEST1"
( "A" VARCHAR2(10) ,
"B" VARCHAR2(10)
)
TABLESPACE "USERS"
/
insert into test1 (a,b) values ('R1','Region1')
/
commit
/
connect kabel2/oracle10g@soa
/
CREATE TABLE "TEST1"
( "A" VARCHAR2(10) ,
"B" VARCHAR2(10)
)
TABLESPACE "USERS"
/
```

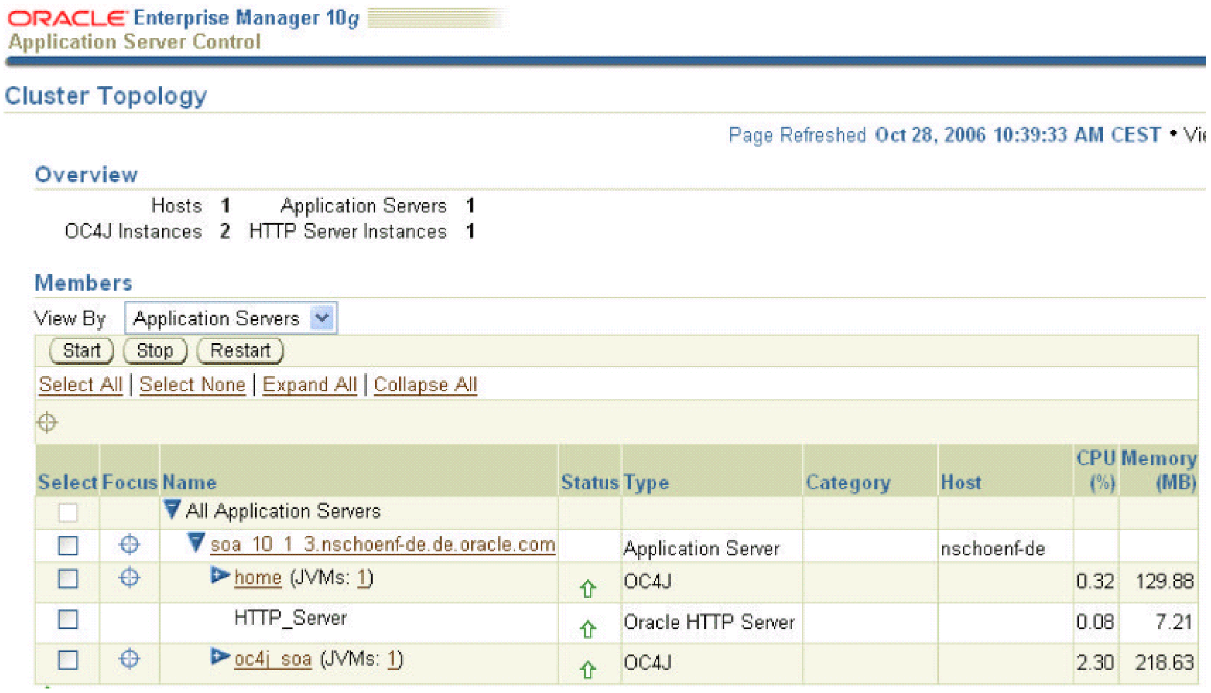
```
insert into test1 (a,b) values ('R2','Region2')
/
commit
/
```

Setting Up the Data Sources

1. Start Oracle SOA Suite.
2. Go to Oracle Enterprise Manager 10*g* Application Server Control Console.

   `http://`*`host_name`*`:`*`HTTP_port`*`/em/console`

   where *`host_name`* is the name of the host on which Oracle Enterprise Manager 10*g* Application Server Control Console is installed and *`HTTP_port`* is the port number to use.
3. Go to the OC4J container in which Oracle SOA Suite is installed.


ORACLE Enterprise Manager 10g
Application Server Control
Cluster Topology

Page Refreshed Oct 28, 2006 10:39:33 AM CEST • Vi

Overview

Hosts	1	Application Servers	1
OC4J Instances	2	HTTP Server Instances	1

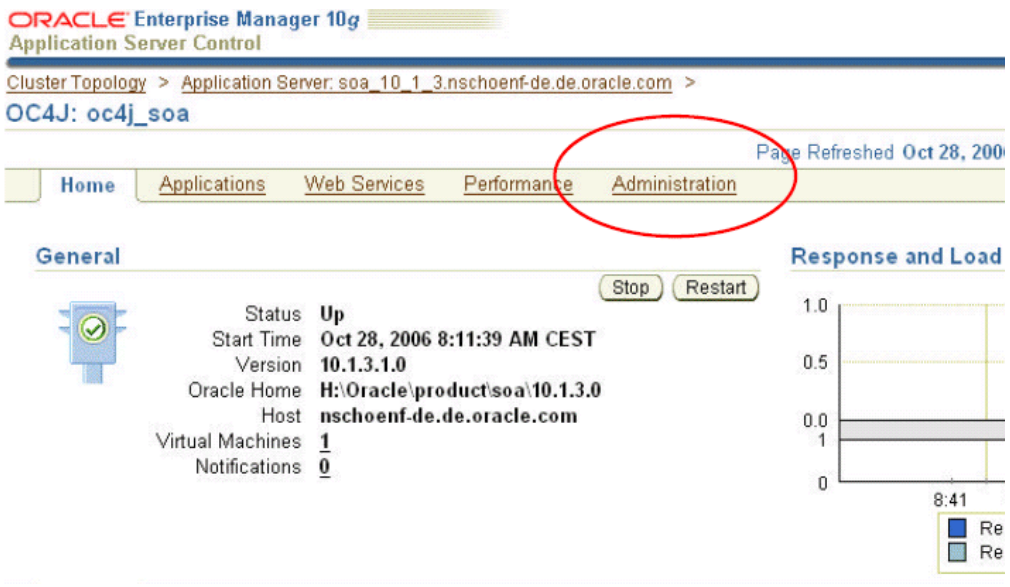
Members

View By: Application Servers

Start
Stop
Restart
Select All | Select None | Expand All | Collapse All

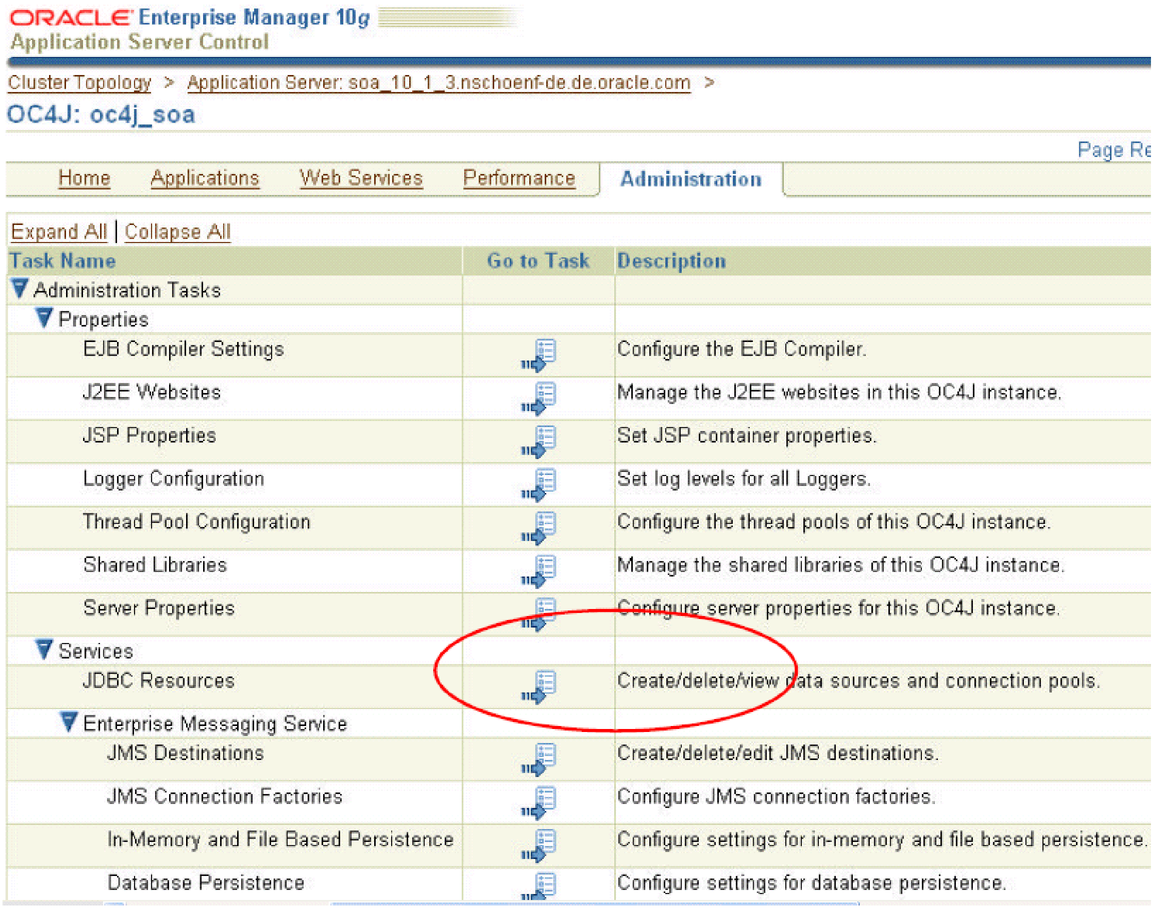
Select	Focus	Name	Status	Type	Category	Host	CPU (%)	Memory (MB)
☐		All Application Servers						
☐	⊕	soa_10_1_3.nschoenf-de.de.oracle.com		Application Server		nschoenf-de		
☐	⊕	home (JVMs: 1)	⇧	OC4J			0.32	129.88
☐		HTTP_Server	⇧	Oracle HTTP Server			0.08	7.21
☐	⊕	oc4j_soa (JVMs: 1)	⇧	OC4J			2.30	218.63

4. Click the **Administration** tab.


ORACLE Enterprise Manager 10g
Application Server Control
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com >
OC4J: oc4j_soa
Page Refreshed Oct 28, 200
Home
Applications
Web Services
Performance
Administration
General
Stop
Restart
Response and Load

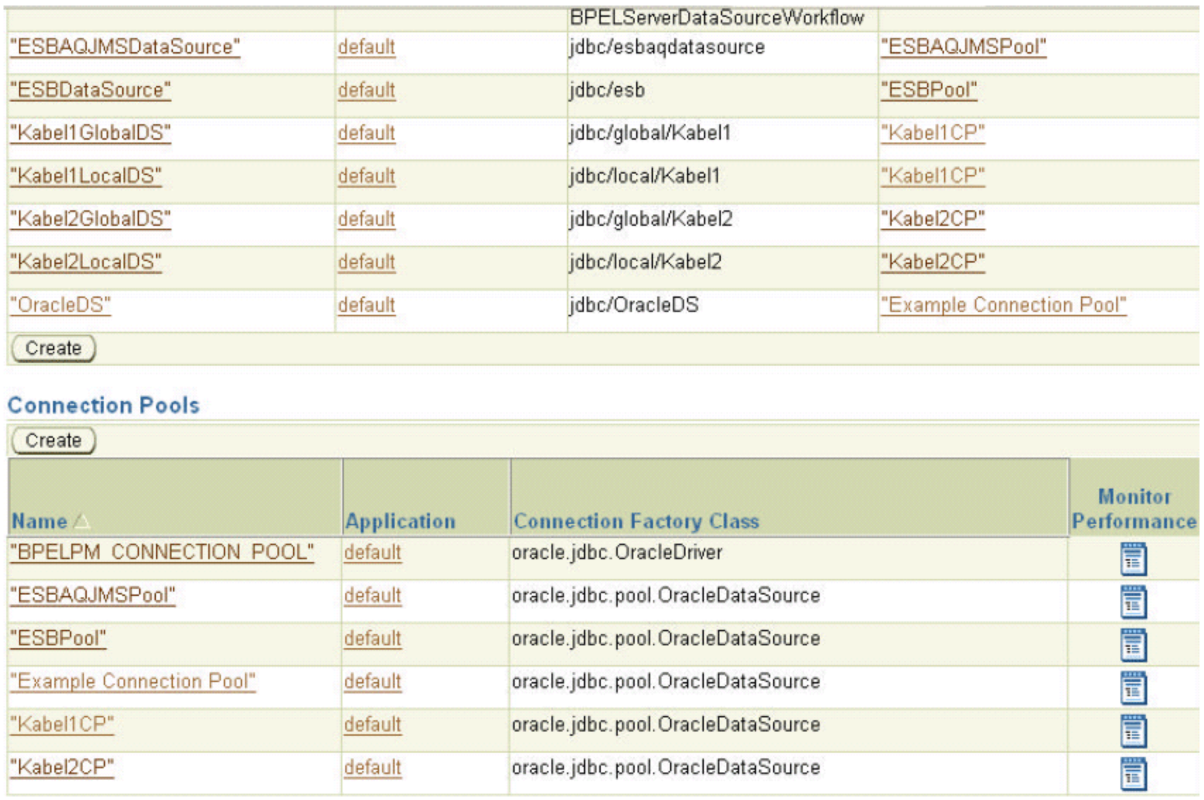
Status Up
Start Time Oct 28, 2006 8:11:39 AM CEST
Version 10.1.3.1.0
Oracle Home H:\Oracle\product\soa\10.1.3.0
Host nschoenf-de.de.oracle.com
Virtual Machines 1
Notifications 0

5. Click **Go to Task** in the **JDBC Resources** section.


ORACLE Enterprise Manager 10g
Application Server Control
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com >
OC4J: oc4j_soa
Page Re
Home
Applications
Web Services
Performance
Administration
Expand All | Collapse All

Task Name	Go to Task	Description
Administration Tasks		
Properties		
EJB Compiler Settings		Configure the EJB Compiler.
J2EE Websites		Manage the J2EE websites in this OC4J instance.
JSP Properties		Set JSP container properties.
Logger Configuration		Set log levels for all Loggers.
Thread Pool Configuration		Configure the thread pools of this OC4J instance.
Shared Libraries		Manage the shared libraries of this OC4J instance.
Server Properties		Configure server properties for this OC4J instance.
Services		
JDBC Resources		Create/delete/view data sources and connection pools.
Enterprise Messaging Service		
JMS Destinations		Create/delete/edit JMS destinations.
JMS Connection Factories		Configure JMS connection factories.
In-Memory and File Based Persistence		Configure settings for in-memory and file based persistence.
Database Persistence		Configure settings for database persistence.

6. Create two connection pools for users **kabel1** and **kabel2**. For this example, **Kabel1CP** and **Kabel2CP** are created.



		BPELServerDataSourceWorkflow	
"ESBAQJMSDataSource"	default	jdbc/esbaqdatasource	"ESBAQJMSPool"
"ESBDataSource"	default	jdbc/esb	"ESBPool"
"Kabel1GlobalDS"	default	jdbc/global/Kabel1	"Kabel1CP"
"Kabel1LocalDS"	default	jdbc/local/Kabel1	"Kabel1CP"
"Kabel2GlobalDS"	default	jdbc/global/Kabel2	"Kabel2CP"
"Kabel2LocalDS"	default	jdbc/local/Kabel2	"Kabel2CP"
"OracleDS"	default	jdbc/OracleDS	"Example Connection Pool"

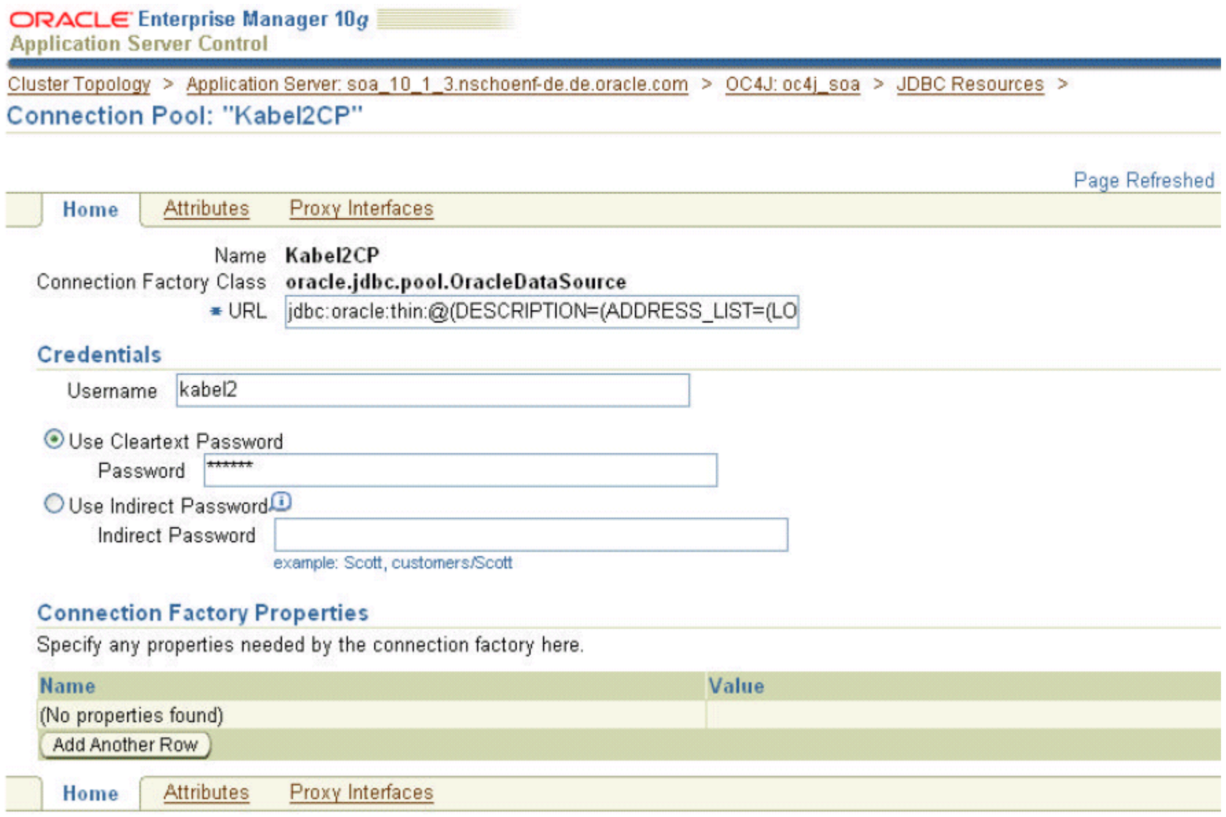
Create

Connection Pools

Create

Name	Application	Connection Factory Class	Monitor Performance
"BPELPM_CONNECTION_POOL"	default	oracle.jdbc.OracleDriver	
"ESBAQJMSPool"	default	oracle.jdbc.pool.OracleDataSource	
"ESBPool"	default	oracle.jdbc.pool.OracleDataSource	
"Example Connection Pool"	default	oracle.jdbc.pool.OracleDataSource	
"Kabel1CP"	default	oracle.jdbc.pool.OracleDataSource	
"Kabel2CP"	default	oracle.jdbc.pool.OracleDataSource	

The page in which to create the connection pool is shown below.



Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com > OC4J: oc4j_soa > JDBC Resources >

Connection Pool: "Kabel2CP"

Page Refreshed

Home | Attributes | Proxy Interfaces

Name	Kabel2CP
Connection Factory Class	oracle.jdbc.pool.OracleDataSource
* URL	jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(LO

Credentials

Username: kabel2

[✓ Use Cleartext Password]

Password: ******

Use Indirect Password

Indirect Password:

example: Scott, customers/Scott

Connection Factory Properties

Specify any properties needed by the connection factory here.

Name	Value
(No properties found)	

Add Another Row

Home | Attributes | Proxy Interfaces

After creating the connection pools, the `data-sources.xml` file (located under *`SOA_ORACLE_HOME`*`\j2ee\`*`oc4j_soa_container`*`\config`) should include two additional entries like those shown in the following example:

```
<connection-pool name="Kabel1CP">
<connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
user="kabel1" password="oracle10g"
url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(LOAD_BALANCE=on)(ADDRESS=(PR
OTOCOL=tcp)(HOST=nschoenf-de)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=soa)))"/>
</connection-pool>
<connection-pool name="Kabel2CP">
<connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
user="kabel2" password="oracle10g"
url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(LOAD_BALANCE=on)(ADDRESS=(PR
OTOCOL=tcp)(HOST=nschoenf-de)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=soa)))"/>
</connection-pool>
```

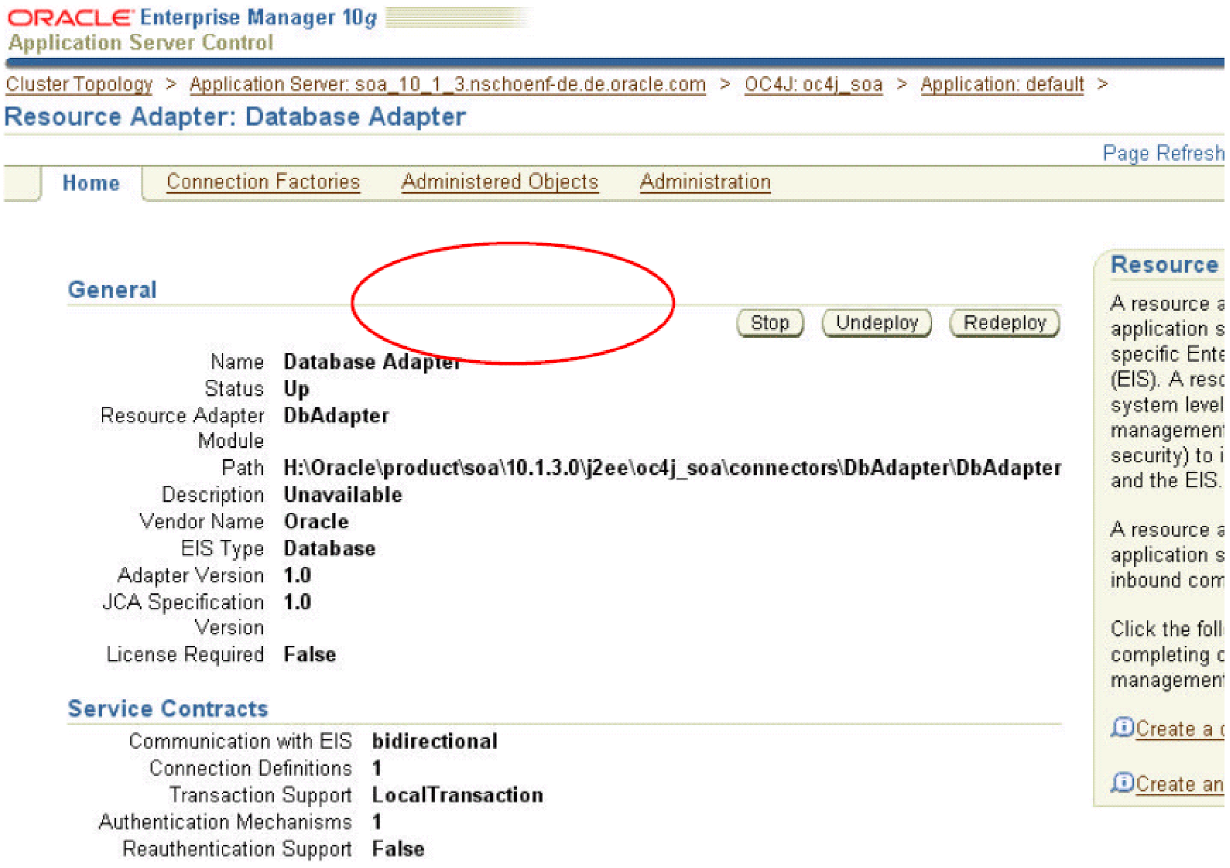
7. Create a new data source using either Oracle Enterprise Manager 10*g* Application Server Control Console or manually editing the `data-sources.xml` file (located under *`SOA_ORACLE_HOME`*`\`*`oc4j_soa_container`*`\config`). The file should include two additional entries like those shown in the following example:

```
<managed-data-source connection-pool-name="Kabel1CP" jndiname="
jdbc/local/Kabel1" name="Kabel1LocalDS" tx-level="local"/>
<managed-data-source connection-pool-name="Kabel2CP" jndiname="
jdbc/local/Kabel2" name="Kabel2LocalDS" tx-level="local"/>
```

Note: Local transactions are used in this use case because only the data from the different data sources must be queried. The sample is not tested with XA data sources.

Setting Up the Database Adapter

1. Go to `http://`*`host_name`*`:`*`HTTP_port`*`/em/console/ias/oc4j/applications`.
2. Select **Modules** from the drop-down list.
3. Select **DBAdapter** in the **Name** column.


ORACLE Enterprise Manager 10g
Application Server Control
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com > OC4J: oc4j_soa > Application: default >
Page Refresh
Home
Connection Factories
Administered Objects
Administration
Stop
Undeploy
Redeploy

Resource Adapter: Database Adapter

General

Name	Database Adapter
Status	Up
Resource Adapter Module	DbAdapter
Path	H:\Oracle\product\soa\10.1.3.0\j2ee\oc4j_soa\connectors\DbAdapter\DbAdapter
Description	Unavailable
Vendor Name	Oracle
EIS Type	Database
Adapter Version	1.0
JCA Specification Version	1.0
License Required	False

Service Contracts

Communication with EIS	bidirectional
Connection Definitions	1
Transaction Support	LocalTransaction
Authentication Mechanisms	1
Reauthentication Support	False

Resource

A resource a
application s
specific Ente
(EIS). A reso
system level
management
security) to i
and the EIS.

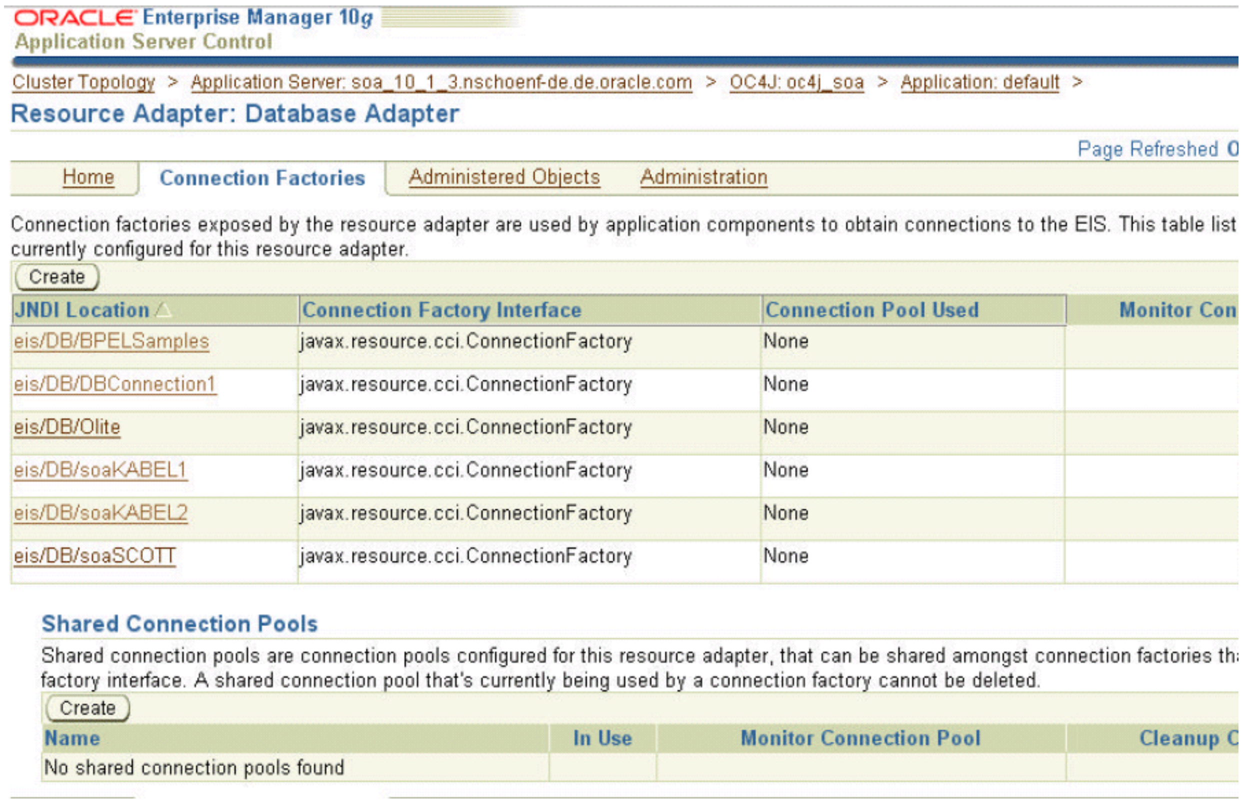
A resource a
application s
inbound com

Click the foll
completing c
management

Create a c

Create an

4. Click the **Connection Factories** tab.
5. Configure a new connection factory for the database adapter to point into the previously-created data source.


ORACLE Enterprise Manager 10g
Application Server Control

Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com > OC4J: oc4j_soa > Application: default >

Resource Adapter: Database Adapter

Page Refreshed O

Home | Connection Factories | Administered Objects | Administration

Connection factories exposed by the resource adapter are used by application components to obtain connections to the EIS. This table list currently configured for this resource adapter.

Create

JNDI Location	Connection Factory Interface	Connection Pool Used	Monitor Con
eis/DB/BPELSamples	javax.resource.cci.ConnectionFactory	None	
eis/DB/DBConnection1	javax.resource.cci.ConnectionFactory	None	
eis/DB/Olite	javax.resource.cci.ConnectionFactory	None	
eis/DB/soaKABEL1	javax.resource.cci.ConnectionFactory	None	
eis/DB/soaKABEL2	javax.resource.cci.ConnectionFactory	None	
eis/DB/soaSCOTT	javax.resource.cci.ConnectionFactory	None	

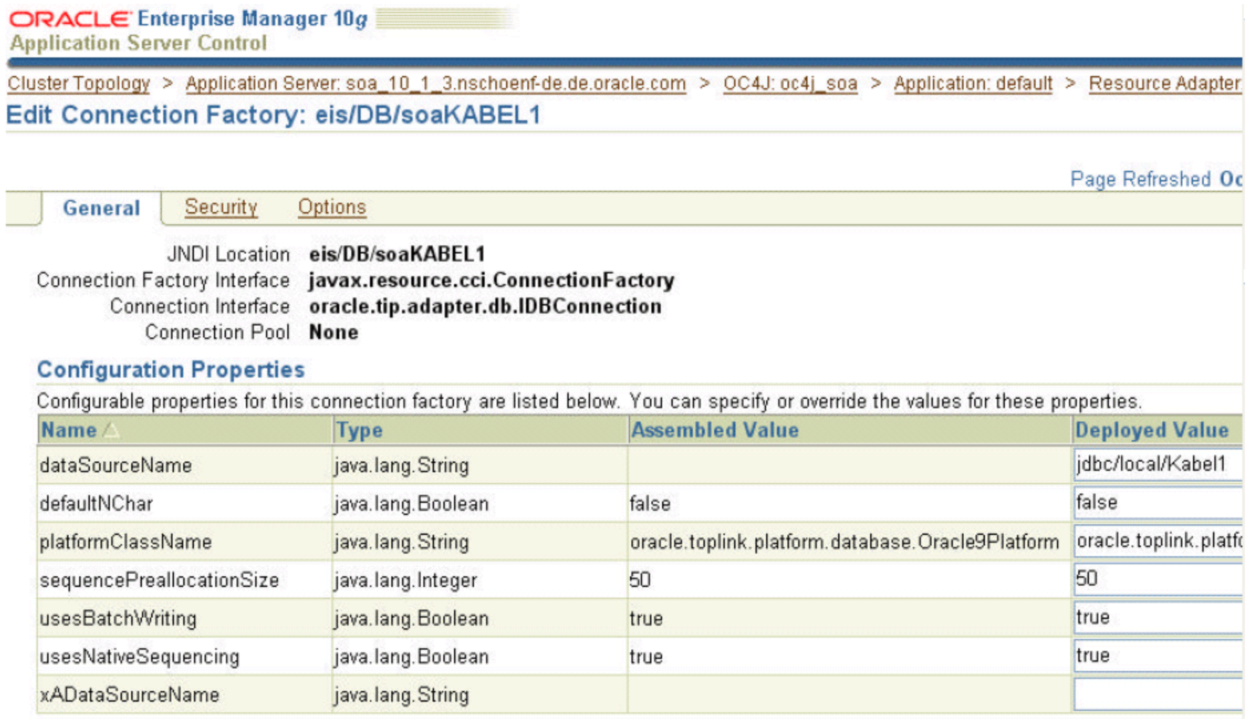
Shared Connection Pools

Shared connection pools are connection pools configured for this resource adapter, that can be shared amongst connection factories th factory interface. A shared connection pool that's currently being used by a connection factory cannot be deleted.

Create

Name	In Use	Monitor Connection Pool	Cleanup C
No shared connection pools found			

6. Click **Create**.
7. Configure the new connection factory for the database adapter, as shown below.


ORACLE Enterprise Manager 10g
Application Server Control

Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com > OC4J: oc4j_soa > Application: default > Resource Adapter

Edit Connection Factory: eis/DB/soaKABEL1

Page Refreshed Oc

General | Security | Options

JNDI Location	eis/DB/soaKABEL1
Connection Factory Interface	javax.resource.cci.ConnectionFactory
Connection Interface	oracle.tip.adapter.db.IDBConnection
Connection Pool	None

Configuration Properties

Configurable properties for this connection factory are listed below. You can specify or override the values for these properties.

Name	Type	Assembled Value	Deployed Value
dataSourceName	java.lang.String		jdbc/local/Kabel1
defaultNChar	java.lang.Boolean	false	false
platformClassName	java.lang.String	oracle.toplink.platform.database.Oracle9Platform	oracle.toplink.platfo
sequencePreallocationSize	java.lang.Integer	50	50
usesBatchWriting	java.lang.Boolean	true	true
usesNativeSequencing	java.lang.Boolean	true	true
xADataSourceName	java.lang.String		

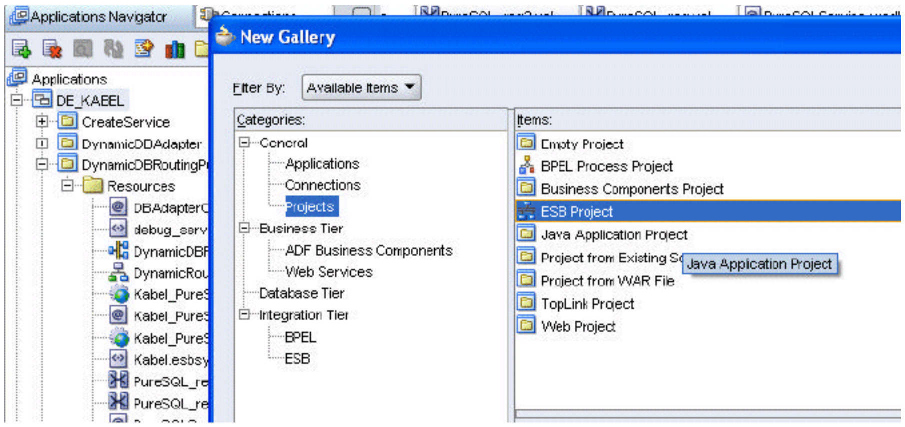
You can also manually modify the `oc4j-ra.xml` file (located under *`SOA_ORACLE_HOME`*`\j2ee\`*`oc4j_soa_`* *container*`\application-deployments\DbAdapter`) by adding the following entry:

```
<connector-factory location="eis/DB/soaKABEL1" connector-name="DbAdapter">
<config-property name="xADataSourceName" value=""/>
<config-property name="dataSourceName" value="jdbc/local/Kabel1"/>
<config-property name="platformClassName"
value="oracle.toplink.platform.database.Oracle9Platform"/>
<config-property name="usesNativeSequencing" value="true"/>
<config-property name="sequencePreallocationSize" value="50"/>
<config-property name="defaultNChar" value="false"/>
<config-property name="usesBatchWriting" value="true"/>
<connection-pooling use="none">
</connection-pooling>
<security-config use="none">
</security-config>
<connectionfactoryinterface>
javax.resource.cci.ConnectionFactory</connectionfactory-interface>
</connector-factory>
```

8. Restart Oracle Application Server.

Modeling the Use Case with ESB Designer

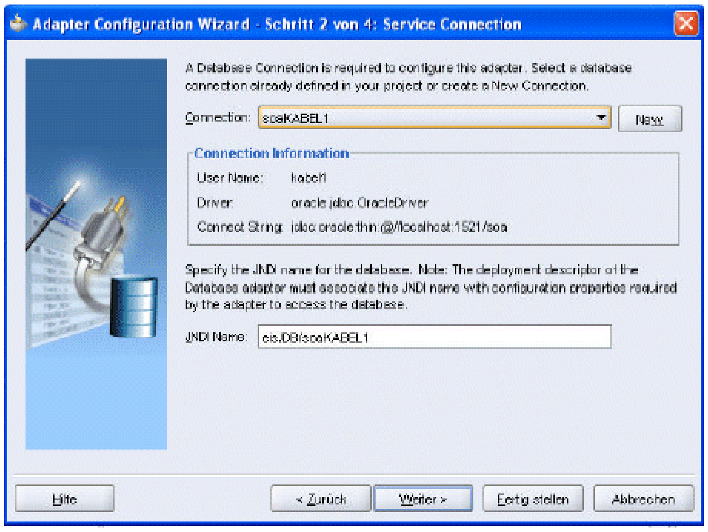
1. Start Oracle JDeveloper.
2. Select **New** > **Application** from the **File** main menu and create a workspace (for this example, named `TestPureSql`).
3. Create a new ESB project.



4. Select **Adapter Services** in the **Component Palette**.
5. Drag and drop a **Database Adapter** into the ESB project area.

   The Create Database Adapter Service window appears.
6. Enter a name. For this example, **PureSQLService** is entered.
7. Click the **Configure adapter service wsdl** icon to the right of the **WSDL File** field.

   The Adapter Configuration Wizard appears.
8. Create a new service based on the previously-created database connection to user **Kabel1**.


Adapter Configuration Wizard - Schritt 2 von 4: Service Connection

A Database Connection is required to configure this adapter. Select a database connection already defined in your project or create a New Connection.

Connection: soaKABEL1

Connection Information

User Name: kabel1

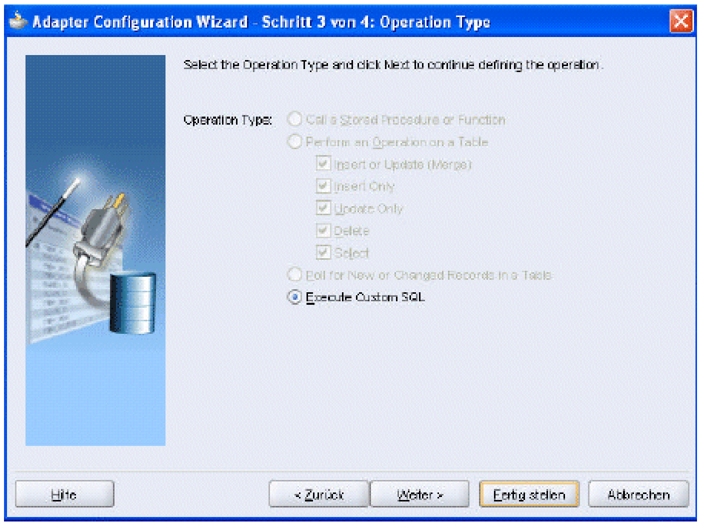
Driver: oracle.jdbc.OracleDriver

Connect String: jdbc:oracle:thin:@//localhost:1521/soa

Specify the JNDI name for the database. Note: The deployment descriptor of the Database adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

JNDI Name: eis/DB/soaKABEL1

9. Click **Execute Custom SQL**.


Adapter Configuration Wizard - Schritt 3 von 4: Operation Type

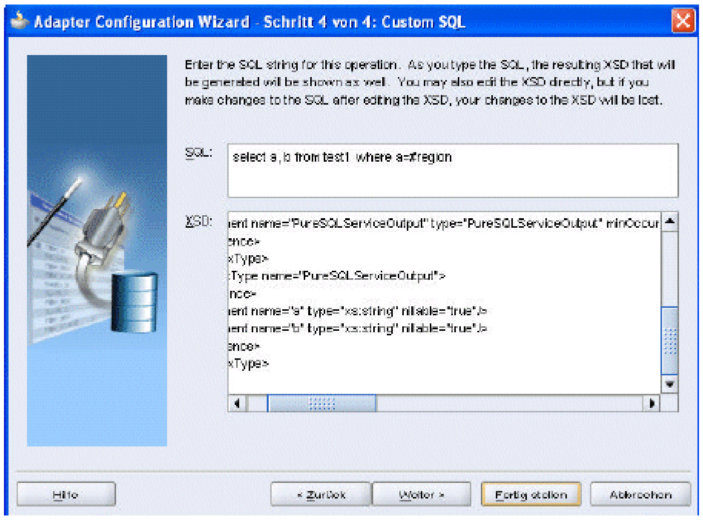
Select the Operation Type and click Next to continue defining the operation.

Operation Type:
- Call a Stored Procedure or Function
- Perform an Operation on a Table
  - ✓ Insert or Update (Merge)
  - ✓ Insert Only
  - ✓ Update Only
  - ✓ Delete
  - ✓ Select
- Poll for New or Changed Records in a Table
- [✓ Execute Custom SQL]

10. Enter the SQL statement you want to use. For this example, the following statement is entered:

```
'select a,b from test1 where a=#region'
```

This statement is used to demonstrate dynamic routing. Complex SQL statements (using union, minus, and subselects) still have some issues in design time, but not in runtime.



Enter the SQL string for this operation. As you type the SQL, the resulting XSD that will be generated will be shown as well. You may also edit the XSD directly, but if you make changes to the SQL after editing the XSD, your changes to the XSD will be lost.

SQL: select a, b from test1 where a=#region

XSD:

```
ent name="PureSQLServiceOutput" type="PureSQLServiceOutput" minOccur
nce>
xType>
Type name="PureSQLServiceOutput">
nce>
ent name="a" type="xs:string" nillable="true"/>
ent name="b" type="xs:string" nillable="true"/>
nce>
xType>
```

The XML Schema is now generated for input and output parameters. The file with the schema is named *`adapter_service_name`*`.xsd` and is located in your Oracle JDeveloper project directory.

11. Manually copy a valid `DBAdapterOutboundHeader.wsdl` file into your project directory. The file `DBAdapterOutboundHeader.wsdl`, normally created if you use other operations with the database adapter, is not automatically created if you use pure SQL.

12. Add the namespace declaration for the database adapter header in the WSDL file created by the Adapter Configuration Wizard.

```
<definitions
name="PureSQLService"
targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/db/PureSQLService/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/db/PureSQLService/"
xmlns:db="http://xmlns.oracle.com/pcbpel/adapter/db/PureSQLService"
xmlns:plt="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:jca="http://xmlns.oracle.com/pcbpel/wsdl/jca/"
xmlns:hdr="http://xmlns.oracle.com/pcbpel/adapter/db/"
```

13. Import the XML schema describing the database adapter outbound header in the WSDL file created by the Adapter Configuration Wizard.

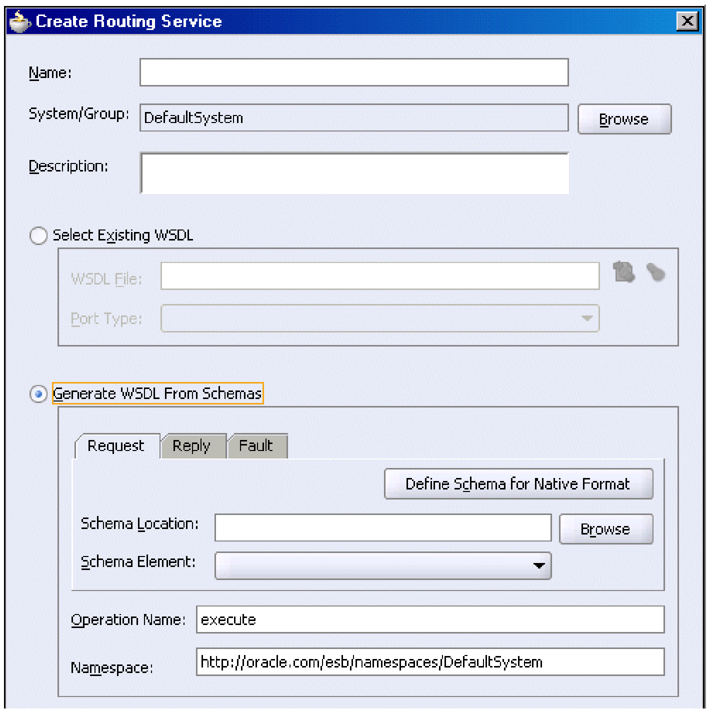
```
<import namespace="http:/ / xmlns.oracle.com/ pcbpel/ adapter/ db/ "
location="DBAdapterOutboundHeader.wsdl"/ >
```

14. Add the following line in the WSDL file created by the Adapter Configuration Wizard:

```
<jca:operation
SqlString="select a, b from test1 where a=#region"
InteractionSpec="oracle.tip.adapter.db.DBPureSQLInteractionSpec" >
</jca:operation>
<input/>
<jca:header message="hdr:OutboundHeader_msg" part="outboundHeader"/>
<output/>
</operation>
```

Creating the Routing Service

1. Select **ESB Services** in the **Component Palette**.
2. Drag and drop a **Routing Service** into the ESB project area. The Create Routing Service window appears.



System/Group: DefaultSystem

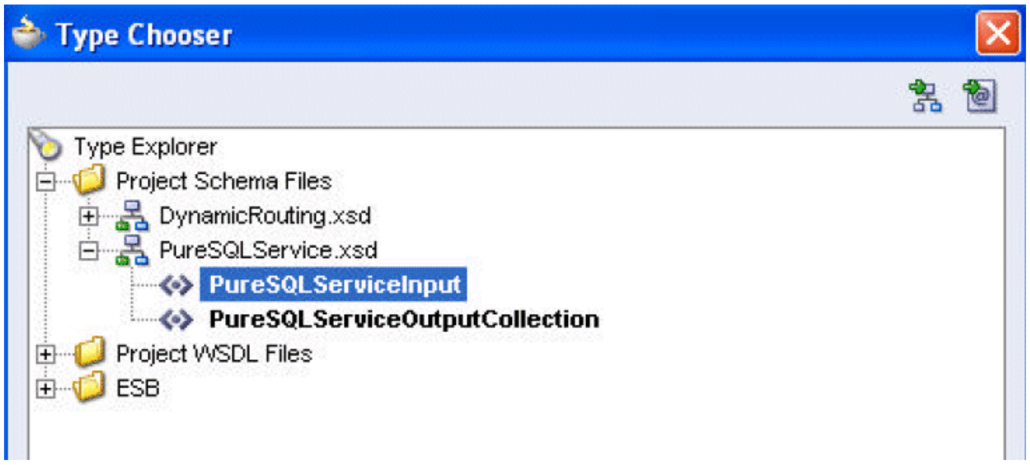
Operation Name: execute

Namespace: http://oracle.com/esb/namespaces/DefaultSystem

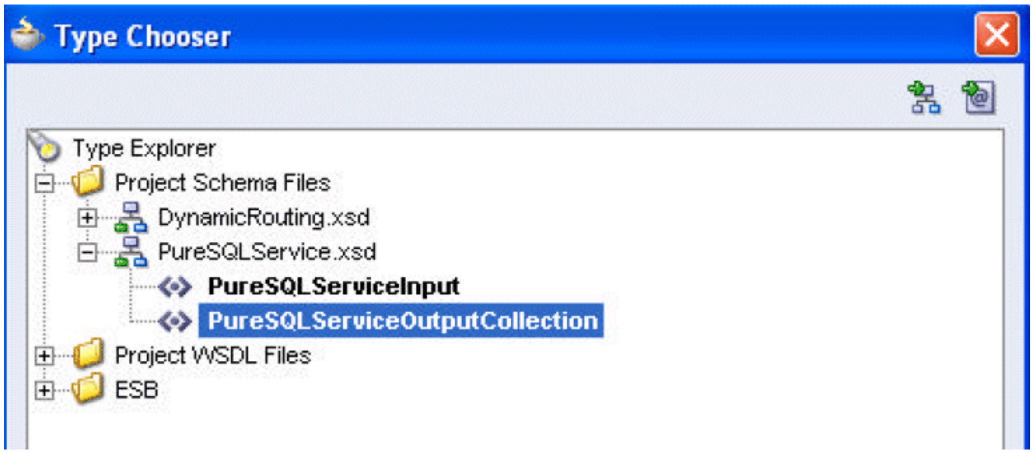
3. Enter the following details:

Field	Description
Name	Enter `PureSQLRS`.
System/Group	Use the default value. **Note:** You can also select a custom system group.

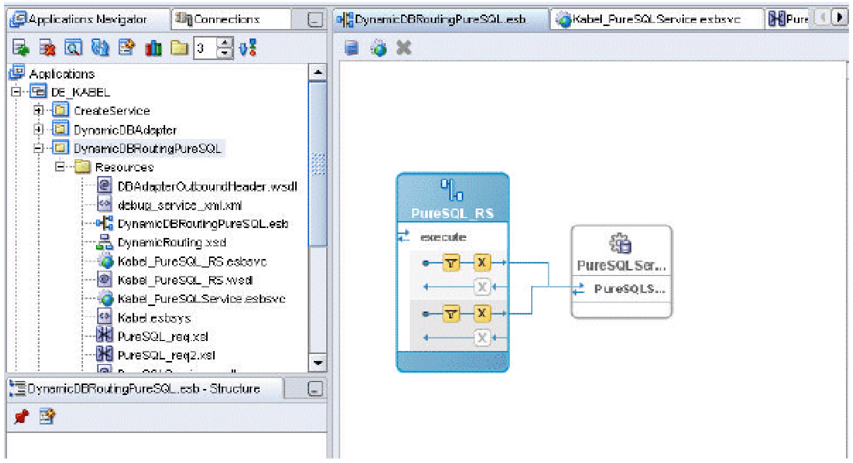
4. Select the **Generate WSDL From Schemas** radio button.
5. Click the **Request** tab.
6. Click **Browse** and select the XML schema previously generated by the Adapter Configuration Wizard in step 10 on page 2-41.



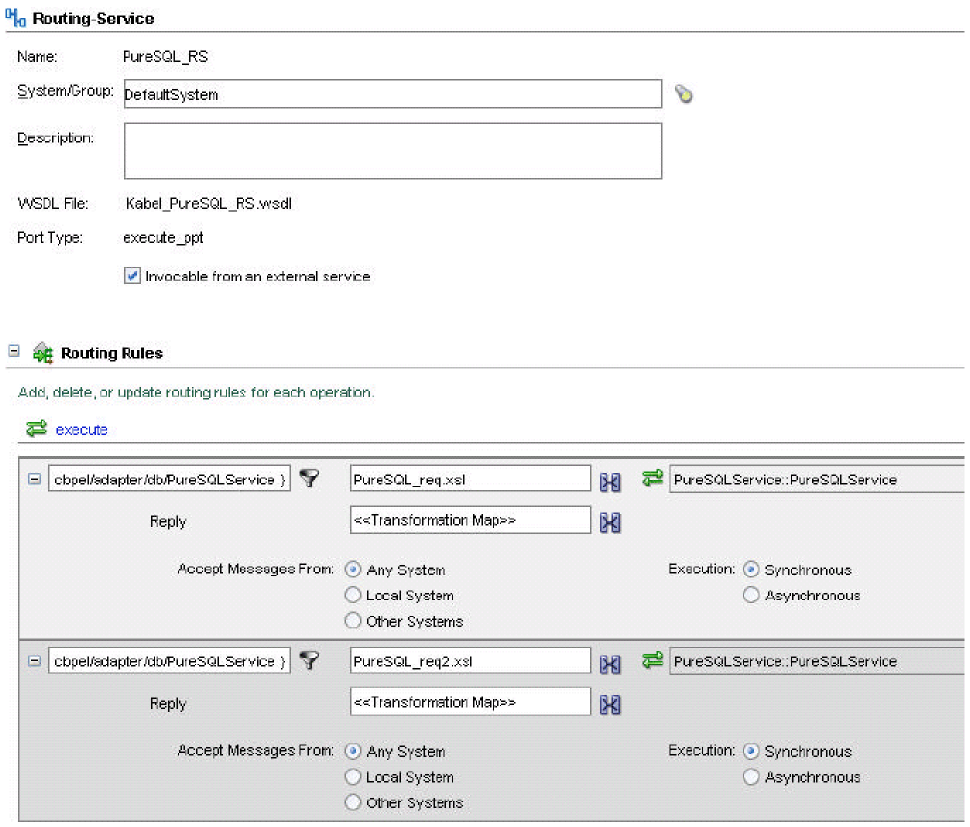
7. Repeat step 2 for the **Reply** tab.



You must now map the database adapter service to the routing service.



8. Double-click the routing service. You must add two routing rules to instruct the database adapter to execute the query using different data sources. The result should be similar to the following.


Routing-Service

Name: PureSQL_RS

System/Group: DefaultSystem

Description:

WSDL File: Kabel_PureSQL_RS.wsdl

Port Type: execute_ppt

[✓] Invocable from an external service

Routing Rules

Add, delete, or update routing rules for each operation.

execute

cbpel/adapter/db/PureSQLService } | PureSQL_req.xsl | PureSQLService::PureSQLService

Reply | <<Transformation Map>>

Accept Messages From: [✓ Any System] Local System Other Systems

Execution: [✓ Synchronous] Asynchronous

cbpel/adapter/db/PureSQLService } | PureSQL_req2.xsl | PureSQLService::PureSQLService

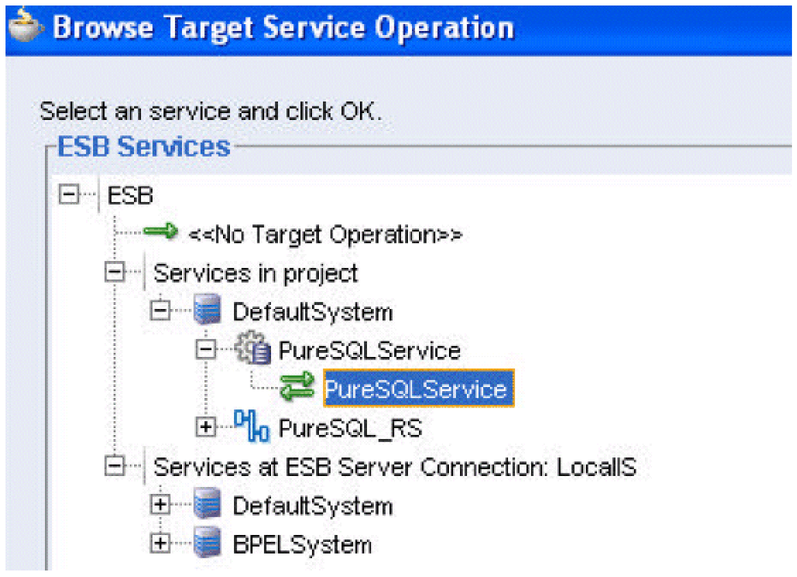
Reply | <<Transformation Map>>

Accept Messages From: [✓ Any System] Local System Other Systems

Execution: [✓ Synchronous] Asynchronous

9. Expand the **Routing Rules** section and click the **+** sign. The Routing Service Configuration Wizard appears.

10. Select the database adapter service created in step 10 on page 2-41 (in this example, **PureSQLService**) and click **OK**. This returns you to the Routing Service Configuration Wizard.


Browse Target Service Operation

Select an service and click OK.

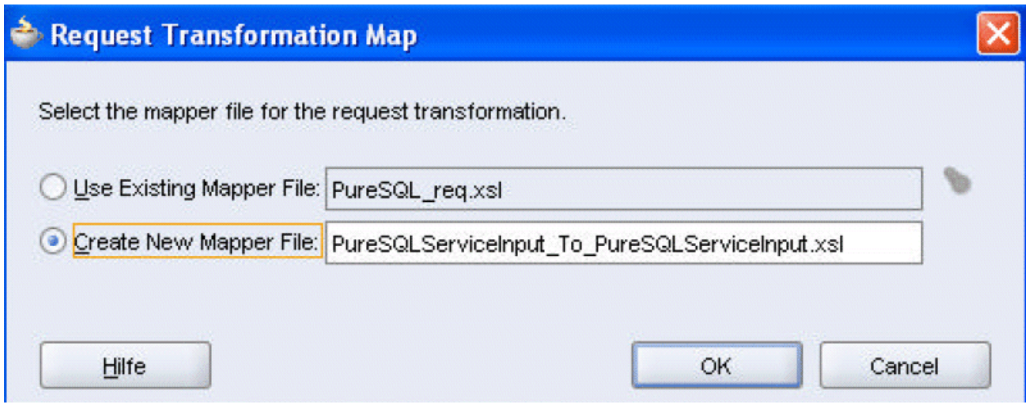
ESB Services

- ESB
  - <<No Target Operation>>
  - Services in project
    - DefaultSystem
      - PureSQLService
        - PureSQLService
      - PureSQL_RS
  - Services at ESB Server Connection: LocalIS
    - DefaultSystem
    - BPELSystem

11. Click the filter button within the newly-created routing rule to add a filter. The expression builder appears. Use the following expression to route the request to the data source owned by the database user **Kabel1**.

```
starts-with(/ out1:PureSQLServiceInput/ out1:region,"R1
```

12. Save your changes.
13. Click the transform button within the same routing rule area. This displays the Request Transformation Map window.



Request Transformation Map

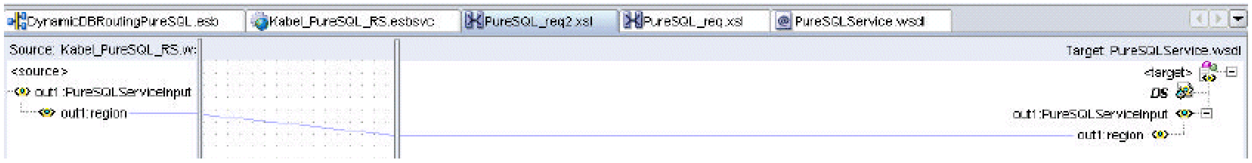
Select the mapper file for the request transformation.

( ) Use Existing Mapper File: PureSQL_req.xsl

(•) Create New Mapper File: PureSQLServiceInput_To_PureSQLServiceInput.xsl

Hilfe | OK | Cancel

14. Open the XSL file from the **Application Navigator**.
15. Create the mappings for the request by dragging and dropping the source to the target.



16. Modify the transformation map to apply header information. Click **Source** in the bottom left corner and add the following line to the XSL source.

```
<xsl:template match="/ ">
<xsl:variable name="DS"
select="ehdr:setOutboundHeader('/ dbhdr:OutboundDBHeaderType/
 dbhdr:dataSourceName',
'jdbc/local/Kabel1', 'dbhdr=http://xmlns.oracle.com/pcbpel/adapter/db/ ;')"/>
```

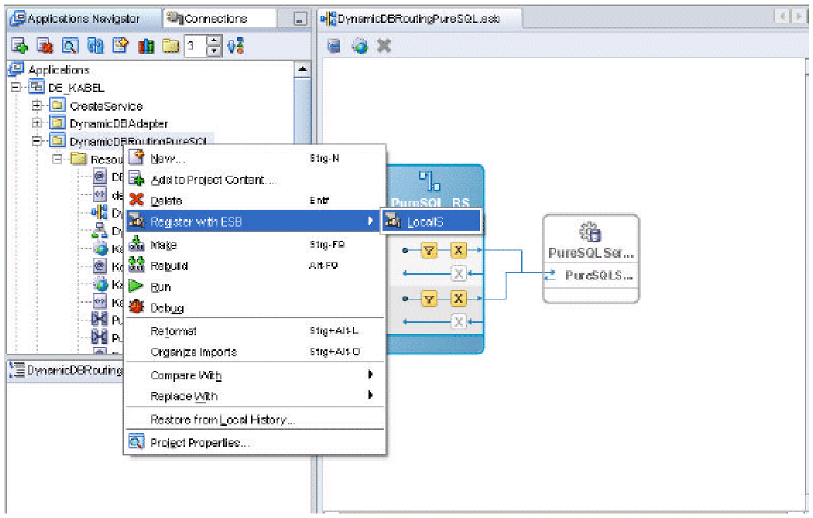
17. Add the namespace information:

```
. . .
xmlns:dbhdr=http:/ / xmlns.oracle.com/ pcbpel/ adapter/ db/
. . .
exclude-result-prefixes="xsl out1 ns0 tns plt jca hdr ns1 bpws ehdr hwf xp20
 ora ids orcl "dbhdr">
```

18. Save your changes.
19. Repeat steps 9 to 17 to add a new routing rule for selecting data from the data source related to database user **Kabel2**. In step 16, replace **'jdbc/local/Kabel1'** through **'jdbc/local/Kabel2'**.

Deploying the Use Case

1. Right-click your project and select **Register with ESB** from the context menu. This step requires that you have already set up application server and integration server connections in Oracle JDeveloper.



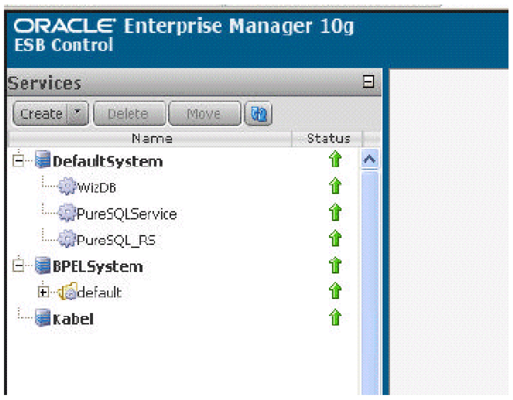
Testing the Use Case

Note: The sample was not tested with XA data sources.

1. Go to Oracle ESB Control:

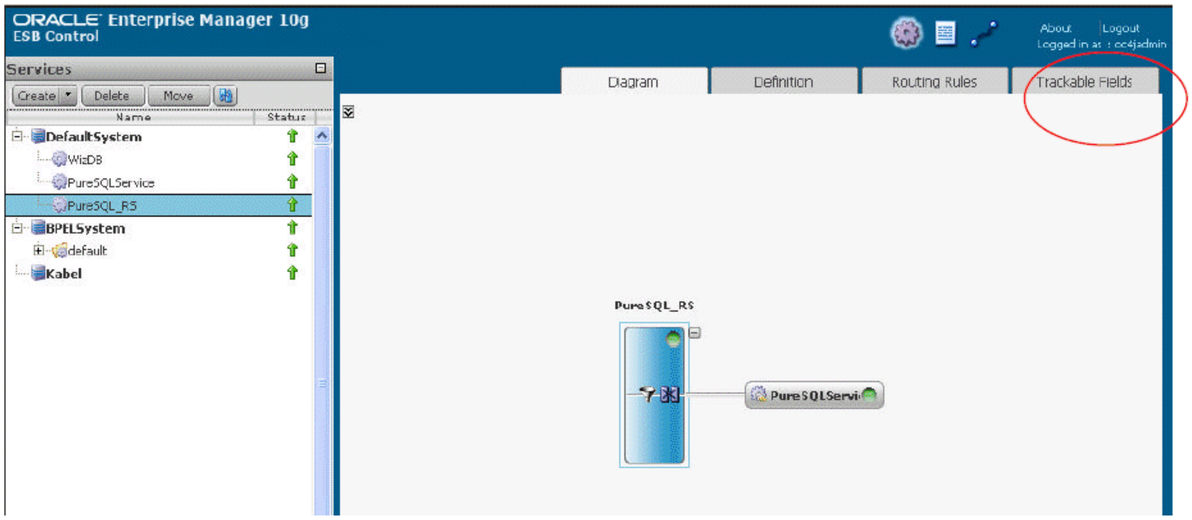
   ```
   http://host_name:HTTP_port/esb/
   ```

   You should see the routing service and the database adapter service in the **Services** panel.

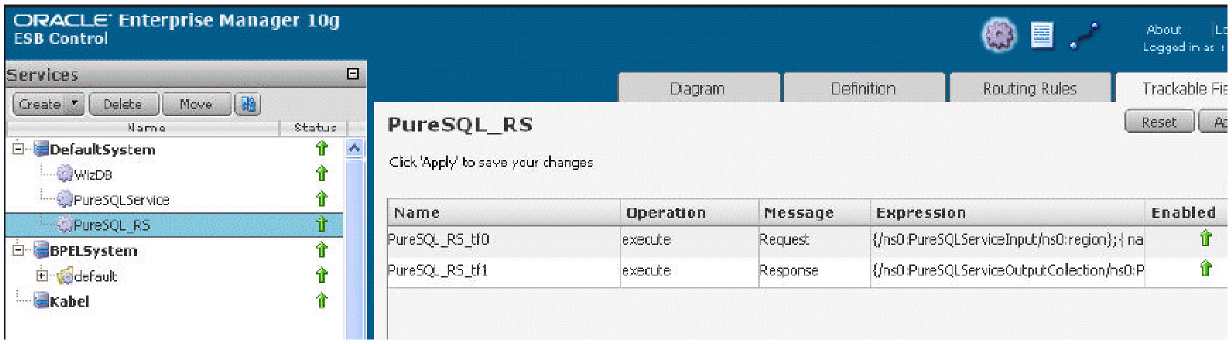


2. Click the routing service.

   The following window appears.



3. Click the **Trackable Fields** tab.
4. Define two trackable fields for tracking the request and response of the routing service.

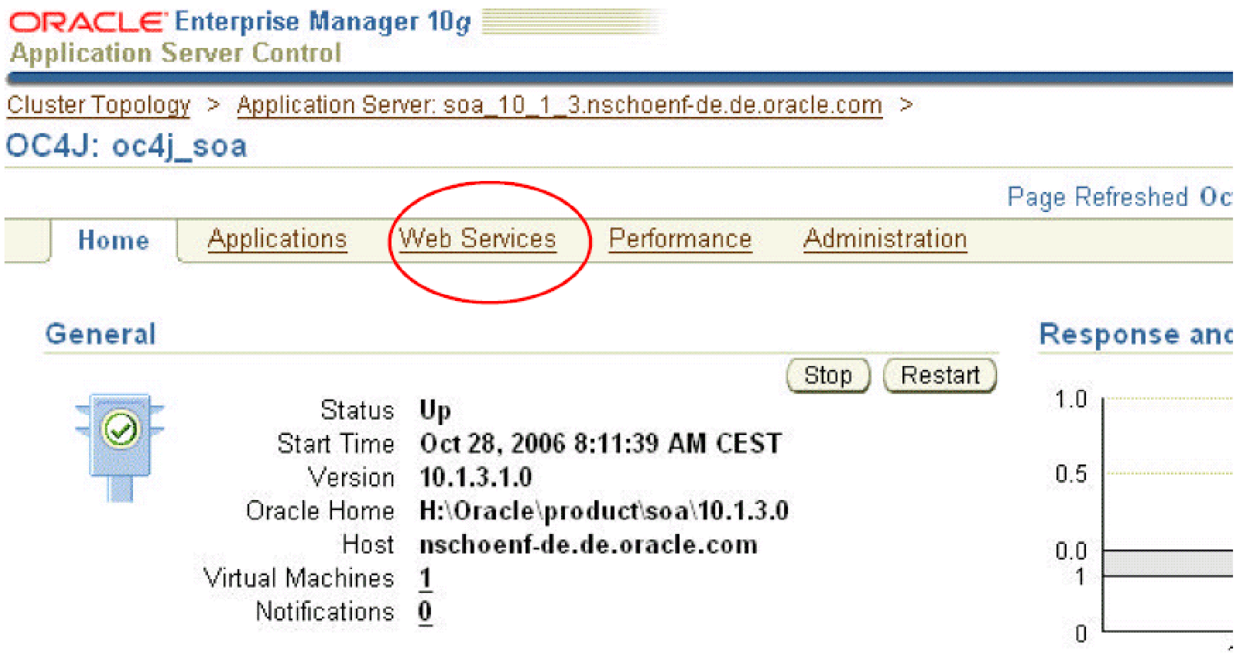


PureSQL_RS

Click 'Apply' to save your changes

Name	Operation	Message	Expression	Enabled
PureSQL_RS_tf0	execute	Request	{/ns0:PureSQLServiceInput/ns0:region};{ na	↑
PureSQL_RS_tf1	execute	Response	{/ns0:PureSQLServiceOutputCollection/ns0:P	↑

5. Apply your changes.
6. Go to Oracle Enterprise Manager 10*g* Application Server Control Console.
7. Click **Web Services**.


ORACLE Enterprise Manager 10g
Application Server Control

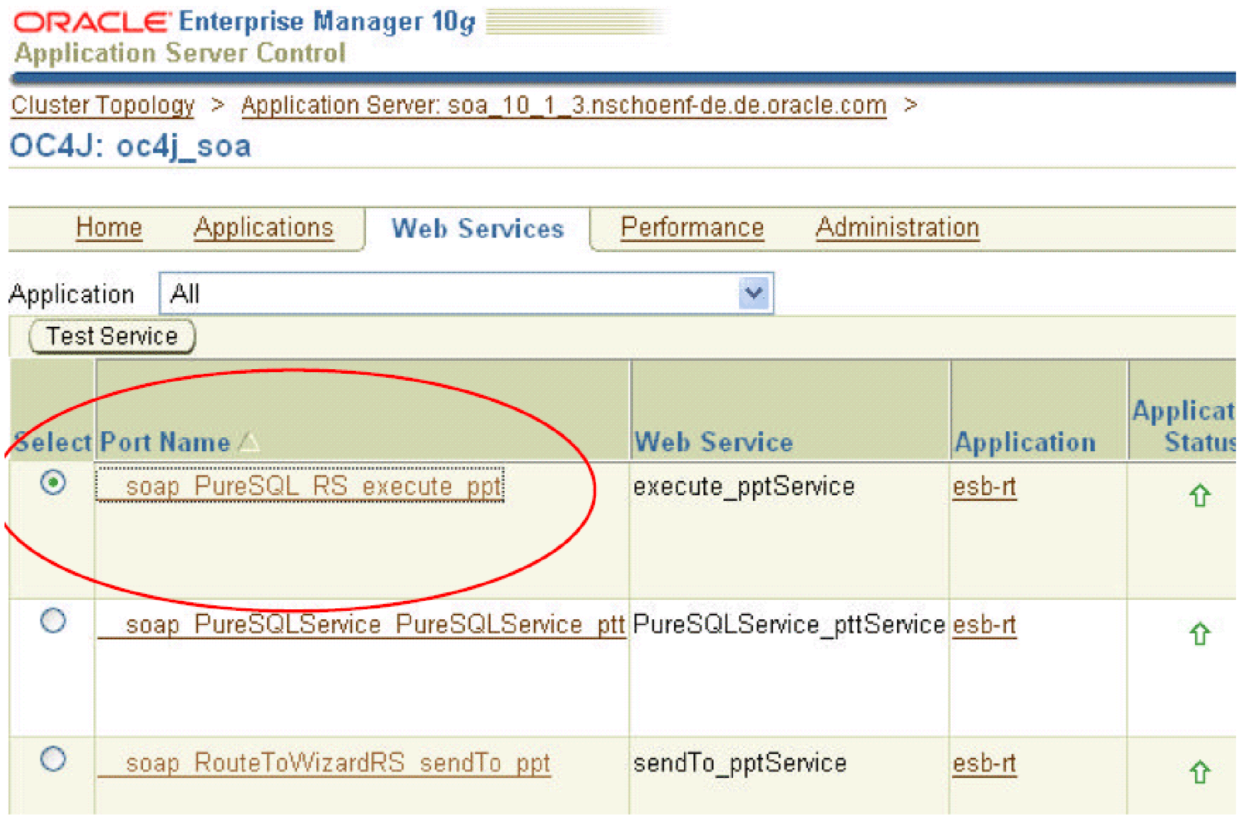
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com >

OC4J: oc4j_soa

Page Refreshed Oc

Status	Up
Start Time	Oct 28, 2006 8:11:39 AM CEST
Version	10.1.3.1.0
Oracle Home	H:\Oracle\product\soa\10.1.3.0
Host	nschoenf-de.de.oracle.com
Virtual Machines	1
Notifications	0

8. Select the Web service representing your routing service.


ORACLE Enterprise Manager 10g
Application Server Control

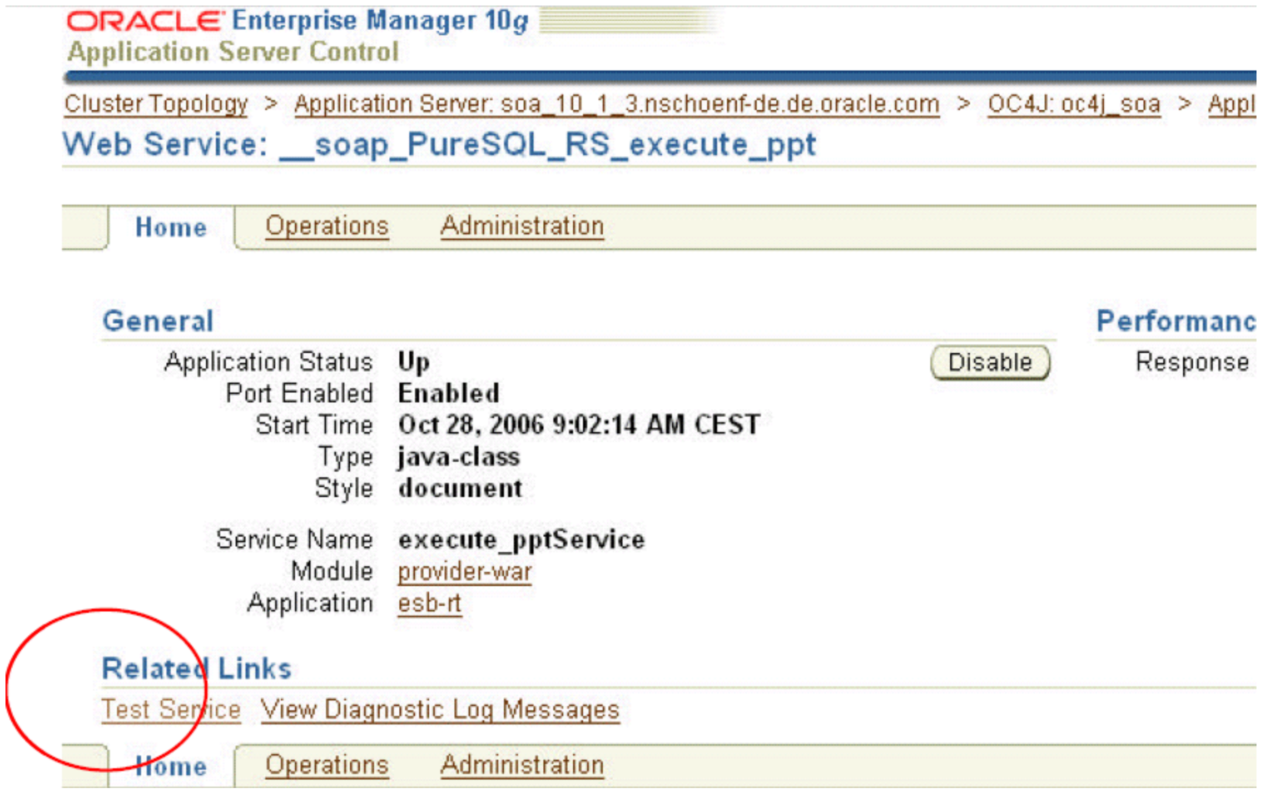
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com >

OC4J: oc4j_soa

Application: All

Select	Port Name	Web Service	Application	Applicat Status
●	soap_PureSQL_RS_execute_ppt	execute_pptService	esb-rt	⇧
○	soap_PureSQLService_PureSQLService_ptt	PureSQLService_pttService	esb-rt	⇧
○	soap_RouteToWizardRS_sendTo_ppt	sendTo_pptService	esb-rt	⇧

9. Click **Test Service**.


ORACLE Enterprise Manager 10g
Application Server Control
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com > OC4J: oc4j_soa > Appl
Home
Operations
Administration
Disable
Performanc
Response
Home
Operations
Administration

Web Service: __soap_PureSQL_RS_execute_ppt

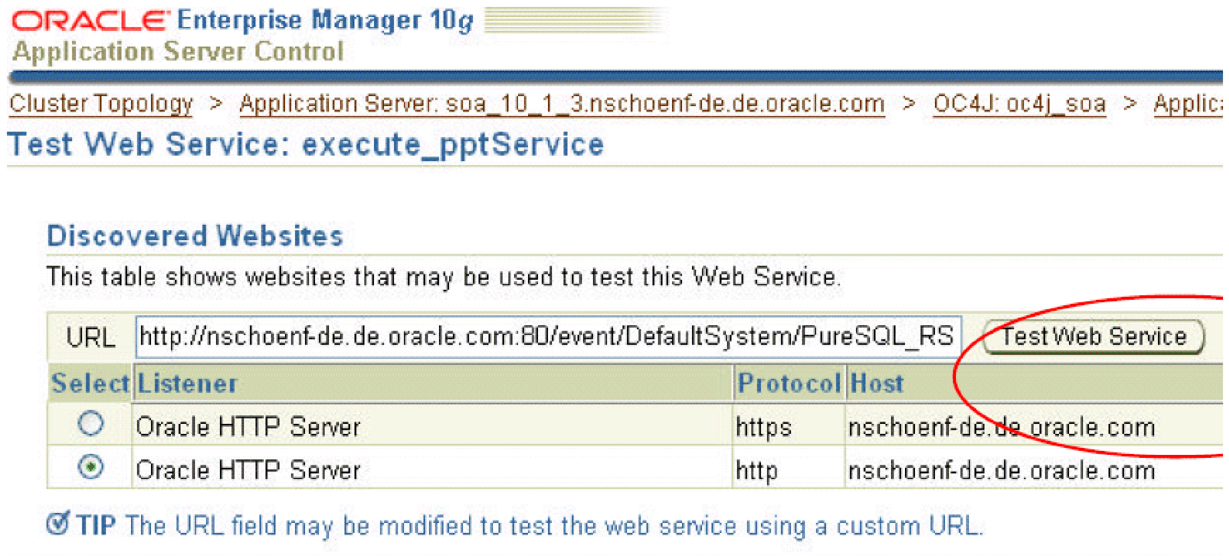
General

Application Status	Up
Port Enabled	Enabled
Start Time	Oct 28, 2006 9:02:14 AM CEST
Type	java-class
Style	document
Service Name	execute_pptService
Module	provider-war
Application	esb-rt

Related Links

Test Service View Diagnostic Log Messages

10. Select the HTTP server.
11. Click **Test Web Service**.


ORACLE Enterprise Manager 10g
Application Server Control
Cluster Topology > Application Server: soa_10_1_3.nschoenf-de.de.oracle.com > OC4J: oc4j_soa > Applic
Test Web Service

Test Web Service: execute_pptService

Discovered Websites

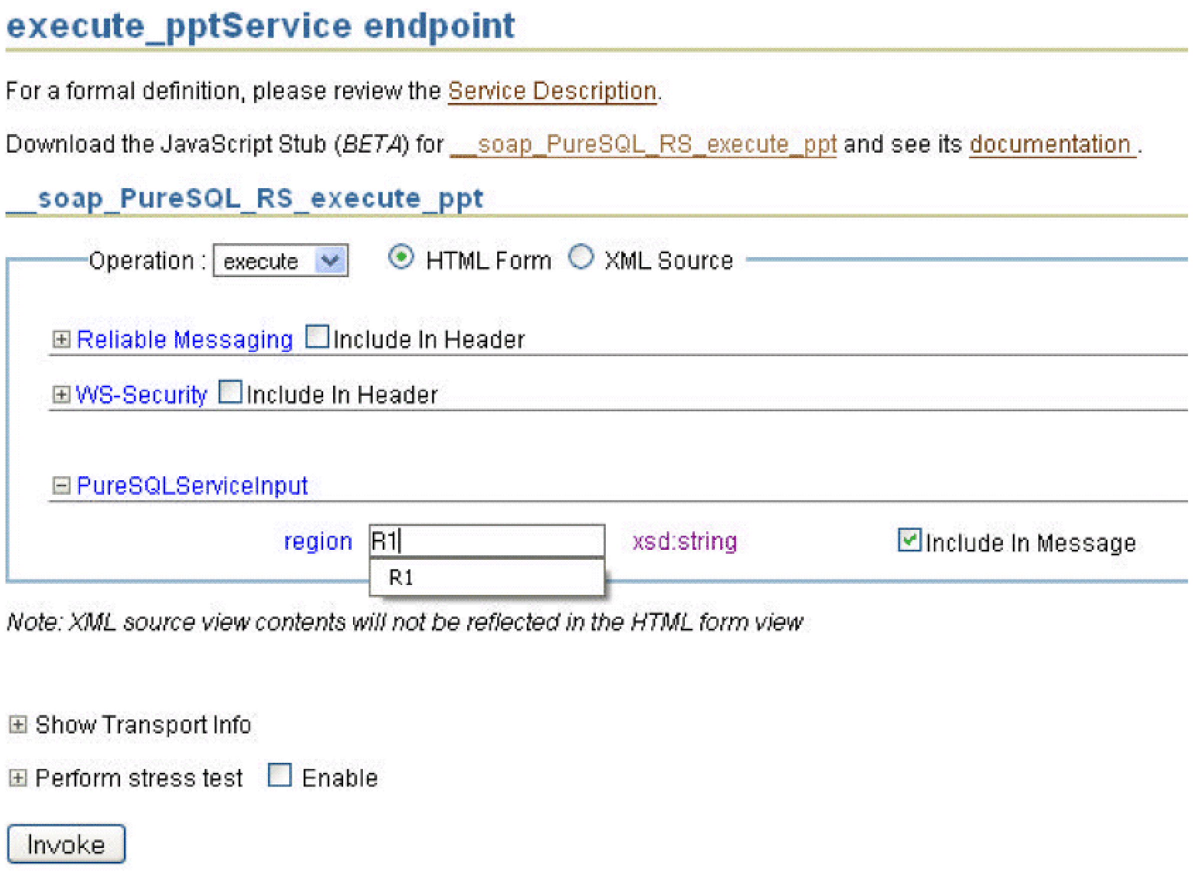
This table shows websites that may be used to test this Web Service.

URL http://nschoenf-de.de.oracle.com:80/event/DefaultSystem/PureSQL_RS

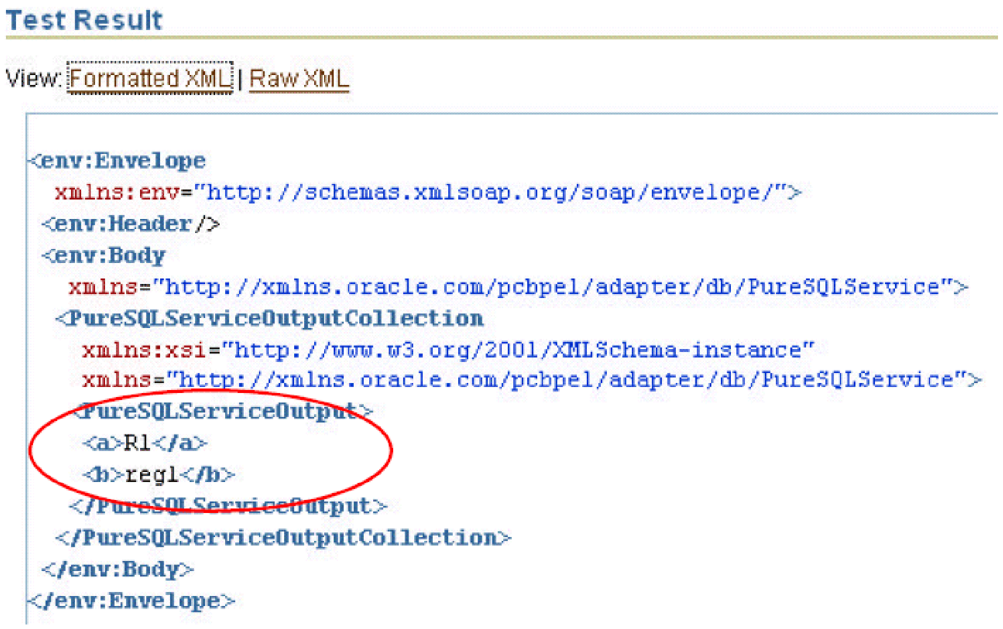
Select	Listener	Protocol	Host
○	Oracle HTTP Server	https	nschoenf-de.de.oracle.com
●	Oracle HTTP Server	http	nschoenf-de.de.oracle.com

TIP The URL field may be modified to test the web service using a custom URL.

12. Enter `R1` in the field region.


execute_pptService endpoint
For a formal definition, please review the Service Description.
Download the JavaScript Stub (BETA) for __soap_PureSQL_RS_execute_ppt and see its documentation .
__soap_PureSQL_RS_execute_ppt
Operation : execute
HTML Form
XML Source
Reliable Messaging
Include In Header
WS-Security
Include In Header
PureSQLServiceInput
region
R1
xsd:string
Include In Message
R1
Note: XML source view contents will not be reflected in the HTML form view
Show Transport Info
Perform stress test
Enable
Invoke

13. Click **Invoke**. The service should return the entries of table **TEST1** of user **Kabel1**.

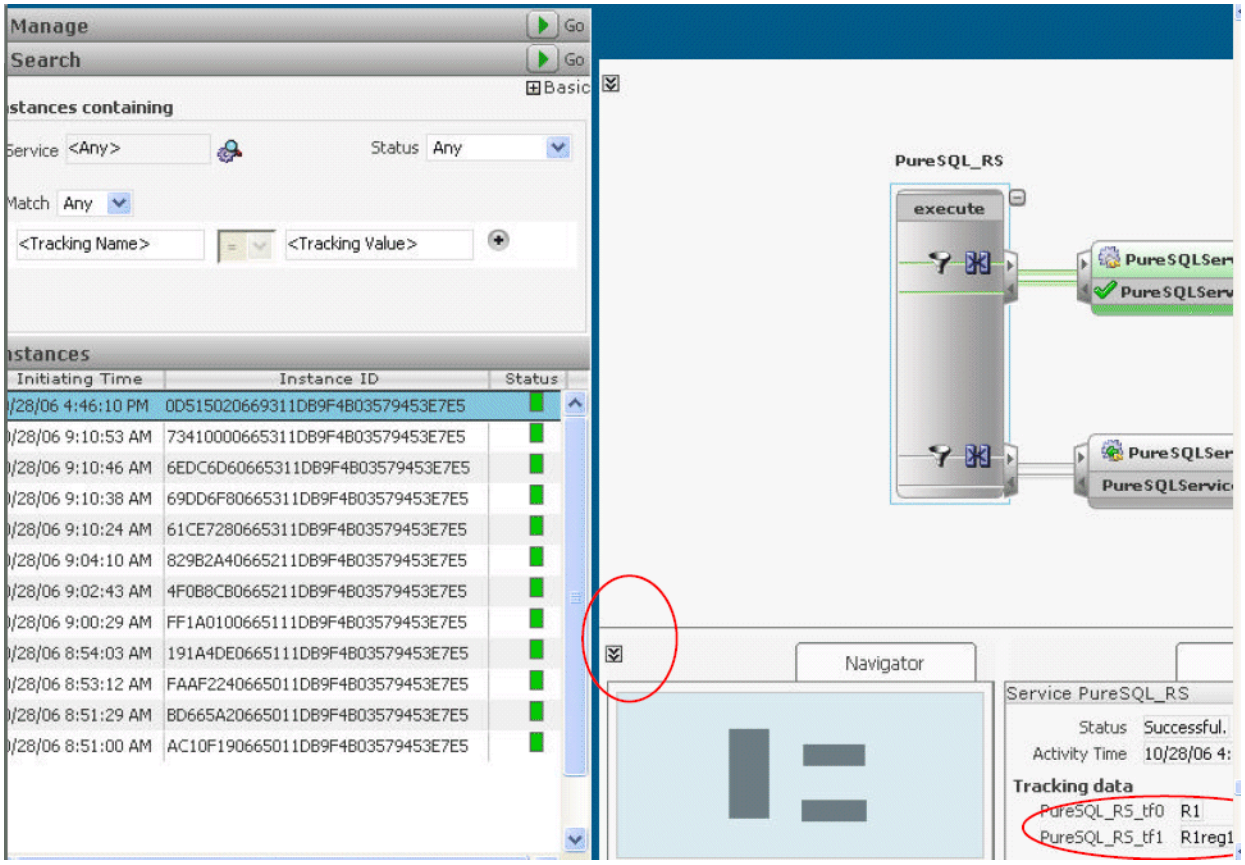


```
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header/>
 <env:Body
   xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/PureSQLService">
  <PureSQLServiceOutputCollection
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/PureSQLService">
   <PureSQLServiceOutput>
    <a>R1</a>
    <b>reg1</b>
   </PureSQLServiceOutput>
  </PureSQLServiceOutputCollection>
 </env:Body>
</env:Envelope>
```

14. Go to Oracle ESB Control and click the **Instances** button.

15. Click the most recently generated instance in the Instances page. This displays the message flow.

16. Click the arrows to display the contents of the tracking fields. The response message corresponds to the contents of the table **TEST1** owned by **Kabel1**.



stances containing

Service <Any>   Status Any

Match Any

<Tracking Name> = <Tracking Value>

stances

Initiating Time	Instance ID	Status
/28/06 4:46:10 PM	0D515020669311DB9F4B03579453E7E5	🟩
/28/06 9:10:53 AM	73410000665311DB9F4B03579453E7E5	🟩
/28/06 9:10:46 AM	6EDC6D60665311DB9F4B03579453E7E5	🟩
/28/06 9:10:38 AM	69DD6F80665311DB9F4B03579453E7E5	🟩
/28/06 9:10:24 AM	61CE7280665311DB9F4B03579453E7E5	🟩
/28/06 9:04:10 AM	829B2A40665211DB9F4B03579453E7E5	🟩
/28/06 9:02:43 AM	4F0B8CB0665211DB9F4B03579453E7E5	🟩
/28/06 9:00:29 AM	FF1A0100665111DB9F4B03579453E7E5	🟩
/28/06 8:54:03 AM	191A4DE0665111DB9F4B03579453E7E5	🟩
/28/06 8:53:12 AM	FAAF2240665011DB9F4B03579453E7E5	🟩
/28/06 8:51:29 AM	BD665A20665011DB9F4B03579453E7E5	🟩
/28/06 8:51:00 AM	AC10F190665011DB9F4B03579453E7E5	🟩

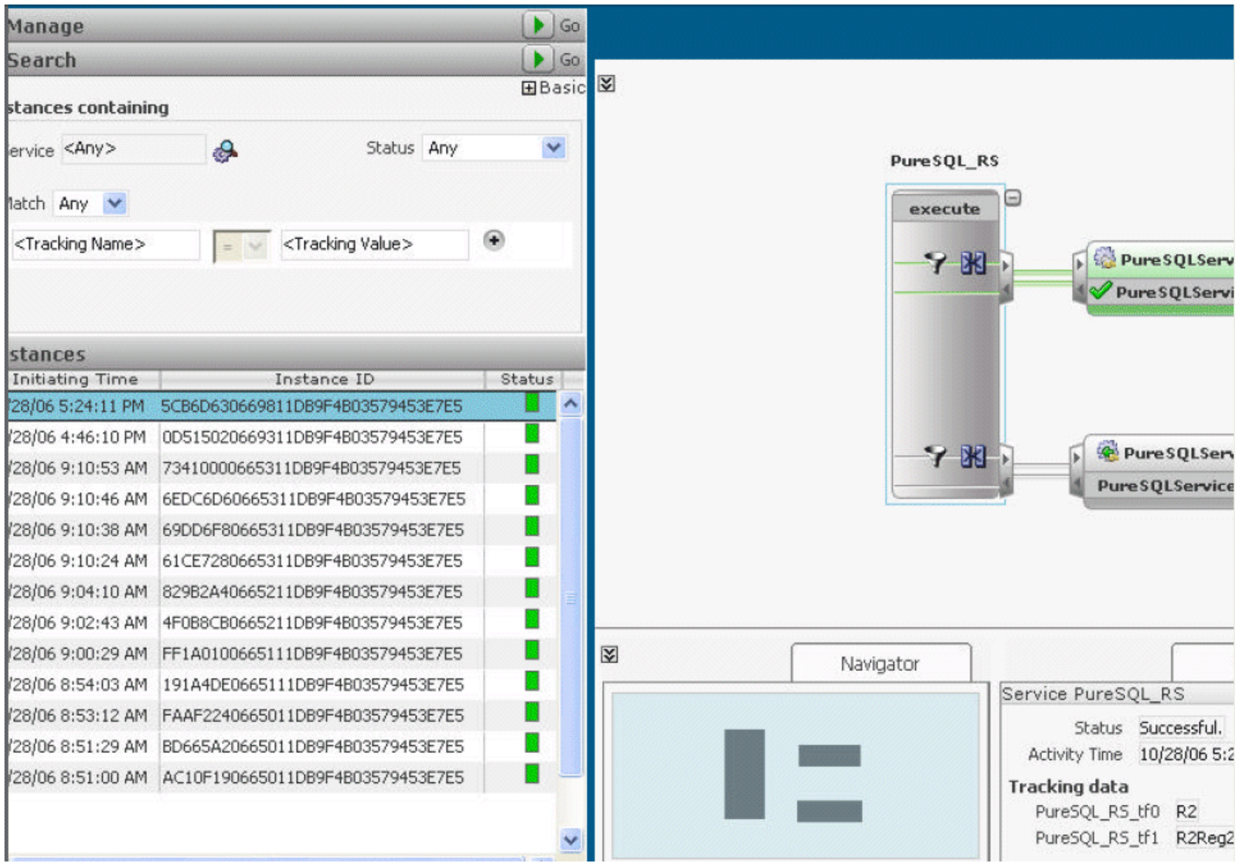
Service PureSQL_RS

Status	Successful.
Activity Time	10/28/06 4:

Tracking data

PureSQL_RS_tf0	R1
PureSQL_RS_tf1	R1reg1

17. Repeat steps 11 to 16 and enter `R2` in the field region in step 12.



Manage Go

Search Go

⊞Basic

stances containing

ervice <Any> Status Any

latch Any

<Tracking Name> = <Tracking Value>

stances

Initiating Time	Instance ID	Status
/28/06 5:24:11 PM	5CB6D630669811DB9F4B03579453E7E5	🟩
/28/06 4:46:10 PM	0D515020669311DB9F4B03579453E7E5	🟩
/28/06 9:10:53 AM	73410000665311DB9F4B03579453E7E5	🟩
/28/06 9:10:46 AM	6EDC6D60665311DB9F4B03579453E7E5	🟩
/28/06 9:10:38 AM	69DD6F80665311DB9F4B03579453E7E5	🟩
/28/06 9:10:24 AM	61CE7280665311DB9F4B03579453E7E5	🟩
/28/06 9:04:10 AM	829B2A40665211DB9F4B03579453E7E5	🟩
/28/06 9:02:43 AM	4F0B8CB0665211DB9F4B03579453E7E5	🟩
/28/06 9:00:29 AM	FF1A0100665111DB9F4B03579453E7E5	🟩
/28/06 8:54:03 AM	191A4DE0665111DB9F4B03579453E7E5	🟩
/28/06 8:53:12 AM	FAAF2240665011DB9F4B03579453E7E5	🟩
/28/06 8:51:29 AM	BD665A20665011DB9F4B03579453E7E5	🟩
/28/06 8:51:00 AM	AC10F190665011DB9F4B03579453E7E5	🟩

Service PureSQL_RS

Status Successful.

Activity Time 10/28/06 5:2

Tracking data

PureSQL_RS_tf0	R2
PureSQL_RS_tf1	R2Reg2

The results displayed in the tracking fields of Oracle ESB Control reflect the content of the table **TEST1** owned by user **Kabel2**.

3 Oracle Technology Adapters

This chapter provides answers to frequently asked questions about adapters.

This chapter contains the following topics:

- Release 10.1.3.3 Enhancements
- Installation Issues
- AQ Adapter
- Database Adapter
- File Adapter
- FTP Adapter
- JMS Adapter
- Adapter Framework

Release 10.1.3.3 Enhancements

See the following documentation that describes 10.1.3.3 enhancements.

- New features technical note

  `http://www.oracle.com/technology/products/ias/bpel/pdf/10133technotes.pdf`

- Patch set notes

  1. Go to Oracle MetaLink.

     `https://metalink.oracle.com/`

  2. Log into Oracle MetaLink.
  3. Search for patch number **6148874**.
  4. Click **View Readme**.

- Patch set notes addendum

  `http://webiv.oraclecorp.com/cgi-bin/webiv/do.pl/Get?WwwID=note:435108.1`

Installation Issues

This section provides answers to frequently asked questions about installation.

Adapters are Not Getting Installed

Problem

You installed Oracle SOA Suite 10.1.3.1 on Redhat Linux using the silent installation method. The installation procedure did not run the configuration assistants. Therefore, you manually ran the configuration commands. Everything appeared to run without errors. However, none of the technology adapters (file, FTP, and so on) were installed.

Solution

Check your `hosts` file for any IPv6 entries, comment them out, and retry the installation.

AQ Adapter

This section provides answers to frequently asked questions about the AQ adapter.

Can I Deploy and Run an AQ Adapter Process Using a Different Schema from the Design Time Schema?

The AQ adapter option of the Adapter Configuration Wizard generates some artifacts associated with the schema name. Releases prior to 10.1.3.3 required that both the design time and runtime schema names be the same for the process to function correctly. With release 10.1.3.3, this limitation has been removed.

- For pre-existing (10.1.3.1) processes

  Ensure that you remove the `DatabaseSchema` attribute, if it exists, from the partner link WSDL before deploying the process.

Table 3–1   DatabaseSchema Attribute Removal

Incorrect	Correct
`<jca:operation` `ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeue` `ActivationSpec"` **`DatabaseSchema="`*`<DESIGN-SCHEMA-NAME>`*`"`** `QueueName="<QUEUE-NAME>"` `OpaqueSchema="false" >` `</jca:operation>`	`<jca:operation` `ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeue` `ActivationSpec"` `QueueName="<QUEUE-NAME>"` `OpaqueSchema="false" >` `</jca:operation>`

- New (10.1.3.3) processes can be generated by selecting the **Default Schema** option in the Adapter Configuration Wizard instead of a particular schema name during design time.

Database Adapter

This section provides answers to frequently asked questions about the database adapter.

This section contains the following topics:

- XA Configuration - Two Database Adapter Invokes Committing or Rolling Back as a Unit

- Database Adapter Cannot Survive Database Restarts
- Explicit Schema Name References Causing Problems
- Missing Query By Example Support in 10.1.3
- Foreign Key and Primary Key Constraints with 1-1 & M-M Master-Detail Scenarios
- Printing Thread on Log Messages
- Database Adapter Tuning Documentation
- Troubleshooting
- DB2 Issue: Journaling Disabled
- Iterating through the Rows Returned by the Database Adapter

XA Configuration - Two Database Adapter Invokes Committing or Rolling Back as a Unit

The following is required to make two invokes commit or roll back as a unit:

- Both the database adapter invokes must be configured to participate in global transactions
- Both invokes must participate in the same global transaction
- The failure of either invoke must cause the global transaction to roll back

Configuring the Database Adapter for Global Transaction Participation

In the deployment descriptor (`oc4j-ra.xml`), you must set `xADataSourceName`. The matching data-source entry (in `data-sources.xml`) must be configured for transaction participation, that is:

```
<managed-data-source name="BPELSamplesDataSource" connection-pool-name="BPEL_POOL"
 jndi-name="jdbc/BPELSamplesDataSource" tx-level="global" />
```

The `tx-level="global"` property can be omitted because `"global"` is the default value. However, if the value is `"local"`, it is incorrectly configured.

True XA: Two-Phase (XA) Versus One-Phase (Emulated) Commit XA is a two-phase commit protocol, which is more robust than a one-phase commit or emulated protocol. The difference is that with a one-phase protocol, you may very rarely still see message loss or other rollback or commit inconsistency on the order of one per one thousand. A true XA configuration setting in `data-sources.xml` is shown below:

```
<connection-pool name='dbSample_CONNECTION_POOL'
inactivity-timeout='60'
validate-connection='false'
num-cached-statements='5'
time-to-live-timeout='600'>

<connection-factory factory-class='oracle.jdbc.xa.client.OracleXADataSource'
 user='scott'
password='tiger'
url="..."
</connection-factory>
</connection-pool>

<managed-data-source name='DBSampleNonEmulatedDS'
 connection-pool-name='dbSample_CONNECTION_POOL'
jndi-name='jdbc/dbSample' tx-level="global" />
```

Note the setting of `connection-factory-class` to `OracleXADataSource`.

Both Invokes in the Same Global Transaction

Once the invokes participate in global transactions, they must participate in the same global transaction in order to commit or roll back as a unit. In BPEL, this requires understanding the transaction's boundaries: at what points a checkpoint to write to the dehydration store commits the current global transaction and starts a new one. The transaction boundaries in a BPEL process are as follows:

- Before a receive activity (but *not* the initial one)
- Before a wait activity (otherwise, Oracle BPEL Server can run into a transaction timeout)
- Before an onMessage or pick (extended onMessage) activity
- When invoking a synchronous child BPEL process, unless the transaction `participate` property is set on the partner link. Otherwise, the parent process is broken into two transactions and the child process executes in its own transaction.

Failure Must Cause a Rollback

Even if both invokes participate in the same global transaction, the failure of either invoke may not cause the global transaction to roll back. The only cases where a failure can actually cause a global rollback are as follows:

- A database adapter operation that inserts or updates multiple tables as part of one invoke fails after having succeeded in some writes, but not others. In this case, the adapter marks the global transaction as *rollback only*, because the invoke operation was not atomic and a commit can cause data corruption.
- The global transaction `participate` property is set in `bpel.xml` so that when Oracle BPEL Process Manager catches a fault from the database adapter invoke, it allows the fault to propagate up and cause a global rollback.
- The invoke retries multiple times in a scenario in which the database is down, until the global transaction times out and is rolled back.
- An explicit `bpelx:rollback` fault is thrown from within the BPEL process. `GetActiveUnitOfWork="true"` in the WSDL file. `GetActiveUnitOfWork` is an advanced WSDL property you can set on any `DBInteractionSpec`. This property causes the invoke to register itself with the two-phase commit callbacks; all writes to the database are performed as part of the two-phase commit. By setting this property on any failure, the transaction is automatically rolled back. This is because there is no way to handle a fault at this late stage. The global transaction `participate` property is not required to be set. Likewise, the same underlying TopLink session is used for both invokes. This means that if you merge the same object twice, it is only inserted or updated once. All merge invokes in which `GetActiveUnitOfWork="true"` are cumulative.

Related Questions

This section describes related XA configuration questions.

This section contains the following topics:

- Can I Guarantee that Both Invokes Use the Same SQL Connection?
- Does a Scope Form a Transaction Boundary?

- Why Does No Instance Appear in Oracle BPEL Control Upon Failure After I Configured the Process for XA?
- How Do You Configure the Transaction participate Property?
- How Do You Throw a Rollback Fault in BPEL?
- How Do I Learn More about XA?
- How Do I Make a Database Adapter Receive (Polling) and Invoke Part of the Same XA Transaction?

Can I Guarantee that Both Invokes Use the Same SQL Connection? You cannot guarantee this behavior, but can typically expect it. If you notice this is not the case, and you are using an `XADataSource`, you can set the property `GetActiveUnitOfWork="true"` on all participating invokes.

Does a Scope Form a Transaction Boundary? No, a scope does not form a transaction boundary. XA commit and roll back actions and compensation are orthogonal to each other, and also slightly mutually exclusive. Choose one or the other.

Why Does No Instance Appear in Oracle BPEL Control Upon Failure After I Configured the Process for XA? This is expected behavior. To test whether the global transaction rolled back or not, see if the instance appears in Oracle BPEL Control. If it appears as faulted, the transaction was committed anyway. Therefore, the auditing to the dehydration store can proceed.

How Do You Configure the Transaction participate Property? On a child process invoke, set the transaction `participate` property on the partner link:

```
<partnerLinkBinding name="...">
<property
 name="transaction">participate</property>
</partnerLinkBinding>
```

To set the global property so that a failed invoke causes the entire transaction to roll back, perform the following:

```
</partnerLinkBindings>
<configurations>
<property name="transaction">participate</property>
</configurations> </BPELProcess>
```

How Do You Throw a Rollback Fault in BPEL?

```
<throw name="Throw" faultName="bpelx:rollback"/>
```

How Do I Learn More about XA? See the following database adapter samples:

- `*SOA_ORACLE_HOME*/bpel/samples/tutorials/122.DBAdapter/advanced/dmlInvoke/XAInsert`
- `*SOA_ORACLE_HOME*/bpel/samples/tutorials/122.DBAdapter/InsertWithCatch`

How Do I Make a Database Adapter Receive (Polling) and Invoke Part of the Same XA Transaction? You must configure database adapter polling to synchronously initiate the BPEL process. See the `*SOA_ORACLE_HOME*/bpel/samples/tutorials/122.DBAdapter/advanced/endToEnd/DirectSqlPerformance/README.txt` file for step-by-step instructions.

Database Adapter Cannot Survive Database Restarts

The database adapter should survive a database restart scenario, and resume without any message loss. If it does not, the following examples provide some possibilities of what may be wrong. The following examples assume you installed the 10.1.3.3 patch set.

- Inbound Polling Gets Stuck and Cannot Close Connection Exceptions
- Not Automatically Retrying Invokes After Remote Faults
- Exception Misclassification
- Missing Instances or Instances Accessible from the Perform Manual Recovery Option

Inbound Polling Gets Stuck and Cannot Close Connection Exceptions

Symptoms

After a database restart when using XA with RAC, all database adapter polling attempts fail with `SQLException: cannot close connection type` exceptions. A bad or corrupted connection can sometimes not close (that is, trying to close the connection fails and throws an exception).

Prognosis

A SQL connection that cannot be closed is eventually cleaned up by the data source because of the `inactivity-timeout` configuration property, which defaults to `60` seconds. However, a typical polling interval for the database adapter may be `10` seconds, which means the `inactivity-timeout` never occurs. This causes the bad connection to stay indefinitely.

Treatment

One solution may be to try the true XA configuration described in "True XA: Two-Phase (XA) Versus One-Phase (Emulated) Commit" on page 3-3. Another solution may be to set an explicit `time-to-live-timeout='60'` value in the `data-sources.xml` file. This ensures that a corrupted SQL connection is eventually recycled.

If you are using RAC, the real problem is likely that you configured for fast connection failover. See note 372456.1 for details.

```
<property name="connectionCachingEnabled" value="true"/>
<property name="fastConnectionFailoverEnabled" value="true" />
 </connection-factory>
```

Not Automatically Retrying Invokes After Remote Faults

In 10.1.3.3, you now must configure a fault policy to automatically retry remote exceptions indefinitely.

The following excerpt is from `*SOA_ORACLE_HOME*/bpel/samples/tutorials/122.DBAdapter/InsertWithCatch/README.txt`:

Remote faults are thrown if the adapter assumes that the exception is connection related and therefore can be retried. Rather than handle these exceptions, configure a general fault policy to automatically retry a given number of times. The fault policy is new in 10.1.3.3, and the partner link retries option of 10.1.3.1 is no longer supported.

To configure a fault policy, go to `bpel/domains/default/config` and look at the following files:

- `fault-bindings.xml`
- `fault-policies/DefaultPolicy.xml`

The default `fault-bindings.xml` file contents are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicyBindings version="2.0.1"
 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<!-- Enabling this will cause all processes in this domain to use this fault
 policy     <process faultPolicy="DefaultPolicy"/> -->
<!-- DefaultPolicy is defined in ./fault-policies/DefaultPolicy.xml -->
 <partnerLink faultPolicy="DefaultPolicy">
<!-- Enabling this will cause all invoke faults at partner link name of
 "creditRatingService" to use fault policy with id id = DefaultPolicy
 <name>creditRatingService</name>         -->
<!-- all invoke faults at partner link below port type use fault policy
with id = DefaultPolicy
The following entry covers the samples/tutorials/122.DBAdapter/InsertWithCatch
 sample. -->
<portType xmlns:db="http://xmlns.oracle.com/pcbpel/adapter/db/insert/">db:insert_
plt</portType>
</partnerLink> </faultPolicyBindings>
```

You must register each partner link to be handled by specifying the `namespace` and `portType`. Be absolutely sure your name ends in `_plt`. Note that it is easy to accidentally specify `_ptt`.

The `DefaultPolicy.xml` file contents are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicy version="2.0.1"
 id="DefaultPolicy" xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<!-- This section describes fault conditions. Build more conditions with
faultName, test and action -->
<Conditions>
<!-- Fault if wsdlRuntimeLocation is not reachable -->
<faultName xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
name="bpelx:remoteFault">
<condition>
<action ref="ora-retry"/>
</condition>
</faultName>
<!- Fault if location port is not reachable->
<faultName xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
 name="bpelx:bindingFault">
<condition>
<action ref="ora-rethrow-fault"/>
</condition>
</faultName>
</Conditions>
<Actions>
<!-- This action will attempt 8 retries at increasing intervals of 2, 4, 8, 16,
32, 64, 128, and 256 seconds. -->
<Action id="ora-retry">
```

```
<retry>
<retryCount>8</retryCount>
<retryInterval>2</retryInterval>
<exponentialBackoff/>
</retry>
</Action>
<!- This is an action will cause a replay scope fault->
<Action id="ora-replay-scope">
<replayScope/>
</Action>
<!- This is an action will bubble up the fault->
<Action id="ora-rethrow-fault">
<rethrowFault/>
</Action>
<!- This is an action will mark the work item to be "pending recovery from
console"->
<Action id="ora-human-intervention">
<humanIntervention/>
</Action>
<!- This action will cause the instance to terminate->
<Action id="ora-terminate">
<abort/>
</Action>
</Actions>
</faultPolicy>
```

This fault policy handles both remote faults (can be retried) and binding faults (cannot be retried). In the latter case, the fault is simply rethrown.

See *Oracle SOA Suite New Features* for details about the fault policy:

http://www.oracle.com/technology/products/ias/bpel/pdf/10133technotes.pdf

Exception Misclassification

Automatic retries can only occur if a remote fault is thrown. If a binding fault is thrown, you may have a problem with exception misclassification. Only perform the following steps if you can see that the wrong type of exception is being thrown. You can see this by auditing the instance in Oracle BPEL Control.

1. Identify the error code of the `SQLException` being thrown. For example, assume it is `17002` (the basic `TNS listener not available` exception).
2. Add the property `nonRetriableSQLErrorCodes` to `oc4j-ra.xml` (you have to first declare the property in `ra.xml`). The value is a space or comma-separated list that identifies what is certainly a binding fault or a remote fault. For example:
   - `"1"` says `1` is certainly a binding fault
   - `"+1 -17002"` says `1` is certainly a binding fault
   - `17002` is certainly a remote fault

   Therefore, if `17002` is accidentally classified as a binding fault, `"-17002"` overrides that and ensures that it is always a remote fault.

Missing Instances or Instances Accessible from the Perform Manual Recovery Option

There is no such concept as missing instances. These instances are accessible from the **Perform Manual Recovery** option of Oracle BPEL Control. In 10.1.3.3, the only time an instance can appear here when there is a remote fault is when the database has been down for a long time, and the global transaction times out. An invoke can only be retried so many times until the enveloping transaction times out.

In this case, you can try configuring manual recovery to automatically retry an *x* number of times without pausing between retries. This prevents you from having to manually perform this task. There is an advanced BPEL property named `nonFatalConnectionMaxRetry` which defaults to `2`. This can be set from the Oracle BPEL Control **Configuration** tab.

If you have missing instances on the outbound database and the instances are shown as completed in Oracle BPEL Control, you likely have not configured for XA. See "XA Configuration - Two Database Adapter Invokes Committing or Rolling Back as a Unit" on page 3-3 for instructions.

Explicit Schema Name References Causing Problems

When you import your tables or stored procedures in Oracle JDeveloper, ensure that you import them from the default schema. This means that if the objects are in schema `SCOTT`, you connect in Oracle JDeveloper as user `SCOTT`. This way, no schema qualifications should appear in any of the generated metadata. Hard-coded schema names cause problems. When you later switch your process to a QA or production environment and log in, for example, as `QA`, it automatically looks for the same objects in schema `QA`.

If you explicitly want the schema qualifications in there (very rare), this means that at runtime you intend to log in as one user, but access objects from a schema other than the one in which you logged in. In this case, you need explicit qualifications. If at the same time this explicit schema reference changes from development to QA to production environments, you have problems.

Try to import the objects without schema references in Oracle JDeveloper. At runtime for tables and views, you can set the `oc4j-ra.xml` property `tableQualifer` to qualify all unqualified names. For stored procedures, you can set the WSDL property `SchemaName`.

There is a bug in JPublisher with logging in as one user and importing a stored procedure belonging to another user. Again, try to avoid this situation.

There are also ways to remove the hard-coded schema references from the tables and view generated metadata after the fact. However, the best advice again is to avoid doing this in the first place.

For tables and views once the schema name is already set and you are trying to fix the problem, the easiest solution is to edit the *`service_name`*`_toplink_mappings.xml` file outside of Oracle JDeveloper and do a search and replace of "*HARD_CODED_SCHEMA_NAME*." with `""` (that is, `SCOTT.TABLE1 -> TABLE1`).

Missing Query By Example Support in 10.1.3

Query by example support is missing from the Adapter Configuration Wizard in 10.1.3. You can manually add support by editing the WSDL file.

1. Create a normal service and choose an outbound `select` statement.
2. Open *`service_name`*`.wsdl`.
3. Add the following parts, assuming for this example that the table is named `Movies`:
   a. Add the following to the top level:

```
<message name="Movies_msg">
<part name="Movies" element="top:Movies"/>
 </message>
```

b. Add the following to the `portType` element:

```
<operation name="queryByExample">
<input message="tns:Movies_msg"/>
<output message="tns:MoviesCollection_msg"/>
</operation>
</portType>
```

c. Add the following to the binding element:

```
<operation name="queryByExample">
<jca:operation
 InteractionSpec="oracle.tip.adapter.db.DBReadInteractionSpec"
DescriptorName="<serviceName>.Movies"
IsQueryByExample="true"
MappingsMetaDataURL="Service_Name_toplink_mappings.xml" />
<input/>
</operation>
```

d. Add the operation element right after a matching `jca:operation` that was created for the plain `select` statement. Reuse the `DescriptorName` and `MappingsMetaDataURL` of the existing select for the `queryByExample` operation.

e. Edit the *`service_name`*`_table.xsd`. You should see:

```
<xs:element name="MoviesCollection" type="MoviesCollection"/>
```

f. Ensure that this line exists or add it immediately afterwards:

```
<xs:element name="Movies" type="Movies"/>
```

Foreign Key and Primary Key Constraints with 1-1 & M-M Master-Detail Scenarios

Normally in master-detail scenarios, it is assumed that the child records are exclusively owned by one parent (a `Dept` and its `Emps`, a purchase order and its line items, and so on). However, there are schemas where the children are referenced from multiple parents. An example is `Employees` with a many-to-many (1-M + 1-1) to `Projects`.

Moving data from one database to another is like transplanting a forest. The trees may be widely spaced so each can be individually uprooted and easily moved. The default database adapter configuration is tuned to excel in these scenarios. If the trees are spaced so that the roots of trees intertwine and weave together to form a tight mesh, the task becomes much harder, but not impossible. This section addresses this latter scenario, which requires a little more caution and some nondefault tuning.

This section contains the following topics:

- The Delete Polling Strategy Cannot Effectively Be Used
- Primary Key Constraints With Merge
- Configuring Private Ownership

The Delete Polling Strategy Cannot Effectively Be Used

Assume employees Bob and Jane both work on the same project. Bob is processed first by the delete polling strategy, and his project is deleted afterwards. All his details are assumed to be exclusively owned. However, deleting his project causes a constraint violation, because Jane has not yet been processed. Therefore, deleting the project causes her reference to become orphaned. Setting `MaxTransactionSize="unlimited"` to process Bob and Jane in the same transaction is not sufficient. They are still processed one at a time.

You can configure the delete polling strategy to delete only the master (employee) or to delete only the master and privately-owned parts (see additional details below).

You can delete only the master by setting `DeleteDetailRows="false"` on the `ActivationSpec`. However, it is best to use a *logical* delete polling strategy. The delete polling strategy does not make sense when individual records share the same detail rows. This is because a record cannot be processed individually.

You can also experiment with setting `UseBatchDestroy="true"`. This setting performs a recursive delete all.

Primary Key Constraints With Merge

Assuming you use the logical delete polling strategy, you may encounter a new problem on the merge invoke when trying to insert into the target database.

Assume Bob and Jane share the same project. With the property `MaxRaiseSize="1"`, they are both raised to Oracle BPEL Process Manager or Oracle ESB at the same time, and are simultaneously merged into the target database. A `select` statement determines that Bob and Jane both must be inserted. This is fine, but as their project does not exist either, both try to insert the (same) project at the same time.

To avoid this, you may need to use in-order delivery and set a large `MaxRaiseSize` value. However, that alone is not enough. Unless you tell TopLink that a master's parts are not exclusively owned, TopLink cheats and assumes that because Bob does not exist, his projects cannot exist either. This option boosts performance, but is incorrect in this scenario. You must configure private ownership.

Configuring Private Ownership

For each relationship in TopLink, you can indicate whether or not it is privately owned. Private ownership means a detail record is exclusively referenced and nobody else references that row.

Theoretically, all one-to-many relationships are privately owned because the detail row has a foreign key that can only point to a single record. In the case of a 1-1, however, the foreign key is on the parent. Therefore, multiple parents can all point to the same child. Therefore, the ideal default configuration may be to make all 1-M mappings privately owned and all 1-1 mappings nonprivately owned. That is the most correct, but this carries a noticeable performance cost. Indicating to TopLink and the database adapter that rows are exclusively owned enables them to be processed far more efficiently.

As an example, if a purchase order does not exist, the database adapter never needs to do an existence check on any of its line items. The adapter knows already it must insert them as they cannot exist without their parent also existing.

To configure this behavior:

1. Open the `toplink-mappings.xml` file outside of Oracle JDeveloper, and look for the following:

   ```
   <opm:private-owned>true</opm:private-owned>
   ```

2. Change this setting to `false` for known 1-1 mappings that are causing the problem.

Printing Thread on Log Messages

Logging sometimes provides excessive information that makes reviewing the results difficult. Likewise, with multiple concurrent threads, it is difficult to determine who is doing what. To help you better understand the information, set `logTopLinkAll="true"` in `oc4j-ra.xml`. Even more logging is produced at the `DEBUG` level, but all TopLink-based logging includes thread and session information. Therefore, you can focus on messages from a single thread at a time.

Ensure that you also uncomment the following information in `ra.xml` before restarting:

```
<config-property>
<config-property-name>logTopLinkAll</config-property-name>
 <config-property-type>java.lang.Boolean</config-property-type>
 <config-property-value>false</config-property-value>
</config-property>
```

A better way to do this in 10.1.3.3 for all Oracle BPEL Process Manager logging is to add `"%t"` in your conversion pattern in `log4j-config.xml`.

Database Adapter Tuning Documentation

For tuning database adapter performance, see the performance section in the Oracle Application Server Adapter for Databases chapter of *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide*. The performance section refers you to the `README` file of the sample in *`SOA_ORACLE_HOME`*`/bpel/samples/tutorials/122.DBAdapter/advanced/endToEnd/MultiTablesPerformance`.

Troubleshooting

If the database adapter is not working, some simple steps to identify the problem are as follows:

- Enable logging (see "Adapter Framework" on page 3-26).
- See the Troubleshooting and Workarounds appendix of *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide*.

Find the Root Exception

If the database adapter is not working, the most important thing to identify is the exception. The easiest way to identify this is to audit the instance in Oracle BPEL Control. Click **Flow**, and then the failed invoke.

If you do not see a faulted instance, it may be because:

- You are configured for XA, so faulted instances do not appear (10.1.3.*x*). Try configuring `dataSourceName` with a `tx-level="local"` pool instead of just for debugging.
- The problem is on the inbound polling side. Either the process failed to initialize, or no records were successfully raised to Oracle BPEL Process Manager. Debugging inbound problems is harder because there is no Oracle BPEL Control to view.

If you cannot see the exception from Oracle BPEL Control, check the logs. In 10.1.3.*x*, the logs are located in `*SOA_ORACLE_HOME*/bpel/domains/default/log/domain.log`.

What Exception Do I Look For? Look for the root exception. The ideal root exception is a `SQLException` on the database. The database adapter catches this exception and wraps it in a `DBResourceException`. The adapter framework catches that and wraps it in its own exception. Finally, BPEL catches the adapter framework exception and throws a BPEL exception. The BPEL exception is caused by the adapter framework exception, which is caused by the database adapter exception, which is caused by the `SQLException`.What adds difficulty is that each exception may show its own stack trace. This is because many times, a top-level stack trace is received that reveals little more than `BPEL exec failed`. Finding the `SQLException` and the `ORA-` code is ideal. If there is no `SQLException`, look for `DBResourceException`. These exceptions typically have messages explaining what the problem may be and what actions to take.

With a `SQLException`, you can use Google to find out more about the error code. You can then cross-reference the `SQLException` with the Troubleshooting and Workarounds appendix of *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide* or the information in this chapter. Some `SQLExceptions` require a more in-depth explanation or interpretation.

If you find the root exception and still cannot identify the problem, see "Finding Out What Is Occurring".

Finding Out What Is Occurring

The following are the two most useful pieces of information to determine what is occurring:

- Input XML messages to an outbound invoke

  If you have a faulted instance in Oracle BPEL Control, you can see the input XML for that invoke, generally by debugging the instance. The following are common problems:

  - A transform that creates an invalid input variable. Try turning on schema validation from Oracle BPEL Control.
  - A transform that builds a separate variable but the invoke's input variable is passed in empty
  - A transform that does not set all fields, sets some fields to invalid values (for example, a `dateTime` field to a non-ISO `dateTime` string), or fails to use the `for-each` feature causing only the first element of a collection to get copied.

- Database adapter logs

  You must enable database adapter debug logging (see "Adapter Framework" on page 3-26 for instructions). The most important information here is the SQL executed by the database adapter. You should see the SQL string and the bind variables.

  You can also look for any warnings or other details that may have appeared.

DB2 Issue: Journaling Disabled

To explicitly begin and commit transactions in DB2, you must enable journaling. Ask your database administrator to enable it.

However, if enabling journaling is not an option:

- In `oc4j-ra.xml`, set `xADataSourceName` to the JNDI name of a `tx-level="local"` data source (and ensure that the database connection from `data-sources.xml` only supports `local`).
- The database adapter assumes the connection is managed, and therefore does not call a begin or commit. Likewise, the local data source assumes it is managed by the database adapter and does not call an explicit begin or commit either. Therefore, all writes occur in automatic commit mode.
- The obvious disclaimer to the `xADataSourceName` + `tx-level="local"` configuration is that compensation can be hard for multistatement invokes (for example, an insert that writes the number rows > `maxBatchWritingSize` (default `50`) or master-detail upserts), and you lose XA support.

Iterating through the Rows Returned by the Database Adapter

This section provides an example of how to iterate though data returned by the database adapter.

Assume you use the database adapter to perform a query that returns a collection of elements you want to treat in a certain way; for example, depending on the data contained in the elements of the collection.

Assume you have the following view to query:

```
SQL> desc approvers_view
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 APPROVER_ID                               NOT NULL VARCHAR2(128)
 APPROVER_NAME                             NOT NULL VARCHAR2(128)
 TECHNOLOGY                                NOT NULL VARCHAR2(128)
```

Query the `technology` column of this view:

```
SQL> col approver_id for a30
SQL> col approver_name for a20
SQL> col technology for a20
SQL> select * from approvers_view where technology = 'Security';

APPROVER_ID                    APPROVER_NAME        TECHNOLOGY
------------------------------ -------------------- --------------------
amit.jasuja@oracle.com         Amit Jasuja          Security
peter.farkas@oracle.com        Pete Farkas          Security
```

Assume you have two people responsible for this technology stack, each of which you want to send a separate e-mail.

Problem

If you use the database adapter, you must eventually deal with a document (or document fragment) similar to the following:

```
<n:ApproversViewCollection
 xmlns:n="http://xmlns.oracle.com/pcbpel/adapter/db/top/RequestTrackerTest"
                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ApproversView>
    <approverId>amit.jasuja@oracle.com</approverId>
    <approverName>Amit Jasuja</approverName>
    <technology>Security</technology>
  </ApproversView>
  <ApproversView>
    <approverId>peter.farkas@oracle.com</approverId>
    <approverName>Pete Farkas</approverName>
    <technology>Security</technology>
  </ApproversView>
</n:ApproversViewCollection>
```

Solution

You can use the BPEL while activity. The key is to find the right formulation for the while condition. The XPath method for accessing the e-mail address in which you are interested is as follows:

```
/ApproversViewCollection/ApproversView[index]/approverId
```

where `index` is the index of the row in the array returned by the database adapter.

The `index` begins at `1`. Most importantly, when `index` exceeds the size of the array, the XPath expression returns an appropriate zero-length string. That is all that is required. You want to loop while the XPath expression returns a string longer than zero.

To achieve this, perform the following tasks:

- Create a variable `xsd:int`, which you use as the index.
- Initialize this variable to `1` before the while loop.
- Define the loop condition as mentioned above, involving the index variable.
- Refer to the row in the loop using the index variable.
- Increment the index variable in the loop.

The key is the expression of the `while` condition:

```
 string-length(bpws:getVariableData('GetApprovers_OutputVariable',

'ApproversViewCollection')/ApproversView[number
                                     (bpws:getVariableData('index'))]/approverId)
                                     > 0
```

This is not what the XPath Expression Builder initially suggests.

You can also use the `count-node` function as an option.

Referring to the data of interest can be done using similar syntax:

```
bpws:getVariableData('GetApprovers_OutputVariable',
                     'ApproversViewCollection')/ApproversView[number
                      (bpws:getVariableData('index'))]/approverId
```

Incrementing the index is easy:

```
number(bpws:getVariableData('index') + 1)
```

Syntax closer to what the XPath Expression Builder suggests by default can also be used:

```
 bpws:getVariableData('GetApprovers_OutputVariable',
                      'ApproversViewCollection',
                       '/ns4:ApproversViewCollection/ApproversView[number
                       (bpws:getVariableData(&quot;index&quot;))]/approverId')
```

Note: You must escape the quotes in the reference to the index variable because you have two embedded expressions.

File Adapter

This section provides answers to frequently asked questions about the file adapter.

This section contains the following topics:

- Is There Still a 7 MB File Size Limit for the File Adapter?
- Can the File Adapter Be Used to Access Windows Network Folders?
- How Do I Append to an Existing File with the File Adapter?
- How Do I Read the File Name of the Generated File from a Write Operation in BPEL?
- How Do I Control the Size of a Rejected Message in the File Adapter?
- Does the File Adapter Support the Debatching of Large XML files?
- Messages Are Lost When Using Time-Pattern in the Outbound Partner Link File Name Because Messages Created with the Same Timestamp Overwrite One Another
- Can I Be Notified When the File Adapter is Done Debatching a File?
- How Do I Set Up the File Adapter in a High Availability Environment?
- How Do I Configure the Number of Threads for the File Adapter?
- How Do I Guarantee the Ordering of Messages Processed by the File Adapter and BPEL

Is There Still a 7 MB File Size Limit for the File Adapter?

The file adapter can process very large (multiple gigabyte) files using the debatching feature. If the inbound file has repeatable structures, then this feature can divide the inbound file into manageable chunks and process them separately.

In previous releases (10.1.3.1 and earlier), the debatching feature worked with native files only. With 10.1.3.3, this feature has been extended to XML files as well. This means that, if you have a large XML file with repeating nodes, you can process them in manageable chunks.

Additionally with 10.1.3.3, the attachment feature is supported. With this feature, you can copy and move large files opaquely from one folder to another. You can also use the streaming XPath functions on the attachment to perform transformations. See *Oracle SOA Suite New Features* for details:

`http://www.oracle.com/technology/products/ias/bpel/pdf/10133technotes.pdf`

Can the File Adapter Be Used to Access Windows Network Folders?

Yes, you can either use UNC paths or you can use mapped drives. However, there may be security issues, particularly when the OC4J instance runs as a Windows service. In such cases, the service runs as the `LocalSystem` user. This user may not have access to the network folder.

One workaround is to run the OC4J service as a specific user (the user with access to the network folder). Another workaround is to use the FTP adapter.

How Do I Append to an Existing File with the File Adapter?

The file adapter allows you to configure outbound interactions that can append to existing files. To append to a file, set `Append="true"` in the interaction specification for the file adapter.

```
<jca:operation FileType="ascii"
PhysicalDirectory="/home/adapter/out"
FileNamingConvention="MyOutputFile.txt"
NumberMessages="1"
Append="true" >
```

The file name can either be specified in the WSDL (as shown in this example) or can come from the header.

How Do I Read the File Name of the Generated File from a Write Operation in BPEL?

You must manually edit the process WSDL file for the outbound file adapter to receive the file name after the interaction. Add an output message of type outbound header to the `portType` as shown below. Remember that the namespace prefix (`hdr`) is already defined in the WSDL.

```
<portType name="Write_ptt"> <operation name="Write"> <input
 message="tns:PurchaseOrder_msg"/> <output message="hdr:OutboundHeader_msg"/>
 </operation> </portType>
```

In the BPEL file, manually create a variable of type outbound header as shown below; the namespace may vary.

```
<variable name="Invoke_1_Write_OutputVariable" messageType="ns3:OutboundHeader_
msg"/>
```

Use this variable as the outbound variable in the invoke activity:

```
<invoke name="Invoke_1" partnerLink="FileOut" portType="ns2:Write_ptt"
 operation="Write" inputVariable="Invoke_1_Write_InputVariable"
 outputVariable="Invoke_1_Write_OutputVariable"/>
```

How Do I Control the Size of a Rejected Message in the File Adapter?

You can now control the size of rejected messages by specifying the following endpoint property for the inbound file adapter partner link. For example, if you want to reject `100` lines from the file because the actual file is too large, then specify the endpoint-property as follows:

```
oracle.tip.adapter.file.debatching.rejection.quantum="100"
```

The acceptable values for this property are: `0`, `EOF`, or any non-negative number. In the absence of this property, the entire message (or what is available of the message) is rejected.

Does the File Adapter Support the Debatching of Large XML files?

Yes. A new 10.1.3.3 feature allows you to debatch XML files. Setting up XML debatching requires you to download and configure the XML pull-parsing library (`StaX`).

1. Download the API (`jsr173_1.0_api.jar`) and RI (`jsr173_1.0_ri.jar`) from the following location:

   ```
   http://jcp.org/aboutJava/communityprocess/final/jsr173/index.html
   ```

   You must go to the section for "Reference Implementations" and select from the available reference implementations.

2. Copy the two jar files to *`SOA_ORACLE_HOME`*`\bpel\lib`.

3. Register both jar files in `server.xml` (available under *`SOA_ORACLE_HOME`*`\j2ee\`*`MID_TIER`*`\config`) under the `oracle.bpel.common` shared library.

   ```
   <shared-library name="oracle.bpel.common"
   version="10.1.3">
    <code-source
   path="C:\product\bpel\lib\jsr173_1.0_api.jar"/> <code-source
    path="C:\product\bpel\lib\jsr173_1.0_ri.jar"/>
   </shared-library>
   ```

See also the following link about XML debatching with the file adapter:

`http://oraintpc.blogspot.com/2007/08/xml-debatching-in-file-ftp-adapter.html`

Messages Are Lost When Using Time-Pattern in the Outbound Partner Link File Name Because Messages Created with the Same Timestamp Overwrite One Another

You can now mix file naming conventions (for example, you can specify `%yyMMddHHmmssSSz%__%SEQ%_OrderBookings.xml`) This ensures that the file names are unique.

Note: This works only for nonbatching cases (for example, setting `NumberMessages="1"` in the interaction spec).

Can I Be Notified When the File Adapter is Done Debatching a File?

The file adapter can notify a dedicated BPEL process about when a batch begins, ends, or fails. This BPEL process must implement the operations specified in the following file:

```
../orabpel/system/xmllib/jca/BatchManager.wsdl
```

In `bpel.xml` (in the file adapter activation agent), you must also define the following property:

```
<activationAgent ...>
    <property name="batchNotificationHandler">
        bpel://domain|process_name
    </property>
```

The BPEL process *`process_name`* receives the callback `onBatchReadComplete` when the last record of the batch has been sent to the BPEL process.

After this point, you can use the XPath function `ora:batchProcessCompleted(batchId, processId)` to determine when actual processing of the spawned instances has completed.

How Do I Set Up the File Adapter in a High Availability Environment?

The file and FTP adapters support the high availability feature for the active-passive topology. Perform the following steps to configure the adapter for this feature:

1. Create a shared folder on a highly available file system. This folder must have write permissions and must be accessible from all systems running the file and FTP adapters.
2. Open the `pc.properties` file available in the *`SOA_ORACLE_HOME`*`\bpel\system\service\config` directory on each node.
3. Set `oracle.tip.adapter.file.controldirpath` to the shared folder name. This is the shared folder that stores the control files for the adapter.
4. Restart the servers.

Note that in the case of Oracle ESB, you must rename *`SOA_ORACLE_HOME`*`\integration\esb\config\pc.properties.esb` to `pc.properties` and make the relevant changes in that file.

How Do I Configure the Number of Threads for the File Adapter?

- For Oracle BPEL Process Manager — edit the `pc.properties` file under *`SOA_ORACLE_HOME`*`\bpel\system\service\config`. Set the value for `oracle.tip.adapter.file.numProcessorThreads`. By default, it is set to `4`.
- For Oracle ESB — rename *`SOA_ORACLE_HOME`*`\integration\esb\config\pc.properties.esb` to `pc.properties`. Set the value for `oracle.tip.adapter.file.numProcessorThreads`.

For Oracle SOA Suite installations, the `pc.properties` file for Oracle BPEL Process Manager takes precedence over that of Oracle ESB. In such cases, the values set in *`SOA_ORACLE_HOME`*`\bpel\system\service\config\pc.properties` are sufficient.

How Do I Guarantee the Ordering of Messages Processed by the File Adapter and BPEL

Even if you publish messages from the file adapter in a certain order, it does not guarantee the order in which BPEL processes these messages. To maintain message ordering, you must do the following:

1. Set `oracle.tip.adapter.file.numProcessorThreads=1` (see "How Do I Configure the Number of Threads for the File Adapter?" on page 3-19).
2. Model the BPEL process as a synchronous file adapter BPEL process:

   The file adapter option of the Adapter Configuration Wizard typically creates only one-way WSDLs. Therefore, you must manually edit it. For example, you modify the generated WSDL so it becomes a request-response type WSDL with input and output messages. For example:

   1. Create an XML schema type for the (dummy) response (output) message.

      ```
      <types>
      <schema xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/fileService/">
      <import namespace="http://TargetNamespace.com/fileService"
       schemaLocation="FileSchema.xsd" />
      <element name="empty">
      <complexType/>
      ```

```
</element>
```

2. Define the WSDL message:

```
<message name="ignore_msg">   <part name="empty" element="file:empty"/>
 </message>
```

3. Add an output message to the file read operation:

```
<portType name="Read_ptt">
<operation name="Read">
<input message="tns:records_msg"/>
<output message="tns:ignore_msg"/>
</operation>
</portType>
```

4. Add an output element in the binding section:

```
<binding name="Read_binding" type="tns:Read_ptt">
<pc:inbound_binding />
<operation name="Read">
<jca:operation .../>
<input>
<jca:header message="hdr:InboundHeader_msg" part="inboundHeader"/>
 </input>
<output/>
</operation>
</binding>
```

   The WSDL is now correct.

5. Add a reply activity in the BPEL process:

```
<variables>
<variable name="ignore" messageType="ns1:ignore_msg"/>
<correlationSets>   <correlationSet name="dummy" properties="ns1:dummy"/>
 </correlationSets
<sequence name="main">
<receive partnerLink="FileReader" portType="ns1:Read_ptt"
 operation="Read"
variable="Receive_1_Read_InputVariable"
 createInstance="yes">
<correlations>
<correlation initiate="yes" set="dummy"/>
 </correlations>
  </receive>
  [...]  <!-- processing -->
  <invoke partnerLink="...."/>
  <invoke partnerLink="...."/>
  <reply partnerLink="FileReader" portType="ns1:Read_ptt" operation="Read"
 variable="ignore"/>
  [...]  <!-- optionally more processing -->
</sequence>
```

6. Note the use of a correlation set on the receive activity, which is needed to establish and determine the `FileReader` partner link conversation ID. The correlation property can be defined in the WSDL file for `FileReader`. For example:

```
<bpws:property name="dummy" type="xsd:string"/>
<bpws:propertyAlias propertyName="tns:dummy"
 messageType="tns:records_msg" part="..." query="..."/>
```

Using a synchronous flow guarantees the ordering between the receive and invoke right before the (first) reply. Note that the process can still continue after the reply, if necessary.

FTP Adapter

This section provides answers to frequently asked questions about the FTP adapter.

This section contains the following topics:

- Will All File Adapter Features Work with the FTP Adapter?
- Does the FTP Adapter Support the ACTIVE Mode of Connections?
- How Can I Set Up the FTP Adapter to Use Connection Pooling?
- Can I Have an FTP Adapter Partner Link for an Invoke Operation to Retry If Connection-Related Errors Exist?

Will All File Adapter Features Work with the FTP Adapter?

Yes, all the features described in "File Adapter" on page 3-16 work with the FTP adapter.

Does the FTP Adapter Support the ACTIVE Mode of Connections?

No, for security concerns the FTP adapter supports PASV (passive) only.

How Can I Set Up the FTP Adapter to Use Connection Pooling?

To set up connection pooling, you must modify `oc4j-ra.xml`. See below for an example. Note that `keepConnections` is set to `true` and the `connection-pool` setting has been modified.

```
<connector-factory location="eis/Ftp/FtpAdapter" connector-name="Ftp Adapter">
 <config-property name="host" value="stacc29.us.oracle.com"/> <config-property
 name="port" value="21"/>
. . .
. . .
 <config-property name="keepConnections" value="true"/>
 <connection-pooling use="private">
<property name="waitTimeout" value="300" />
 <property name="scheme" value="fixed_wait" />
<property name="maxConnections"
 value="100" />
<property name="minConnections" value="20" />
 </connection-pooling> <security-config use="none">
</security-config>
 </connector-factory>
</oc4j-connector-factories>
```

Can I Have an FTP Adapter Partner Link for an Invoke Operation to Retry If Connection-Related Errors Exist?

You can configure the `retryInterval` and `retryMaxCount` properties to set up retries for outbound FTP adapter partner links.

JMS Adapter

This section provides answers to frequently asked questions about the JMS adapter.

This section contains the following topics:

- How Do I Create a JMS Dequeue Operation in a Global Transaction?
- XA Configuration
- Does the JMS Adapter Work with the WebLogic JMS Provider?

How Do I Create a JMS Dequeue Operation in a Global Transaction?

Assume you have to design the following simple scenario for a customer:

The BPEL process is initiated by a JMS message, stores some data to the database, and sends a confirmation message through another JMS queue.

The challenge is error handling. There are two types of errors:

- Unrecoverable
- Recoverable

Unrecoverable errors (for example, XML messages not being valid against the schema) need human interaction and are sent through a response queue back with an error code. This is easy to implement. In addition, recoverable errors also appear (for example, a database server is down). For those errors, messages are supposed to stay in the original queue. Therefore, you must roll back the dequeue operation. Oracle BPEL Process Manager by default reads a message from the queue, stores the message to an internal queue, and commits this transaction. A free WorkerBean then picks the message up and initiates an asynchronous BPEL process.

If you must perform a dequeue operation within one global transaction with the BPEL process itself, you must do following:

- The JMS adapter must be an XA adapter (`xa-queue-connection-factory` element in `jms.xml`).
- The JMS adapter must participate in a global transaction (`isTransacted` property set to `true` in `oc4j-ra.xml`).
- The JMS adapter must invoke a BPEL process synchronously (the `ConsumeMessage` operation must have a response definition in the adapter's WSDL file). See the Adapter Life-Cycle Management chapter of *Oracle Application Server Adapter Concepts*.

XA Configuration

The following is required to make two invokes commit or roll back as a unit:

- Both the JMS adapter invokes must be configured to participate in global transactions
- Both invokes must participate in the same global transaction
- The failure of either invoke must cause the global transaction to roll back

Configuring JMS Adapter for Global Transaction Participation

- Deployment descriptor (`oc4j-ra.xml`) changes:

Property `connectionFactoryLocation` must point to the X-enabled connection factory. The property `isTransacted` must also be set to `false`.

An example of `oc4j-ra.xml` when using the AQJMS Provider is shown below:

```
<connector-factory location=....
<config-property name="connectionFactoryLocation"
value="java:comp/resource/ojmsdemo/XAQueueConnectionFactories/myQCF"/>
<config-property name="isTransacted" value="false"/>
<config-property name="username" value="jmsuser"/>
<config-property name="password" value="jmsuser"/>
 </connector-factory>
```

- Configure separate connector factories (in `oc4j-ra.xml`) for inbound and outbound adapter interactions:

  For the AQJMS provider, it implies using separate OJMS resource providers (defined in *`J2EE_HOME`*`/config/application.xml`) for inbound, outbound JMS destinations (queues or topics), and a connection factory participating in the same global transaction.

- Set a new partner link property (Oracle BPEL Process Manager) or endpoint property (Oracle ESB) named `cacheConnections` to `false`. If unspecified, then the default value of `true` is used (the default behavior in both 10.1.2 and 10.1.3).

Both Invokes in the Same Global Transaction

Once the invokes participate in global transactions, they must participate in the same global transaction in order to commit or roll back as a unit. In Oracle BPEL Process Manager, this requires understanding the transaction's boundaries: at what points a checkpoint to write to the dehydration store commits the current global transaction and starts a new one. The transaction boundaries in a BPEL process are:

- Before a receive activity (but *not* the initial one)
- Before a wait activity (otherwise, Oracle BPEL Server can run into a transaction timeout)
- Before an onMessage or pick (extended onMessage) activity
- When invoking a synchronous child BPEL process, unless the transaction `participate` property is set on the partner link. Otherwise, the parent process is broken into two transactions and the child process executes in its own transaction.

Failure Must Cause Rollback

The global transaction `participate` property is set in `bpel.xml`. When Oracle BPEL Process Manager catches a fault from the database adapter invoke, it allows the fault to propagate up and cause a global rollback.

How Do You Configure Transaction Participation? On a child process invoke, set the transaction `participate` property on the partner link:

```
<partnerLinkBinding name="...">
   <property name="transaction">participate</property>
   </partnerLinkBinding>
```

To set the global property so that a failed invoke causes the entire transaction to roll back, perform the following:

```
</partnerLinkBindings> <property name="transaction">participate</property>
 </BPELProcess>
```

Related Questions

This section describes related XA configuration questions.

Why Do I Receive a 'CCI Local Transaction COMMIT failed' Error When Using XA? When the property `isTransacted` is set to `true`, it results in the above error. A sample error is shown below:

```
ORABPEL-12101 ERRJMS_TRX_COMMIT. CCI Local Transaction COMMIT failed due to:
 ERRJMS_COMMIT_FAIL. Unable to commit transaction. Please examine the log file to
 determine the problem.

 at
oracle.tip.adapter.jms.JmsCciLocalTransactionImpl.commit(JmsCciLocalTransactionImp
l.java:94)
at
oracle.tip.adapter.fw.wsif.jca.WSIFOperation_
 JCA.executeRequestResponseOperation(WSIFOperation_JCA.java:514)
oracle.tip.adapter.fw.AdapterFrameworkListenerBase.executeDeliveryServiceSend(Adap
terFrameworkListenerBase.java:573)
at

oracle.tip.adapter.fw.AdapterFrameworkListenerBase.deliveryServiceSend(AdapterFram
eworkListenerBase.java:629)
at

oracle.tip.adapter.fw.jca.AdapterFrameworkListenerImpl.performSingleActivation(Ada
pterFrameworkListenerImpl.java:966)
at

oracle.tip.adapter.fw.jca.AdapterFrameworkListenerImpl.onMessage(AdapterFrameworkL
istenerImpl.java:813)
at

oracle.tip.adapter.fw.jca.messageinflow.MessageEndpointImpl.onMessage(MessageEndpo
intImpl.java:293)
at
oracle.tip.adapter.jms.inbound.JmsConsumer.doSend(JmsConsumer.java:571)
at
oracle.tip.adapter.jms.inbound.JmsConsumer.sendInboundMessage(JmsConsumer.java:507
)
at oracle.tip.adapter.jms.inbound.JmsConsumer.send(JmsConsumer.java:353)
 at oracle.tip.adapter.jms.inbound.JmsConsumer.run(JmsConsumer.java:274)
at oracle.tip.adapter.fw.jca.work.WorkerJob.go(WorkerJob.java:51)
at oracle.tip.adapter.fw.common.ThreadPool.run(ThreadPool.java:272)
at java.lang.Thread.run(Thread.java:595) Caused by:
javax.jms.TransactionInProgressException: JMS-239: Illegal attempt to call commit
 method on a XASession.
at oracle.jms.AQjmsError.throwTranInProgressEx(AQjmsError.java:592)
at oracle.jms.AQjmsXASession.commit(AQjmsXASession.java:147)
at oracle.tip.adapter.jms.JmsTransactionImpl.commit(JmsTransactionImpl.java:90)
at

oracle.tip.adapter.jms.JmsCciLocalTransactionImpl.commit(JmsCciLocalTransactionImp
l.java:89)
```

Ensure that the property is set to `false` when running under Global (XA) transaction semantics.

Does the JMS Adapter Work with the WebLogic JMS Provider?

With release 10.1.3.3, WebLogic JMS integration support is provided through the JMS adapter. The integration has been tested successfully with WebLogic 7.*x*, 8.1.*x*, and 9.*x*. See the version-specific instructions in the following sections.

Note: Because of limitations in the BEA JMS client implementation, integration using WebLogic JMS XA is not supported. Ensure that the WebLogic JMS connection factories used with the JMS adapter are not configured for XA support.

WebLogic JMS (For All Tested Versions)

1. Copy `weblogic.jar` to the *`SOA_ORACLE_HOME`*`/j2ee/`*`instance_name`*`/connectors/JmsAdapter/JmsAdapter` directory.
2. Add connection factory properties to `oc4j-ra.xml` based on the following examples. Substitute your server values based on your WebLogic configuration.

```
<connector-factory location="eis/wljms/Queue" connector-name="Jms Adapter">
<config-property name="connectionFactoryLocation"
 value="<global-jndi-name-of-your-non-XA-jmsconnectionfactory-in-weblogic>"/>
 <config-property name="factoryProperties"
value="java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory;java.n
aming.provider.url=t3://<your-server-name>:<your-server-port;java.naming.securi
ty.principal=<username>;java.naming.security.credentials=<password>"/>
 <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE"/>
 <config-property name="isTopic" value="false"/>
<config-property name="isTransacted" value="true"/>
<config-property name="username" value=""/>
<config-property name="password" value=""/>
</connector-factory>
<connector-factory location="eis/wljms/Topic" connector-name="Jms Adapter">
 <config-property name="connectionFactoryLocation"
 value="<global-jndi-name-of-your-non-XA-jmsconnectionfactory-in-weblogic>"/>
 <config-property name="factoryProperties"
value="java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory;java.n
aming.provider.url=t3://<your-server-name>:<your-server-port;java.naming.securi
ty.principal=<username>;java.naming.security.credentials=<password>"/>
 <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE"/>
 <config-property name="isTopic" value="true"/>
<config-property name="isTransacted" value="true"/>
<config-property name="username" value=""/>
<config-property name="password" value=""/>
</connector-factory>
```

   Since the WebLogic Server does its authentication during JNDI `InitialContext` creation, you must never specify a value for the `"username"` and `"password"` `config-property` fields. These fields must be present and set to empty strings as shown above. The WebLogic username and password are specified only in the `java.naming.security.principal=`*`username`* and `java.naming.security.credentials=`*`password`* fields of the `factoryProperties` `config-property`. If authentication is not required, remove these two fields from the `factoryProperties` `config-property`.

Notes:

- The `isTransacted` config-`property` value can be set to `true` or `false`. If set to `true`, this tells the JMS adapter to use a JMS transacted session (which means that only JMS operations performed on this JMS session are included in a transaction). As mentioned earlier, XA integration with WebLogic JMS is not currently supported.
- For all WebLogic JMS versions, ensure that the JMS adapter `ActivationSpec` property `UseMessageListener` is set to `false`.

WebLogic 9.*x* JMS Specific Configuration

WebLogic 9.*x* changed the JNDI client-side implementation in a way that causes OC4J JNDI lookups to fail. As a result, you must make the following changes to your Oracle SOA Suite configuration. Failure to do so generates Oracle BPEL Process Manager and Oracle ESB errors related to not finding JMS destinations and transaction errors.

- `server.xml` — Add the property `environment-naming-url-factory-enabled="true"` in the `server.xml` configuration file for your instance for any application that includes `weblogic.jar` from WebLogic 9.*x*.

  ```
  <application-server  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/applicat
  ion-server-10_1.xsd"
  localhostIsAdmin="true"
  application-directory="../applications"
  check-for-updates="adminClientOnly"
   deployment-directory="../application-deployments"
   connector-directory="../connectors"
   environment-naming-url-factory-enabled="true"
   schema-major-version="10" schema-minor-version="0">
  ```

- The following two patches on top of 10.1.3.3 are also required for an instance that has Oracle ESB installed (whether it is being used or not):
  - Bug 6081699 Patch — Available on Metalink and ARU (request number 9410526)
  - Bug 6316554 Patch — Available on Metalink and ARU (request number 9423888)

Adapter Framework

This section provides answers to frequently asked questions about the adapter framework.

This section contains the following topics:

- Changing Inbound Adapter Parameters at Runtime
- How Do I Throttle Inbound Message Flow?
- Adapter Clustering — JGroups Configuration
- Inbound Message Flow Multiplexing (Server Distribution)

- Scheduling Endpoint Activation
- Tuning JCA Connection Cache
- Adapter SDK
- Rejection Handlers
- Dynamic BPEL Process Routing Support by the Adapter Framework
- Enabling Debug Logging for Detailed Troubleshooting

See Also: The following adapter framework sample:

SOA_ORACLE_HOME\bpel\samples\tutorials\140.AdapterFramework

Changing Inbound Adapter Parameters at Runtime

The inbound (`ActivationSpec`) properties can only be set dynamically (typically through Oracle BPEL Control) if their values are bound through property values defined in `bpel.xml` for the particular JCA activation agent in question. This is fundamentally as per the JCA 1.5 design where `EndpointActivation` (which receives an instance of `ActivationSpec`) takes place when the BPEL process starts (for example, only once).

More often than not, the BPEL process must be restarted (off/on) for `ActivationSpec` attribute changes to take effect (thereby allowing the JCA 1.5 inbound message endpoint to be restarted and reinitialized).

Some adapters look dynamically for changes to `bpel.xml`-bound values (subscribed to through the adapter framework) and react appropriately without the need for restarting the process (for example, adjusting a timeout or `retryCount` or other secondary attribute that does not change the first order attributes of the endpoint, such as the database host name, and so on).

An example of an activation property that requires the adapter to restart is the JMS adapter `MessageSelector` property.

If possible, try to use a synchronous `InteractionSpec` (invoke), which can perform the required read-type interaction (for example, `JmsReceiveNoWaitInteractionSpec`).

How Do I Throttle Inbound Message Flow?

The `minimumDelayBetweenMessages` activation agent property (set in `bpel.xml`, and available since 10.1.3.1) can control the speed at which the adapter posts messages to Oracle BPEL Process Manager.

```
<activationAgents>
<activationAgent partnerLink="JmsDequeuePL" ... >
<property name="minimumDelayBetweenMessages">1000</property>
</activationAgent>
</activationAgents>
</BPELProcess>
</BPELSuitcase>
```

This setting ensures at least a `1000` milliseconds delay between two consecutive messages being posted to the BPEL process.

Note: This setting pertains only to Oracle BPEL Process Manager, and only to one adapter polling thread. If multiple adapter polling threads (for example, multiple JMS dequeuer threads) are configured, this setting controls the speed of each thread, not the combined speed.

Adapter Clustering — JGroups Configuration

The adapter framework supports active failover of inbound adapter services. You can achieve this by adding a property to a particular JCA activation agent (in `bpel.xml`), as shown in the following example:

```
<activationAgents>
  <activationAgent className="..." partnerLink="MyInboundAdapterPL">
    <property name="clusterGroupId">myBpelCluster</property>
    <property name="clusterAcrossSubnet">false</property>
```

If the Oracle BPEL Servers (JVMs) in the cluster are located across TCP/IP subnet boundaries, then you must add the attribute `clusterAcrossSubnet=true`.

In a cluster group, the multiple activations of the same (for example, file) adapter activation agent (for a specific partner link) are detected implicitly and automatically by all instances of the adapter framework active in that cluster. Only one activation is allowed to start reading or publishing messages. The adapter framework instances randomly choose one among them to assume the primary activation responsibility. The other activations (instances) in the cluster initiate to a hot stand-by state, without actually invoking endpoint activation on the JCA resource adapter.

If a primary activation at some point becomes unresponsive, it is deactivated manually. If it crashes and exits, then any one of the remaining adapter framework members of the cluster group immediately detect this, and reassign the primary activation responsibility to one of the standby activation agents. This feature uses JGroups underneath for the implementation (therefore, the reason for the `clusterGroupId` property).

Each individual adapter endpoint activation (per activation agent and cluster node) joins the group (JChannel) defined in (name after) the `clusterGroupId` property.

Each time a particular endpoint or node is activated or deactivated, the endpoint either joins or leaves the particular group. The JGroups implementation is relied upon entirely to notify you about other endpoints or nodes either joining or leaving the group. In that case, JGroups also decides who the next primary member of the group is. The (presumably idle) endpoint gets notified that it is now the primary activation, then performs JCA endpoint activation. This is one of the key reasons that OC4J JCA container-managed endpoint activation cannot be relied on, which is tied to the application server life cycle.

This mechanism is entirely generic in its nature, not tied to a specific adapter type. For example, two (nonclustered) Oracle BPEL Process Manager nodes may have a file adapter endpoint on one node and a JMS adapter endpoint on the second node join the same cluster group. Depending on the starting order of the two nodes, either the file adapter or the JMS adapter becomes active.

The reason for making this point is that the active and passive failover adapter framework feature does not consider endpoint characteristics or actual payload or payload location (directory, file name, and so on).

You must configure JGroups details in *`SOA_ORACLE_HOME`*`/bpel/system/config/jgroups-protocol.xml`.

See *Oracle Application Server Enterprise Deployment Guide* for additional information.

Inbound Message Flow Multiplexing (Server Distribution)

Within Oracle BPEL Process Manager, the adapter framework supports Oracle BPEL Server distribution (fan-out) for inbound messages. This enables a particular endpoint activation to evenly spread (round robin) the inbound message load among the Oracle BPEL Servers in a cluster or to an individual (nonclustered) Oracle BPEL Server. This feature is available in 10.1.2 and onwards.

This load distribution feature can be configured in the `bpel.xml` JCA activation agent as follows:

```
<activationAgents>
<activationAgent className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
 partnerLink="AdapterInboundPL">
<property name="bpelServers">
bpel.host1.net:23791, bpel.host2.net:23791, bpel.host3.net:23791
</property>
</activationAgent>
</activationAgents>
</BPELProcess>
</BPELSuitcase>
```

The port number is the ORMI (request) port of the OC4J instance on which Oracle BPEL Process Manager is running.

With the above sample configuration, the adapter framework distributes received inbound messages (for a particular partner link or endpoint) to the three configured servers in a plain round robin fashion.

Notes:

- For a nonclustered case, only specify the activation agent in `bpel.xml` for one of the three servers. Otherwise, all three nodes start performing this distribution. Leaving out the activation agent in `bpel.xml` for server two and three simply prevents JCA endpoint activation. However, these servers are still ready to receive messages (internally through ORMI). One additional (trivial) assumption is that the same BPEL process (suitcase) has been deployed to all Oracle BPEL Servers in the distribution set.
- For a clustered case, you must also specify the activation agent property `clusterGroupId` to avoid all cluster nodes in order to apply the above load distribution. All Oracle BPEL Servers mentioned in the `bpelServers` property must have the same server credentials.
- See bug 5594867.

Scheduling Endpoint Activation

This adapter framework feature (since 10.1.3.1), which is only available in Oracle BPEL Process Manager (and not Oracle ESB), is enabled in `bpel.xml` as follows. The attributes and their values are shown in bold:

```
<activationAgents>
<activationAgent className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
partnerLink="FileFtpInboundPL" heartBeatInterval="10">
```

```
<property name="schedulerCallout">DefaultSchedulerCalloutImpl</property>
<property name="endpointScheduleOn">*0 6 1 ? * **</property>
<property name="endpointScheduleOff">*0 8 1 ? * **</property>
 </activationAgent>
</activationAgents>
</BPELProcess>
</BPELSuitcase>
```

After adding the new properties and attributes, only change the values. The heartbeat interval is measured in seconds (determines how frequently the schedule is checked).

These properties can also be changed at runtime through Oracle BPEL Control (**Descriptor** tab), and take effect immediately.

The scheduler expressions follow Quartz syntax, which is described in this document:

`http://quartz.sourceforge.net/javadoc/org/quartz/CronTrigger.html`

You can also replace `DefaultSchedulerCalloutImpl` (shipped) with your own. Just ensure that it implements the following:

```
oracle.tip.adapter.api.callout.SchedulerCallout {
public void init(java.util.Map partnerlinkProperties, LogManager logger) throws
 Exception;
public boolean isEndpointActive(java.util.Map partnerlinkProperties, boolean
 isActive) throws Exception;
```

If you package your class in `oracle.tip.adapter.fw.common`, you can leave out the Java package name when specifying it in `bpel.xml`.

Notes:

- This feature is not using Quartz as such. It only uses it from a syntactical point of view, meaning the activation schedule uses Quartz notation. The scheduling mechanism itself is jointly (and directly) controlled by the BPEL activation framework and the adapter framework without reliance on any other component (using the heartbeat option of the BPEL activation framework).
- To use this feature, the first activity must be a receive, since the adapter framework must always be wired to an activity that can consume the payload being delivered (from the adapter).
- The endpoint activation itself does not necessarily mean the triggering of the BPEL process. A ready message must be available on the endpoint for that to occur.

Tuning JCA Connection Cache

The JCA WSIF provider supports a JCA connection pool, which resides in the JCA `WSIFPort` implementation class.

The JCA connection pool is typically boundless, but it can be capped (through a partner link property). For example, if the maximum size is `10`, and `15` concurrent threads are trying to invoke the same endpoint, `5` of them throw a (retryable) remote fault. This connection pool ensures single threading through JCA `LocalTransactions`.

For Oracle BPEL Process Manager, these properties are configured in `bpel.xml`. For Oracle ESB, they are configured in the `.esbsvc` file as endpoint properties and normally set through Oracle ESB Control.

```
<property name="useJCAConnectionPool">true</property>
```

The above property is typically derived from the declared transactional support of the adapter (for example, the file adapter does not use this connection pool since it is multithread safe), but can be overridden through this property:

```
<property name="maxSizeJCAConnectionPool">500</property>
```

If the above property is not specified, unbounded is assumed. This applies on a per `WSIFPort` (partner link) basis.

```
<property name="lruConnectionMaxIdleAge">50000</property>
```

The above property specifies the maximum age of idle connections in the pool. This is important because some types of connections hold onto expensive external resources (for example, database shadow processes). The value is measured in milliseconds.

```
<property name="lruConnectionCheckInterval">10000</property>
```

The above property specifies how often to perform the idle connection scan (measured in milliseconds).

Adapter SDK

Contact Oracle Support Services for the adapter SDK.

Here are some additional pointers:

The J2EE connector architecture defines a standard architecture for bidirectional connectivity between the J2EE platform and heterogeneous legacy systems. For different backend systems, you need a different adapter. Note that a different approach is required for writing different adapters. For example, what is the backend system for your adapter? The best way to determine this is to follow the JCA 1.5 specification. You must implement all the system contracts, including transaction management, connection management, security management, and so on. If this is for a customer, Oracle may suggest that you use an Oracle adapter or third party adapter. The following URL provides information for review, including information on how to write an adapter for a Telnet client:

```
http://www.oracle.com/technology/tech/java/newsletter/september04.html
```

Oracle Containers for J2EE Resource Adapter Administrator's Guide is available at the following URL:

```
http://iasdocs/iasdl/101310_final/web.1013/b28956/toc.htm
```

JCA information on how to use and deploy adapters is available at the following URLs:

```
http://www.oracle.com/technology/tech/java/oc4j/1013/how_
to/how-to-jca-intro/doc/how-to-jca-intro.html

http://www.oracle.com/technology/tech/java/oc4j/1013/how_
to/how-to-connect-to-mqseries/doc/how-to-connect-to-mqseries.html
```

Rejection Handlers

This feature lets you configure your BPEL process to execute the correct records of a file and write only the rejected records to an archive directory by setting the `rejectedMessageHandlers` parameter in the `bpel.xml` file. However, the rejection handlers come in four different types.

All of the following `bpel.xml` properties must be defined as activation agent properties; for example:

```
<BPELSuitcase>
  <BPELProcess src="ErrorTest.bpel"  id="ErrorTest">
    <activationAgents>

<activationAgentclassName="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
                        partnerLink="inboundPL">
         <property name="rejectedMessageHandlers">
           file://C:/orabpel/samples/test/errorTest/rejectedMessages
         </property>
```

Therefore, they only apply to inbound (WSDL) operations (BPEL receive).

See Also: *Oracle Application Server Adapter Concepts* for additional details about rejection handlers

File System-Based Rejection Handler

```
<property name="rejectedMessageHandlers">
 file://directory_path
</property>
```

for example

```
<property name="rejectedMessageHandlers">
 file://C:/orabpel/domains/default/rejectedMessages
</property>
```

This rejection handler is straight forward. Bad messages are written to the configured directory using the file name pattern `INVALID_MSG_` + *`process_name`* + *`operation_name`* + *`current_time`*.

RAW Oracle Advanced Queue-Based Rejection Handler

```
<property name="rejectedMessageHandlers">
 queue://jdbc:oracle:thin:@db_host:tns_port:sid|user/password|queue_name
</property>
```

The *`password`* can be encrypted. For example:

```
<property name="rejectedMessageHandlers">
 queue://jdbc:oracle:thin:@acme-sun:1521:ORCL|scott/tiger|JCA_BAD_MESSAGES
</property>
```

This rejection handler allows you to designate an Oracle RDBMS RAW AQ queue as the rejection storage. Note that the symbols `:` and `|` must appear in the places shown. Also note that the password can be encrypted using the `encrypt.bat` utility in `orabpel/bin`.

BPEL Process Rejection Handler

```
<property name="rejectedMessageHandlers">
 bpel://bpel_domain[:password]|process_name|operation_name|input_message_part_name
```

```
</property>
```

The *password* for the domain (if not `bpel`) can be encrypted. The symbols `[` and `]` indicate optional entries. For example:

```
<property name="rejectedMessageHandlers">
  bpel://default|JCA-RejectionHandler|handleRejection|message
</property>
```

This rejection handler sends the bad message to another (designated error handling) BPEL process. Therefore, you can define a process with a receive operation of your own choosing (WSDL and BPEL source). The only constraint is on the message type of the message that is sent to this rejection handler.

It must be declared to have the type `RejectedMessage`. This can conveniently be achieved by importing the `xmllib` resident WSDL `RejectionMessage.wsdl`, which defines such a message:

```
<message name="RejectionMessage">
  <part name="message" element="err:RejectedMessage"/>
</message>
```

An `xmllib` WSDL import (from another WSDL) is achieved using the well-known URL:

```
<import namespace="http://xmlns.oracle.com/pcbpel/rejectionHandler"
      location="http://localhost:9700/orabpel/xmllib/jca/RejectionMessage.wsdl"/>
```

For example, the receive operation WSDL that you define for the rejection handler BPEL process simply contains this import. The port type then references this:

```
<definitions ...
     xmlns:rej="http://xmlns.oracle.com/pcbpel/rejectionHandler"
  <portType name="MyRejectionHandlerPortType">
    <operation name="myHandleRejectionOperation">
      <input message="rej:RejectionMessage"/>
    </operation>
  </portType>
```

WSIF-Based Rejection Handler

```
<property name="rejectedMessageHandlers">
 wsif://wsif_wsdl_location\|operation_name\|input_message_part_name
</property>
```

For example:

```
<property name="rejectedMessageHandlers">

wsif://file:/C:/orabpel/samples/test/ErrorTest/FileAdapterWrite.wsdl|write|message
</property>
```

This last rejection handler lets you configure any type of WSIF WSDL (JCA, EJB, JMS, HTTP, Java, and so on). That is, any kind of service that can be reached through WSIF as the bad message handler. The same constraint for the message type, as described in "BPEL Process Rejection Handler" on page 3-32, also applies here.

Adapter Fatal Error Failover BPEL Process

```
<property name="fatalErrorFailoverProcess">
  bpel://bpel_domain[:password]|process_name|operation_name|input_message_part_
name
```

```
</property>
```

The *password* for the domain (if not `bpel`) can be encrypted. The symbols `[` and `]` indicate optional entries. For example:

```
<property name="fatalErrorFailoverProcess">
  bpel://default|JCA-FatalErrorHandler|handleError|message
</property>
```

When an adapter detects a disastrous, unrecoverable situation (no more memory, file handles, disk space, or something similar), it can instruct the adapter framework to shut down the BPEL process that owns the adapter endpoint activation. To activate some type of compensating BPEL process, you can configure this `bpel.xml` activation agent property.

As with the BPEL process rejection handler, the fatal error BPEL process must use a message type as described above. However, in this case, the message type is defined in `FatalErrorMessage.wsdl`. Therefore, the fatal error WSDL imports the following:

```
<import namespace="http://xmlns.oracle.com/pcbpel/fatalErrorHandler"
 location="http://localhost:9700/orabpel/xmllib/jca/FatalErrorMessage.wsdl"/>
```

Dynamic BPEL Process Routing Support by the Adapter Framework

This section describes dynamic BPEL process routing support by the adapter framework.

Motivation

In high performance systems where BPEL processes are instantiated through *inbound* JCA adapter event notifications and an intermediate (content-based router) BPEL process using a switch construct must determine which ultimate BPEL process (actual Request Handler) must process the message, there is an option that lets the adapter framework perform the switch and dispatch operation. This option enables one BPEL process to be bypassed (the dispatcher).

Example Configuration 1

One BPEL process always owns the JMS adapter endpoint activation (for example, the most basic request handler). This is required for bootstrapping the JMS adapter. However, the routing happens in the inbound adapter framework layer. It is facilitated through the following `bpel.xml` activation agent properties:

```
<activationAgents>
   <activationAgent partnerLink="JMS_DispatcherPL" className=
               "oracle.tip.adapter.fw.agent.jca.JCAActivationAgent" >
       <property name="processNameRoutingRule">
          <![CDATA[
            <xpath-expression operand="HEADERS"
                     xmlns:ns1="http://www.mycompany.com/REQUEST" >
              /ns1:Request/ns1:Header/ns1:HeaderName
            </xpath-expression>
          ]]>
       </property>
       <property name="operationNameRoutingRule">
          <![CDATA[
            <xpath-expression operand="PAYLOAD"
                     xmlns:ns1="http://www.mycompany.com/REQUEST" >
              ora:extractmycompanyOperationName(
                       '/ns1:Request/ns1:Header/ns1:SomeOtherHeader')
            </xpath-expression>
          ]]>
        </property>
    </activationAgent>
</activationAgents>
```

This (pseudo example) construct causes the adapter framework to invoke whichever BPEL process and operation is determined by the evaluated values of these two properties. This is determined either through plain XPath expressions (process name above) or invocation of custom XPath functions (operation name above) — operating on either the adapter headers or payload DOM — as instructed through the operand attribute.

In the example, the BPEL process name is directly determined as the value of a payload XML element. The operation name is obtained through an XPath function call.

Also note that each of the (derived) target process and operation pairs can have different WSDL message definitions, since no XML validation is taking place in the JMS adapter or in the adapter framework. This approach alleviates two problems:

- Having to redeploy the BPEL dispatcher process whenever a new process type must be introduced
- Enabling dynamic determination of message routing, if the XPath function obtains routing information from a database table

There is no user interface support for this, since ultimately Oracle ESB owns this type of functionality.

Example Configuration 2

The dispatcher process is just a dummy process used to start the (JMS) adapter endpoint activation (event reader).

`HandlerX` of the process receives incoming messages (`inputX.xml`) based on contents in the input files.

This is controlled by the following construct in `bpel.xml`:

```
<activationAgents>
    <activationAgent
 className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
                    partnerLink="inboundPL">
        <property name="routingRuleCallout">test.TestRouter</property>
        <property name="processNameRoutingRule">
            <![CDATA[
                <xpath-expression operand="PAYLOAD"
                xmlns:ns1="http://www.oracle.com/pcbpel/demo/csv">
                    //ns1:Contact
                </xpath-expression>
            ]]>
        </property>
        <property name="operationNameRoutingRule">
            <![CDATA[
                <xpath-expression operand="PAYLOAD"
                xmlns:ns1="http://www.oracle.com/pcbpel/demo/csv">
                    //ns1:Mailstop[1]/text()
                </xpath-expression>
            ]]>
        </property>
    </activationAgent>
</activationAgents>
```

This example is a bit overloaded, since it uses both a Java callout (always used first, example is below), and the XPath expressions. Also note that currently the XPath expressions can only contain path constructs, and not functions.

Here is the Java source code for the routing rule callout referenced above:

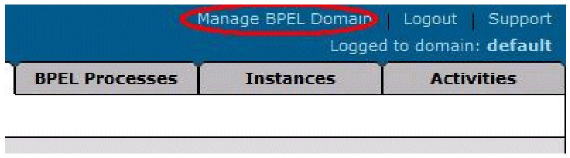
```
package test;

import com.oracle.bpel.client.NormalizedMessage;

import oracle.tip.adapter.api.callout.RoutingRuleCallout;

import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class TestRouter implements RoutingRuleCallout
{
    private long counter = 0;

    public TestRouter() {
    }

   public String getProcessName(Element payload, Element headers)
    {

        NodeList nl = payload.getElementsByTagName("Mailstop");

        if (nl != null && nl.getLength() > 0 && nl.item(0) != null)
        {
            Element e = (Element)nl.item(0);
            if ("1OP601".equals(e.getFirstChild().getNodeValue()))
                return "Handler1";
            else
            if ("1OP602".equals(e.getFirstChild().getNodeValue()))
                return "Handler2";
            else
            if ("1OP603".equals(e.getFirstChild().getNodeValue()))
                return "Handler3";
        }
        return "Dispatcher";
    }
   public String getOperationName(String processName, Element payload,
                                  Element headers)
    {
        NodeList nl = payload.getElementsByTagName("Mailstop");

        if (nl != null && nl.getLength() > 0 && nl.item(0) != null)
        {
            Element e = (Element)nl.item(0);
            if ("1OP601".equals(e.getFirstChild().getNodeValue()))
                return "Receive1";
            else
            if ("1OP602".equals(e.getFirstChild().getNodeValue()))
                return "Receive2";
            else
            if ("1OP603".equals(e.getFirstChild().getNodeValue()))
                return "Receive3";
        }
        return "Receive";
    }
    public String getRevision(String processName, Element payload,
                              Element headers) {
        return null;
    }

    public String getPartnerlinkName(String processName, Element payload,
                                     Element headers) {
        return null;
    }

    public String getRoleName(String processName, Element payload,
                              Element headers) {
        return null;
    }
}
```

Enabling Debug Logging for Detailed Troubleshooting

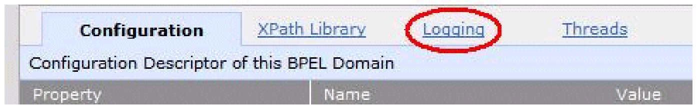
Enabling **Debug** level logging for BPEL loggers in Oracle BPEL Control is essential to adapter troubleshooting. The loggers to set are as follows:

```
* _<domain>_.collaxa.cube.activation
* _<domain>_.collaxa.cube.ws
```

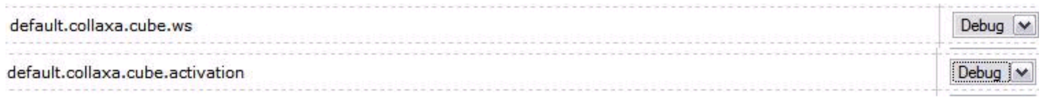
1. Click **Manage BPEL Domain** on the Oracle BPEL Control main page.


Manage BPEL Domain | Logout | Support
Logged to domain: default
BPEL Processes
Instances
Activities

2. Click **Logging**.



3. Set the levels of the loggers to the **Debug** level:



default.collaxa.cube.ws Debug
default.collaxa.cube.activation Debug

4. Click **Apply** in the lower right corner of the page.
5. Rerun the scenario with these settings and upload the log file generated in the following directory to some shared location.

   SOA_ORACLE_HOME/bpel/domains/*domain_name*/logs/domain.xml

4

Oracle BPM Human Workflow

This chapter describes how to use the Workflow Services API Reference for optimal performance.

This chapter contains the following topics:

- Using the Workflow Services API Reference
- Workflow Modeling

Using the Workflow Services API Reference

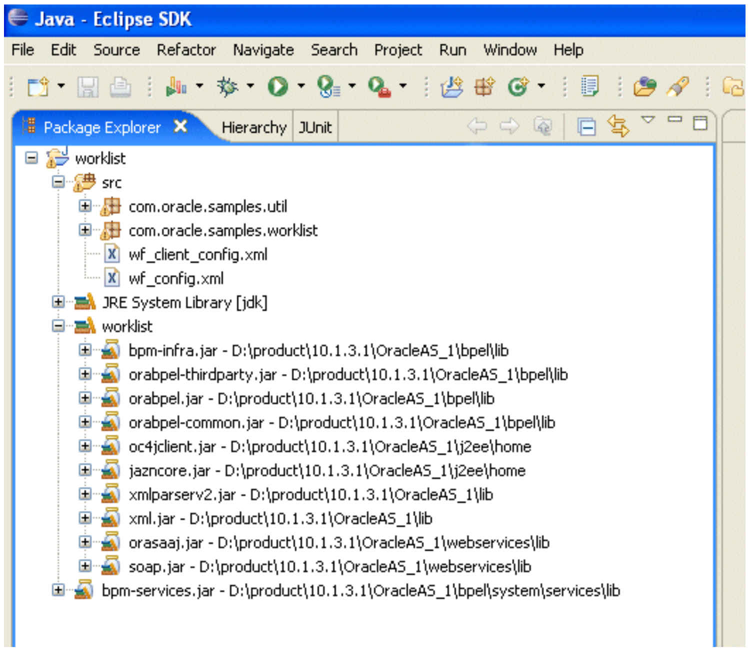
This section describes how to use the Workflow Services API Reference.

This section contains the following topics:

- Configuring the Workflow Services API Reference
- Java Interface that Exposes the Methods to Get the Task List, Task Details, and Update Outcomes
- Implementation and Usage in main() Method
- Task Utility that Prints Payload Contents

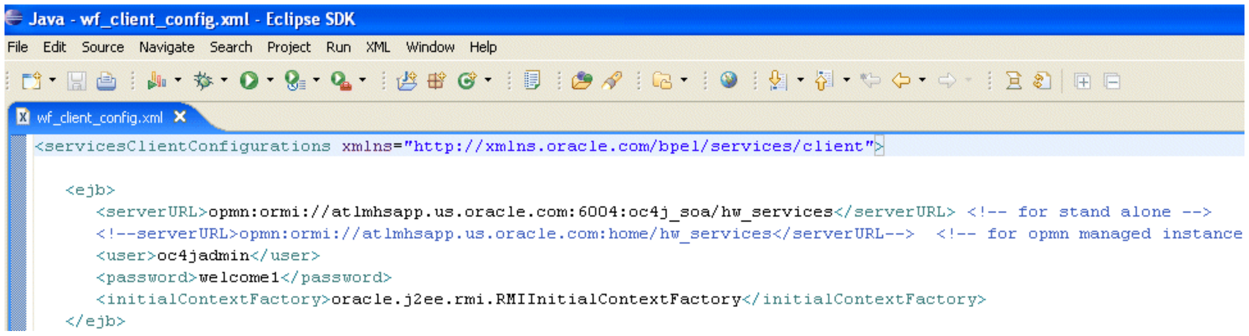
Configuring the Workflow Services API Reference

1. Include the following jar files in your classpath. For this example, the Oracle home directory is `d:\product\10.1.3.1\OracleAS_1`.


Java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Hierarchy
JUnit

```
worklist
    src
        com.oracle.samples.util
        com.oracle.samples.worklist
        wf_client_config.xml
        wf_config.xml
    JRE System Library [jdk]
    worklist
        bpm-infra.jar - D:\product\10.1.3.1\OracleAS_1\bpel\lib
        orabpel-thirdparty.jar - D:\product\10.1.3.1\OracleAS_1\bpel\lib
        orabpel.jar - D:\product\10.1.3.1\OracleAS_1\bpel\lib
        orabpel-common.jar - D:\product\10.1.3.1\OracleAS_1\bpel\lib
        oc4jclient.jar - D:\product\10.1.3.1\OracleAS_1\j2ee\home
        jazncore.jar - D:\product\10.1.3.1\OracleAS_1\j2ee\home
        xmlparserv2.jar - D:\product\10.1.3.1\OracleAS_1\lib
        xml.jar - D:\product\10.1.3.1\OracleAS_1\lib
        orasaaj.jar - D:\product\10.1.3.1\OracleAS_1\webservices\lib
        soap.jar - D:\product\10.1.3.1\OracleAS_1\webservices\lib
    bpm-services.jar - D:\product\10.1.3.1\OracleAS_1\bpel\system\services\lib
```

2. Specify the `serverURL` value in the `wf_client_config.xml` file. This value takes the form of `opmn:ormi://`*`hostname`*`:`*`opmn_request_port`*`:`*`instance_name`*`/hw_services`:


Java - wf_client_config.xml - Eclipse SDK
File Edit Source Navigate Search Project Run XML Window Help
wf_client_config.xml

```
<servicesClientConfigurations xmlns="http://xmlns.oracle.com/bpel/services/client">

   <ejb>
      <serverURL>opmn:ormi://atlmhsapp.us.oracle.com:6004:oc4j_soa/hw_services</serverURL> <!-- for stand alone -->
      <!--serverURL>opmn:ormi://atlmhsapp.us.oracle.com:home/hw_services</serverURL-->  <!-- for opmn managed instance
      <user>oc4jadmin</user>
      <password>welcome1</password>
      <initialContextFactory>oracle.j2ee.rmi.RMIInitialContextFactory</initialContextFactory>
   </ejb>
```

If you are accessing over SOAP, change the SOAP endpoint appropriately.

Java Interface that Exposes the Methods to Get the Task List, Task Details, and Update Outcomes

The Java interface that exposes the `getTaskList`, `getTaskDetail`, and `updateTaskOutCome` methods is shown below:

```
package com.oracle.samples.worklist;

import java.util.List;
import java.util.Map;

import oracle.bpel.services.workflow.StaleObjectException;
import oracle.bpel.services.workflow.WorkflowException;
import oracle.bpel.services.workflow.metadata.TaskMetadataServiceException;
import oracle.bpel.services.workflow.task.model.Task;

/**
 * Get Task List for a User
 * @return
 * @throws WorkflowException
 */
public List<Task> getTaskList() throws WorkflowException;


/**
 * Get Task Detail based on Task ID
 * @param taskId
 * @return
 * @throws WorkflowException
 */
public Task getTaskDetail(String taskId) throws WorkflowException;


/**
 * Get Task Outcomes for a Task
 * @param task
 * @return
 * @throws TaskMetadataServiceException
 */
public Map<String, String> getOutComes(Task task) throws
 TaskMetadataServiceException;


/**
 * Update Task Outcomes for a Task ID
 * @param taskId
 * @param outCome
 * @throws WorkflowException
 * @throws StaleObjectException
 */
public void updateTaskOutCome(String taskId, String outCome) throws
 WorkflowException, StaleObjectException;
}
```

Implementation and Usage in main() Method

The implementation and usage in the `main()` method is shown below. An example of logging in on behalf of `jstein` in the `jazn.com` realm is shown in bold.

```
package com.oracle.samples.worklist;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import com.oracle.samples.util.TaskUtil;

import oracle.bpel.services.workflow.IWorkflowConstants;
import oracle.bpel.services.workflow.StaleObjectException;
import oracle.bpel.services.workflow.WorkflowException;
import oracle.bpel.services.workflow.client.IWorkflowServiceClient;
import oracle.bpel.services.workflow.client.WorkflowServiceClientFactory;
import oracle.bpel.services.workflow.metadata.ITaskMetadataService;
import oracle.bpel.services.workflow.metadata.TaskMetadataServiceException;
import oracle.bpel.services.workflow.query.ITaskQueryService;
import oracle.bpel.services.workflow.repos.Ordering;
import oracle.bpel.services.workflow.repos.Predicate;
import oracle.bpel.services.workflow.repos.TableConstants;
import oracle.bpel.services.workflow.task.ITaskService;
import oracle.bpel.services.workflow.task.model.Task;
import oracle.bpel.services.workflow.verification.IWorkflowContext;
import oracle.tip.pc.services.identity.config.BPMConfigException;
public class TaskServiceImpl implements
 com.oracle.samples.worklist.IHumanTaskService {
private static final String MY_GROUP = "My+Group";
private ITaskQueryService querySvc = null;
private String userId;
private String password;
private IWorkflowContext ctx;
private String realm;
private ITaskMetadataService taskMetadataSvc;
private ITaskService taskSvc;

public TaskServiceImpl(String _userId, String _password, String _realm, String
 behalfOfUser) throws BPMConfigException, WorkflowException {
initUser(_userId, _password, _realm, behalfOfUser);
}

//public TaskService(HttpS)
public List<Task> getTaskList() throws WorkflowException {
return querySvc.queryTasks(ctx,
getQueryColumns(),
                getOptionalInfo(),
                MY_GROUP,
                null,
                getPredicate(null), getOrdering(), 0, 0);
}
public Task getTaskDetail(String taskId) throws WorkflowException {
 return querySvc.getTaskDetailsById(ctx,
                 taskId);
}

public Map<String, String> getOutComes(Task task) throws
 TaskMetadataServiceException {
return taskMetadataSvc.getOutcomes(ctx, task, null);
}

public void updateTaskOutCome(String taskId, String outCome) throws
 WorkflowException, StaleObjectException {
Task task = getTaskDetail(taskId);
updateTask(taskId, outCome, task);
}

private void updateTask(String taskId, String outCome, Task task) throws
 StaleObjectException, WorkflowException {
boolean acquiredBy = task.getSystemAttributes().getAcquiredBy() == null;
List assigneeUsers = task.getSystemAttributes().getAssigneeUsers();
        List assigneeGroups = task.getSystemAttributes().getAssigneeGroups();
if (acquiredBy && assigneeUsers.size() != 1 && assigneeGroups.size() == 0)
           task = taskSvc.acquireTask(ctx, taskId);
taskSvc.updateTaskOutcome(ctx, task, outCome);
}
public void updateTaskOutCome(Task task, String outCome) throws
 StaleObjectException, WorkflowException {
String taskId = task.getSystemAttributes().getTaskId();
updateTask(taskId, outCome, task);
}
```

```
private void initUser(String userId, String password, String realm, String
 behalfOfUser)
throws WorkflowException, BPMConfigException {
System.out.println("Initializing Service ===>");
initService();
System.out.println("Initializing Service Done ===>");
this.userId = userId;
this.password = password;
this.realm = realm;
System.out.println("Authenticating on behalf of ===>" + behalfOfUser);
ctx = querySvc.authenticate(this.userId, this.password, this.realm, behalfOfUser);
System.out.println("Authentication Done ===>");
}
private void initService() {
IWorkflowServiceClient wfSvcClient = WorkflowServiceClientFactory
.getWorkflowServiceClient(WorkflowServiceClientFactory.REMOTE_CLIENT);
querySvc = wfSvcClient.getTaskQueryService();
taskMetadataSvc = wfSvcClient.getTaskMetadataService();
taskSvc = wfSvcClient.getTaskService();
}
private Ordering getOrdering() throws WorkflowException {
Ordering ordering = new Ordering(
TableConstants.WFTASK_TASKNUMBER_COLUMN, true // Ascending
// order
, false // Nulls last
);
return ordering;
}
private Predicate getPredicate(String processName) throws WorkflowException {
Predicate predicate = new Predicate(TableConstants.WFTASK_STATE_COLUMN,
Predicate.OP_EQ, IWorkflowConstants.TASK_STATE_ASSIGNED);
if (processName != null) {
predicate.addClause(Predicate.AND,
TableConstants.WFTASK_PROCESSID_COLUMN, Predicate.OP_EQ,
(String) processName);
}
return predicate;
}
private List getOptionalInfo() {
List optionalInfo = new ArrayList();
return null;
}
private List<String> getQueryColumns() {
List<String> queryColumns = new ArrayList<String>();
queryColumns.add("CREATOR");
queryColumns.add("TITLE");
queryColumns.add("CREATEDDATE");
queryColumns.add("EXPIRATIONDATE");
queryColumns.add("PRIORITY");
queryColumns.add("TASKNUMBER");
queryColumns.add("OUTCOME");
return queryColumns;
}
public static void main(String[] args) throws Exception {
com.oracle.samples.worklist.IHumanTaskService svc = new
 TaskServiceImpl("bpeladmin", "welcome1", "jazn.com", "jstein");
List<Task> tasks = svc.getTaskList();
System.out.println("Task Size = " + tasks.size());
String outcome = "APPROVE";
for (Task task: tasks){
System.out.println("*** Begin of Task ***");
System.out.println(task.getTitle() + ", " + task.getPriority() + "," +
 task.getCreator() + "," + task.getSystemAttributes().getTaskId() + "," +
 task.getSystemAttributes().getTaskNumber() + "," +
 task.getSystemAttributes().getOutcome() + ","  +
 task.getSystemAttributes().getCreatedDate().getTime() + "," +
 task.getPriority());
task = svc.getTaskDetail( task.getSystemAttributes().getTaskId());
TaskUtil taskUtil = new TaskUtil(task);
taskUtil.printTaskContents();

Map<String, String> map = svc.getOutComes(task);
if (map != null) {
for (String key: map.values()) {
System.out.println("OutCome =" + map.get(key));
if (outcome == null) outcome = map.get(key);
}
}

System.out.println("*** End of Task ***");
}


// Update Task OutCome of Last Task
if (outcome != null) {
Task task = tasks.get(tasks.size()-1);
//TODO Update Payload
System.out.println("Task " + task.getSystemAttributes().getTaskNumber() + " will
 be " + outcome);
svc.updateTaskOutCome(task.getSystemAttributes().getTaskId(), outcome);
}

}
}
```

Task Utility that Prints Payload Contents

The task utility for printing the payload contents is shown below.

```
package com.oracle.samples.util;

import java.io.IOException;

import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import oracle.bpel.services.workflow.task.model.Task;
import sqlj.runtime.profile.SerializedProfile;
public class TaskUtil {
private Task task;

public TaskUtil(Task _task) {
this.task = _task;
}

public Element getRootElement() {
return (Element)this.task.getPayload().getContent().get(0);
}

public void printTaskContents() throws IOException {
serialize(getRootElement());
}

private void serialize(Node node) throws IOException {
switch (node.getNodeType()) {
case Node.ELEMENT_NODE:
System.out.print("Node :: " + node.getNodeName() + ", Local Name

::" + node.getLocalName() + "=");
NamedNodeMap attributes = node.getAttributes();
NodeList children = node.getChildNodes();
if (children != null) {
for (int i = 0; i < children.getLength(); i++) {
serialize(children.item(i));
}
}
break;
case Node.TEXT_NODE:
System.out.println(node.getNodeValue());
break;
}

}
```

Workflow Modeling

This section describes workflow modeling best practices.

This section contains the following topics:

- Routing
- Chaining
- Callbacks
- Escalations
- Notifications
- Resource Bundles
- Attachments

Routing

Table 4–1 lists several task routing capabilities and refers you for additional details to the appropriate section in Chapter 15, "Oracle BPEL Process Manager Workflow Services" of *Oracle BPEL Process Manager Developer's Guide*.

Table 4–1 Routing

Best Capabilities	See Section...
You can perform dynamic assignments based on task content.	"Assigning Task Participants"

Table 4–1 (Cont.) Routing

Best Capabilities	See Section...
You can mix and match patterns (participant types). For example, you can add a single pattern followed by a parallel pattern. Similarly, you can have expressions associated to skip a particular assignee. You can also specify abrupt termination conditions.	▪ For skipping a particular assignee, see "Bypassing a Task Participant" ▪ For specifying abrupt termination conditions, see "Abruptly Completing a Condition"
You can enable adhoc routing at a global level or a participant level	▪ For global levels, see "Allowing All Participants to Invite Other Participants" ▪ For participant levels, see "Inviting Additional Participants to a Task"

Chaining

In addition to using complex routing patterns, you can also chain multiple tasks that have intervening BPEL activities. For example, assume you have a procurement process that goes for management approval first (task 1), then there are many BPEL activities, and then another human task (task 2) for the procurement group. You can link history, comments, and attachments of task 1 and task 2 so that you get a complete end-to-end approval and comment history. This is done by chaining the subsequent task to the previous task.

> **See Also:** Section "Including the Task History of Other Human Tasks" of chapter 15, "Oracle BPEL Process Manager Workflow Services" of *Oracle BPEL Process Manager Developer's Guide* for details about chaining tasks

Callbacks

There are two types of callbacks.

1. The BPEL process is generally called back only when all approvers have acted on the task and the task is complete. However, you can have finer grained callbacks to the BPEL process when the task status changes. For this, there are two capabilities, as shown in the following table. References are provided for additional details to the section in Chapter 15, "Oracle BPEL Process Manager Workflow Services" of *Oracle BPEL Process Manager Developer's Guide*.

Callback Capabilities	See Section...
You can enable callbacks in the human task activity. This ensures that the generated BPEL scope has onMessage handlers for the callbacks. This capability is accessible from the **Advanced** tab of the Human Task window.	"Allowing Task and Routing Customizations in BPEL Callbacks"
You can enable callbacks in the task metadata (in the advanced section). This capability is accessible from the **Advanced** section of the Human Task Editor.	"BPEL Callbacks"

2. You can use Java callbacks for task status changes. This is configured in the **Advanced** section of the Human Task Editor. See section "BPEL Callbacks" for details.

Escalations

Escalations can be manual (through the worklist or by using the API) or automatic (based on task expiration). Table 4–2 describes these escalations and refers you for additional details to the appropriate section in Chapter 15, "Oracle BPEL Process Manager Workflow Services" of *Oracle BPEL Process Manager Developer's Guide*.

Table 4–2 Escalations

Escalation	See
For both manual and automatic escalations, the default behavior is to escalate to the assignee's manager. If you want to configure that behavior to route to another user, then you can implement an alternate escalation function.	"Specifying Escalation Rules" for details about implementing an alternate escalation function
For automatic escalations, you can configure how many times to perform the escalation. For example, assume it is escalated from user A to A's manager (B). If B does not act on it and the task expires, then it can be escalated further automatically to B's manager. You can configure how many times the task is escalated and to which level.	"Escalating, Renewing, or Ending the Task" for details on configuring how many times the task is escalated and to which level

Notifications

Notifications can be sent to various task stakeholders (an assignee, all participants, an owner, and so on) as the task status changes. You configure notifications in the Human Task Editor. By default, the task form is also sent with the notification.

You can mark the task as actionable. In this case, the user can perform actions (such as approve or reject) through e-mail. Additionally, you can also add comments and attachments through the reply e-mail. This requires the outgoing and incoming e-mail server configuration to be configured in the `ns_emails.xml` file.

Additionally, you can also configure the task to have reminders sent out to task assignees before task expiration or after assignment.

See Also: The following sections of Chapter 15, "Oracle BPEL Process Manager Workflow Services" of *Oracle BPEL Process Manager Developer's Guide* for additional details:

- For notifications, see section "Specifying Participant Notification Preferences"
- For marking tasks as actionable, see section "Sending Actionable E-mails"
- For reminders, see section "Sending Reminders"

Resource Bundles

You can configure the resource bundle to be used for task notification content. Additionally, you must use the corresponding XPath functions to retrieve the translation strings from these bundles.

See Also: section "Specifying Multilingual Settings" of Chapter 15, "Oracle BPEL Process Manager Workflow Services" of *Oracle BPEL Process Manager Developer's Guide* for details about configuring the resource bundle

Attachments

Tasks can have one or more attachments. Attachments can be added when initiating the task from BPEL. Attachments can also be added from the worklist. You can also configure whether or not to send attachments along with e-mail notifications.

5 Oracle B2B

This chapter describes best practices for Oracle B2B.

This chapter contains the following topics:

- E-Commerce
- Oracle B2B
- Interoperability
- Errors Diagnostics

E-Commerce

This section provides an overview of E-Commerce.

This section contains the following topics:

- Vision
- Misconceptions
- Concepts: Components
- Concepts: Acknowledgments Types

Vision

The success of any value chain is that it is driven by business requirements, and not technology.

- The focus must be business process management.
- It is an integral component of an enterprise's integration strategy.
- E-Commerce is the entire business process, not the gateway
- The gateway software is becoming a commodity.

The barriers between A2A, C2A, and B2B integration are disappearing. You have endpoints and require business process integration. You must perform the following:

- Orchestrate processes
- Mitigate errors
- Translate and transform data
- Address security, compliance, visibility, and management issues

Misconceptions

B2B is the complete end-to-end process.

- IT considers B2B as gateway software.
- Implementers consider B2B as the complete end-to-end process.
- Implementers do not distinguish between the components.
- If it's a B2B implementation, it is considered a B2B problem.
- It is a suite of components, not a single tool.

Concepts: Components

- A transaction set is a single document sent over the Internet.
- There are six components. The concepts are no more difficult then sending a package through DHL, FedEx, UPS, or the US Postal Service. Table 5–1 describes these components.

Table 5–1 Components

Layer	Question to Ask	Shipping Analogy	E-Commerce Protocol
Document	What is the item?	Cell phone	EDI, HL7, HIPAA, UCCnet, OAG, UBL, cXML, xCBL, RosettaNet
Packaging	How is the item packaged?	Box, bubble wrap	MIME, SMIME, XMLDSig, XMLEncrypt
Transport	How is the item sent and received?	Truck, ship, plane	HTTPs, file, FTPs, TCP/IP, SMTP, MLLP
Messaging system	Who is the carrier?	DHL, FedEx, UPS, USPS	RNIF, AS1, AS2, AS3, ebMS
	What carrier services are required?	Requirements: ■ Signed receipt ■ Overnight/next day ■ Delivery attempts	Requirements: ■ Nonrepudiation ■ Time to acknowledge/respond ■ Retry counts
Profile	What are the Trading partner's capabilities?	What are the sender's /receiver's capabilities?	Trading partner profile
Agreement	What did we agree on?	What did we agree on?	Trading partner agreement

Concepts: Acknowledgments Types

- There are five types of acknowledgments.
- They can be accumulative and vary by protocol.

Table 5–2 describes the acknowledgments.

Table 5–2 Acknowledgments

Layer	Shipping Analogy	Protocol	Acknowledgment Description
Transport	Destination exists	HTTP	Codes: 200/400
Messaging system	Item arrived	RNIF	Receipt acknowledgment
		AS1, AS2, AS3	Message disposition notification (MDN)
		ebMS	Receipt acknowledgment
		Web services	WS-ReliableMessage sequencing
Functional	Item is not damaged	RN ACK, EDI 997 / CONTRL, HL7 ACK, OAG Confirm BOD	The structure and codes are valid
Business	A related item was shipped back	RosettaNet	A business response
Process	Item received was not processed	RosettaNet	PIP0A1 Notification of Failure (NoF)

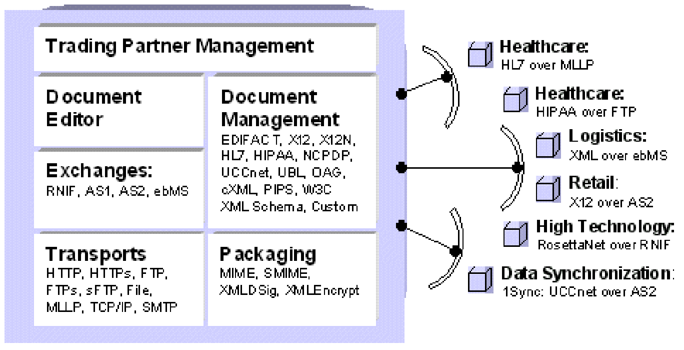
Oracle B2B

Oracle B2B provides the following features:

- Document management
- Trading partner management
- Extensive B2B protocol support
- Secure and reliable message exchange
- Complete end-to-end processes enabled with all assets

Figure 5–1 provides an overview.

Figure 5–1 Oracle B2B



How Does Oracle B2B Fit into Oracle SOA Suite?

E-Commerce represents the complete end-to-end process, from the wire to the application. Oracle B2B is the gateway software and only one component of Oracle SOA Suite for enabling the implementation of business processes.

Oracle SOA Suite components perform the following tasks:

- Orchestrate processes
- Mitigate errors
- Define a canonical standard
- Translate and transform data
- Address security, compliance, visibility, and management

Table 5–3 describes the Oracle SOA Suite components and their responsibilities:

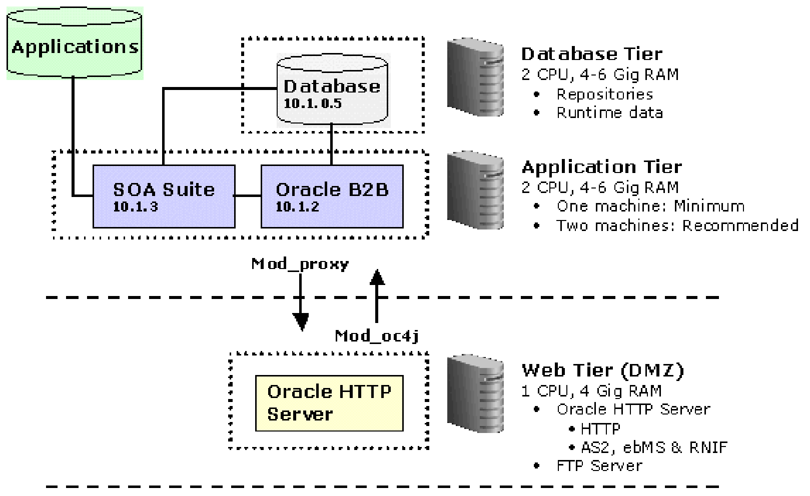
Table 5–3 Oracle SOA Suite Components

SOA Component	Description
Oracle B2B	Trading partner management
Oracle BPEL Process Manager	■ Orchestrate processes ■ Human workflow ■ Business rules
Oracle Web Services Manager	Manage and govern services
Oracle Business Activity Monitoring (BAM)	Monitor and optimize
Oracle Enterprise Service Bus (ESB)	Provide connectivity, transformation, code conversion, and routing

Typical Topology

Figure 5–2 describes a typical Oracle B2B topology.

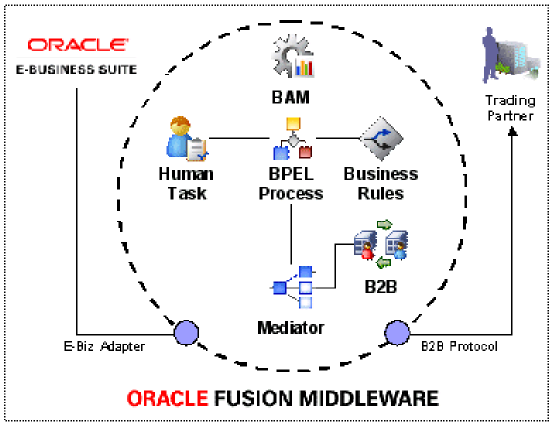
Figure 5–2 Oracle B2B Topology



Use Case: Outbound Purchase Order

Figure 5–3 shows an outbound purchase order use case.

Figure 5–3 Outbound Purchase Order



1. E-Business Suite (eBiz)
   a. Purchase order process is initiated
2. Oracle ESB
   a. Receives purchase order from E-Business Suite
   b. Validates and translates to XML, transforms to target (XSLT), and converts codes
   c. Routes message to Oracle BPEL Process Manager
3. Oracle BPEL Process Manager
   a. Receives purchase order
   b. Executes business processes
   c. Human workflow
   d. Business rules
   e. Error handing
   f. Sends purchase order to Oracle ESB
4. Oracle ESB
   a. Receives purchase order
   b. Validates, transforms to target (XSLT), and converts codes
   c. Routes message to Oracle B2B
5. Oracle B2B
   a. Receives purchase order
   b. Translates to EDI native format
   c. Manages interaction with trading partner
6. Oracle BAM
   a. Monitors end-to-end process

Interoperability

This section describes constructs for Oracle B2B and Oracle ESB/Oracle BPEL Process Manager interoperability.

This section contains the following topics:

- Routing ID/Consumer
- IP_MESSAGE_TYPE
- Adapter Service
- Vertical Required Data

Routing ID/Consumer

Associating an inbound document to a consumer

- Multiconsumer — routing ID/consumer
  - Inbound — matches document in the queue to a process
  - Outbound — fixed: B2BUSER
- The routing ID is set in Oracle B2B — document definition
- Adapter Service — creates all constructs in Oracle BPEL Process Manager/Oracle ESB for deployed Oracle B2B configurations
- AQ Adapter — constructs are manually created in Oracle BPEL Process Manager/Oracle ESB

Table 5–4 describes the interactions.

Table 5–4 Routing ID/Consumer

Queues	Queue Name	Oracle B2B	ESB / BPEL Process Manager	
		Schema =		B2B
Inbound	`IP_IN_QUEUE`	Document routing ID =	Consumer =	X12_4010_850_PO
Outbound	`IP_OUT_QUEUE`	n/a	Consumer =	B2BUSER

IP_MESSAGE_TYPE

Identify the document, partner, and agreement:

- Header — `IP_MESSAGE_TYPE`
- Inbound — identifies partner and document information
- Outbound — enables Oracle B2B to identify the agreement

Table 5–5 describes the `IP_MESSAGE_TYPE` fields.

Table 5–5 IP_MESSAGE_TYPE Fields

Field	Description
`MSG_ID`	Unique identifier
`INREPLYTO_MSG_ID`	Original `MSG_ID`
`FROM_PARTY`	Partner sending the message
`TO_PARTY`	Partner receiving the message

Table 5–5 (Cont.) IP_MESSAGE_TYPE Fields

Field	Description
`ACTION_NAME`	Action that defines the message
`DOCTYPE_NAME`	Message type
`DOCTYPE_REVISION`	Message revision
`MSG_TYPE`	Message type
`PAYLOAD`	Optional
`ATTACHMENT`	Optional

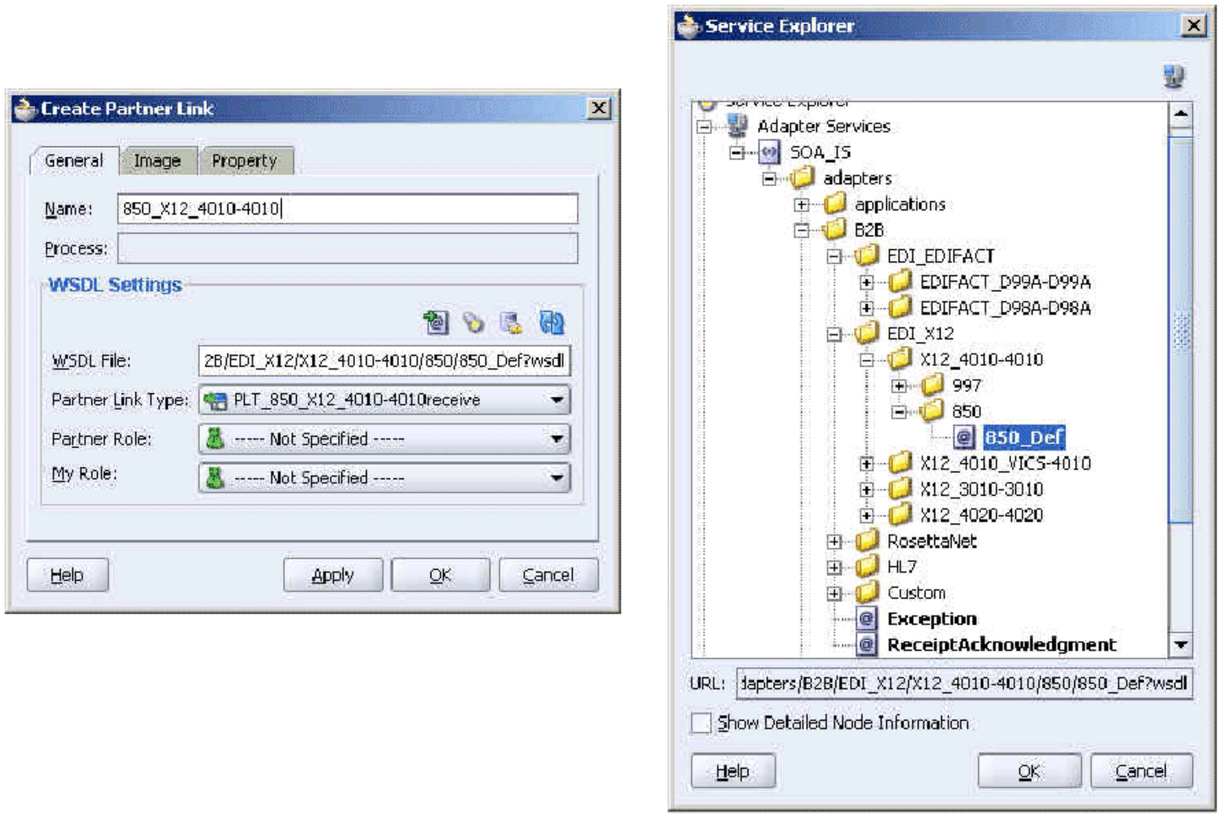
Adapter Service

Creating these constructs:

- Enables the user to browse the Oracle B2B deployed configurations
- Uses Oracle AQ as the default mechanism

Figure 5–4 shows where you select the Oracle B2B deployed configurations in Oracle JDeveloper.

Figure 5–4 Adapter Service



Name: 850_X12_4010-4010

Process:

WSDL File: 2B/EDI_X12/X12_4010-4010/850/850_Def?wsdl

Partner Link Type: PLT_850_X12_4010-4010receive

Partner Role: ----- Not Specified -----

My Role: ----- Not Specified -----

```
Adapter Services
  SOA_IS
    adapters
      applications
      B2B
        EDI_EDIFACT
          EDIFACT_D99A-D99A
          EDIFACT_D98A-D98A
        EDI_X12
          X12_4010-4010
            997
            850
              850_Def
          X12_4010_VICS-4010
          X12_3010-3010
          X12_4020-4020
        RosettaNet
        HL7
        Custom
        Exception
        ReceiptAcknowledgment
```

URL: dapters/B2B/EDI_X12/X12_4010-4010/850/850_Def?wsdl

Vertical Required Data

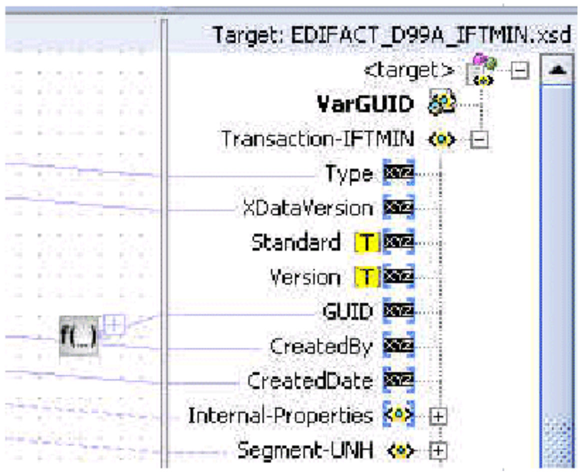
EDI, NCPDP, and HL7 documents:

- XSD generated by document editor

- Required data:
  - Standard — X12
  - Version — V4010
  - GUID — {12345678-1234-1234-1234-123456789012}
  - Inbound — n/a
  - Outbound — enables translator document identification
- Envelope — internal properties (X12, EDIFACT, HL7)
  - Inbound — provides envelope information for processing
  - Outbound — overrides Oracle B2B envelope defaults

Figure 5–5 shows the transformation mapping.

Figure 5–5 Transformation Mapping



Errors Diagnostics

This section describes how to diagnose Oracle B2B errors.

This section contains the following topics:

- Trading Partner Identification
- Typical Oracle B2B Errors
- Inbound Direction
- Outbound Direction
- General Check List
- Typical Errors

Trading Partner Identification

Oracle B2B identifies a configuration in this order:

1. Trading partner
2. Document
3. Agreement

4. Configuration

Typical Oracle B2B Errors

- Document protocol error
- Validation errors
- Unable to identify trading partner
- Unable to identify agreement
- Connection errors

Inbound Direction

Oracle B2B to Oracle ESB/Oracle BPEL Process Manager:

- If the document is on `IP_IN_QUEUE`:
  - Oracle B2B is done
- If Oracle ESB/Oracle BPEL Process Manager:
  - Does not dequeue:
    - Check the document definition (routing ID)
    - Check the partner link (consumer)
  - Dequeues, but does not process:
    - Check for the deployed process
    - Check the Oracle ESB Control/Oracle BPEL Control
    - Check the Oracle BPEL Control **Perform Manual Recovery** option
    - Check the Oracle ESB/Oracle BPEL Process Manager log files

Outbound Direction

Oracle ESB/Oracle BPEL Process Manager to Oracle B2B:

- If Oracle B2B reports errors:
  - Document protocol error
  - Unable to identify trading partner
  - Unable to identify agreement
- Then check the following:
  - Oracle AQ:
    - Consumer — B2BUSER
  - `IP_MESSAGE_TYPE`:
    - `MSG_ID`, `FROM_PARTY`, `TO_PARTY`, `DOCTYPE_NAME`, `DOCTYPE_REVISION`, `MSG_TYPE`
  - Vertical required data (X12, EDFACT, NCPDP, HL7)
    - Standard/version/graphical user interface
  - Internal properties (X12, EDFACT, NCPDP, HL7)

  - Envelope overrides

General Check List

- Assume *all* names and parameters are case sensitive.
- The naming of objects may be critical.
  - These are used for inbound validation and outbound generation of headers. (business actions, document types, document type revisions, agreement IDs, and document protocol parameters)
- Verify use of the acknowledgment mode (none/async/sync)
- Verify use of acknowledgments:
  - Messaging system — AS1, AS2, ebMS, and RNIF
  - Functional — EDI and HL7
- Errors may be caused by the backend system or the partner
- Examine the `b2b.log`:
  - System log — `Oracle_Home\ip\log\b2b\b2b.log`
  - Configuration file — `Oracle_Home\ip\config\tip.properties`

Oracle Technology Network Documentation

See the following technical notes on the Oracle Technology Network (OTN):

- `IP_MESSAGE_TYPE` —
  http://www.oracle.com/technology/products/integration/b2b/pdf//B2B_TN_008_IP_MESSAGE_TYPE.pdf
- Oracle B2B configuration file properties and parameters —
  http://www.oracle.com/technology/products/integration/b2b/pdf/B2B_TN_021_TIP_Properties.pdf
- Exception handling —
  http://www.oracle.com/technology/products/integration/b2b/pdf/B2B_TN_007_Exception_Handling.pdf
- Technical tips —
  http://www.oracle.com/technology/products/integration/b2b/pdf/B2B_Technical_Tips.pdf

Additional OTN documentation:

- Oracle Fusion Middleware —
  http://www.oracle.com/technology/products/middleware/index.html
- Service-oriented architecture (SCA) —
  http://www.oracle.com/technologies/soa/index.html
- Oracle B2B —
  http://www.oracle.com/technology/products/integration/b2b/index.html
- Oracle B2B/SOA installation best practices —
  http://www.oracle.com/technology/products/integration/b2b/pdf/B2B_SOA_Suite_Installation.pdf

Typical Errors

Table 5–6 shows several typical Oracle B2B errors.

Table 5–6 Typical Errors

Type of Error	Example error message	Most Probable Cause Check (Case, Mistyped, ...)
Connection	`Transport error: [IPT_ HttpSendConnectionRefused] HTTP connection is refused. Connection refused: connect`	Outbound: ▪ Delivery channels ▪ Is the trading partner up
Document protocol identification	`Document protocol identification error`	Inbound: ▪ Identification xPath, positional ▪ Characters in file Outbound: ▪ `IP_MESSAGE_TYPE` ▪ Vertical required data
Document validation	`General Validation Error`	Inbound (EDI, HIPAA, HL7) ▪ Document revision protocol parameters — envelope Payload (if `validation=True`) Inbound/outbound: ▪ XML schema ▪ ECS file (EDI, HIPAA, HL7)
Agreement/partner identification	`Trading partner agreement not found for the given input values: From party [NAME-ROLE] "GlobalChips", To party [NAME-ROLE] "Acme", also verify agreement effectiveToDate`	Inbound /Outbound: ▪ Trading partner ID ▪ Trading partner name

6 Oracle Business Activity Monitoring

This chapter provides an overview of available Oracle Business Activity Monitoring (BAM) technical notes.

This chapter contains the following topics:

- How Do I Configure Oracle BAM with the AQ Adapter?
- How Do I Configure Oracle BAM with IBM WebSphere MQ Messages?
- How Do I Configure Oracle BAM with OC4J Version 10.1.3?
- How Do I Design BPEL Process Sensors to Send Events to Oracle BAM?
- How Do I Design BPEL Process Sensors for Oracle BAM Sensor Actions?
- How Do I Design and Use BPEL JMS Sensors to Publish Events to Oracle BAM?
- How Do I Configure Oracle BAM to Read Data from External Data Sources?
- How Do I Read External Tables into Oracle BAM Using Enterprise Link Plans?
- How Do I Configure Oracle Enterprise Manager with Oracle BAM?
- How Do I Beautify Reports with Good Color Schemes?
- How Do I Build a Simple Report?
- How Do I Create a Custom HTML Body in a Report for Custom Display?
- What are the Oracle BAM View Types?
- How Do I Design a Report Showing the Ratio of Two Aggregated Values?
- How Do I Design Predictive Alerts?
- How Do I Call External Web Services through Oracle BAM?
- How Do I Call External URLs?
- How Do I Collect and Parse Incoming Data?
- How Do I Configure a Plan to Run Continuously?
- What are the Deployment Recommendations?
- How Do I Configure Oracle BAM for LDAP Authentication and Authorization?
- How Do I Migrate Oracle BAM to Different Environments?
- What are the Best Practices for Oracle BAM Repository Maintenance?
- How Do I Use Oracle BAM without Enterprise Link?

See Also: The following URL for frequently asked questions about Oracle BAM:

http://www.oracle.com/technology/products/integration/bam/10.1.3/htdocs/bam_1013_faq.html

How Do I Configure Oracle BAM with the AQ Adapter?

Technical Note Objectives

- Understand the Oracle BAM configuration to define the AQ message source
- Configure Oracle BAM to connect to the AQ message source
- Define triggers on database tables to push data into AQ (JMS bus)
- Connect and collect data from the Oracle AQ source and populate Oracle BAM data objects
- Similar concepts can be used to push incremental data from other databases using triggers or changing logs and sending the XML message (changed data) through a JMS bus to Oracle BAM.

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_AQ_Configuration.pdf

How Do I Configure Oracle BAM with IBM WebSphere MQ Messages?

Technical Note Objectives

- Understand Oracle BAM message sources and configuration parameters for an IBM Websphere MQ client
- Configure the Oracle enterprise message source
- Verify connection settings for the IBM MQ message source

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_ConfigurationForWebSphereMQClient.pdf

How Do I Configure Oracle BAM with OC4J Version 10.1.3?

Technical Note Objectives

- Understand Oracle BAM message sources and configuration parameters
- Verify connection settings for the OJMS message source
- Configure the Oracle EMS message source type

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNo tes/TechNote_BAM_Configurefor1013OC4J.pdf

How Do I Design BPEL Process Sensors to Send Events to Oracle BAM?

Technical Note Objectives

- Design and build a real time BPEL process monitoring dashboard
- Understand BPEL sensor design and sensor messages
- Correlate BPEL sensor data from different instances and different activities from the same instance and store the data in Oracle BAM
- Design Oracle BAM dashboards for process monitoring through put, response times, process states, and performance figures

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_RealTimeBPELMonitoring.pdf

How Do I Design BPEL Process Sensors for Oracle BAM Sensor Actions?

Technical Note Objectives

- Understand Oracle BPEL process design for Oracle BAM sensor actions
- Configure Oracle BPEL process activities with Oracle BAM sensor definitions
- Configure and connect Oracle JDeveloper to Oracle BAM Server
- Connect and browse Oracle BAM ADC data objects
- Define an Oracle BPEL Process Manager - Oracle BAM sensor action to populate Oracle BAM ADC data objects, including insert and update operations, and mapping BPEL attributes to Oracle BAM ADC data object fields using XSLT translation

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BPELSensorsForBAM.pdf

How Do I Design and Use BPEL JMS Sensors to Publish Events to Oracle BAM?

Technical Note Objectives

- Understand sending BPEL sensor events to Oracle BAM
- Understand Oracle BAM message sources
- Verify connection settings for an OJMS message source
- Configure the Oracle EMS message source type
- Understand Design Studio and Enterprise Link to collect data from a JMS source

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_BPELJMSSensors.pdf

How Do I Configure Oracle BAM to Read Data from External Data Sources?

Technical Note Objectives

- To read static data into ADC objects
- This is recommended only to get static data into ADC. This is not recommended to get real-time data into ADC.

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_ExternalDataSource.pdf

How Do I Read External Tables into Oracle BAM Using Enterprise Link Plans?

Technical Note Objectives

- To read database table data into ADC objects
- This is not recommended to get real-time data into ADC
- This can be used only for the periodic, controlled refresh of data

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_ReadingExternalTableintoADC.pdf

How Do I Configure Oracle Enterprise Manager with Oracle BAM?

Technical Note Objective

Integrate Oracle Enterprise Manager with Oracle BAM using the EM data exchange connector.

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/em_de.pdf

How Do I Beautify Reports with Good Color Schemes?

Technical Note Objective

Beautify the reports for better visualization and end user perception.

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_BeautifyingReports.pdf

How Do I Build a Simple Report?

Technical Note Objective

Build a simple report with several selected view types

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_BuildingSimpleReport.pdf

How Do I Create a Custom HTML Body in a Report for Custom Display?

Technical Note Objective

Design reports with custom HTML strings

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_CustomHTMLReport.pdf

What are the Oracle BAM View Types?

Technical Note Objective

Illustrate the various view types supported in the reports and provide a brief description of these types

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_ViewTypes.pdf

How Do I Design a Report Showing the Ratio of Two Aggregated Values?

Technical Note Objective

To create an Oracle BAM report that shows the ratio of two aggregated values summarized hourly

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_RatioOfTwoAggregations.pdf

How Do I Design Predictive Alerts?

Technical Note Objectives

- Demonstrate the predictive alerting capability of Oracle BAM
- Design alert architecture to trigger an action, if events do not occur

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_DesigningPredictiveAlerts.pdf

How Do I Call External Web Services through Oracle BAM?

Technical Note Objective

Configure the Oracle BAM event engine and alert settings to directly call (invoke) an external Web service

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_CallingExternal_WebServices.pdf

How Do I Call External URLs?

Technical Note Objectives

- Design calling static and dynamic URLs from the Oracle BAM dashboard
- Oracle BPEL Server URLs are used in this document for examples and illustrations.

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_CallingExternalURL.pdf

How Do I Collect and Parse Incoming Data?

Technical Note Objective

Define the enterprise message source for receiving data from a JMS bus

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_CollectAndParseIncomingData.pdf

How Do I Configure a Plan to Run Continuously?

Technical Note Objective

Configure a plan to run continuously and collect data in real time

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_ConfigurePlanForRealTime.pdf

What are the Deployment Recommendations?

Technical Note Objective

Develop an estimate of the system resources required to deliver performance objectives that meet a forecasted level of business computing activities and objectives

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_Deployment_Recommendations.pdf

How Do I Configure Oracle BAM for LDAP Authentication and Authorization?

Technical Note Objective

Explains how to authenticate Oracle BAM against the Sun Java System Directory Server

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_LDAPConfiguration.pdf

How Do I Migrate Oracle BAM to Different Environments?

Technical Note Objective

Explain how to migrate from one Oracle BAM platform to another

Technical Note Location

See the following technical note for details:

http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_Migration.pdf

What are the Best Practices for Oracle BAM Repository Maintenance?

Technical Note Objective

Understand the Oracle BAM repository components for maintenance and backup

Technical Note Location

See the following technical note for details:

`http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_RepositoryMaintenance.pdf`

How Do I Use Oracle BAM without Enterprise Link?

Technical Note Objective

Explain how to use Oracle BAM when the Enterprise Link is not installed or available

Technical Note Location

See the following technical note for details:

`http://www.oracle.com/technology/products/integration/bam/10.1.3/TechNotes/TechNote_BAM_BAM10131_WithoutEL.pdf`

7 Oracle Data Integrator

This chapter describes best practices for using Oracle Data Integrator (ODI).

This chapter contains the following topics:

- Repositories
- Agents
- JDBC Drivers
- Topology
- Designer
- Operator
- Operations

Repositories

This section describes repository best practices.

This section contains the following topics:

- Number of Database Schemas for the Repositories
- ID of the Work Repository
- Name of the Work Repository
- Number of Work Repositories
- Types of Repositories (Development versus Execution)
- Promotion from Development to QA and from QA to Production
- Database to Use for the Repositories
- Remote Connectivity to the Repositories from the Graphical User Interface

Number of Database Schemas for the Repositories

There should be one dedicated schema for the master repository, and another one for each work repository. The master and work repositories do not have to be on the same server.

Rationale

This simplifies backup procedures and repository management.

ID of the Work Repository

When creating a new work repository, make sure that its ID is *unique*.

Rationale

This ID is used by ODI as part of each object's ID. By ensuring that each repository has a unique ID, you ensure that all objects created have a universally unique ID throughout the installation.

Name of the Work Repository

Do not use spaces for the name of the work repository.

Rationale

This name is used in many places, including configuration files. Avoiding spaces makes it easier to maintain these files.

Number of Work Repositories

Oracle recommends at least three and ideally four work repositories.

Rationale

You need a work repository for your development work, a second repository for testing and qualification, and a third repository for production. This enables developers to work on the next release while the current release is in production. The fourth repository can be used to restore the source code of production jobs if these jobs need to be fixed (a hot fix). Again, this allows developers to have a more advanced version of the code in their development repository.

Types of Repositories (Development versus Execution)

Development and hot fix repositories are development repositories. QA/testing and production repositories are execution repositories.

Rationale

You need access to the source code only in repositories where developers are allowed to make modifications. Removing the source code from QA and production guarantees that no last minute modification inadvertently finds its way into the production system.

Promotion from Development to QA and from QA to Production

Create a solution that saves together the following:

- Scenarios
- Source code for the project
- Metadata associated with the project

Each object in the solution is versioned.

In QA and production environments, restore the scenario only.

In hot fix environments, restore the solution to retrieve the associated source code (project and metadata).

Rationale

Saving solutions guarantees the consistency between the code of the transformations and the associated version of the metadata.

A scenario should never be deployed without a solution first being created to ensure that the source code for that specific scenario can be retrieved at any time.

Database to Use for the Repositories

Oracle is the recommended database.

The following databases are also supported:

- Microsoft SQL Server
- DB2/400
- DB2/UDB
- Sybase ASE (row level lock only)
- Informix
- Hypersonic SQL

Remote Connectivity to the Repositories from the Graphical User Interface

For remote connectivity to the repositories, consider using remote terminal services.

Rationale

The graphical user interface is quite communicative with the repositories. Installing the graphical user interface on a remote host results in poor performance for the developer. Remote terminal services allow for the best possible performance between the graphical user interface and the repository. In addition, the display on the remote host is much faster.

Agents

This section describes agent best practices.

This section contains the following topics:

- Parameters for the Agent
- Starting the Agent
- Agent Location
- Sizing for the Agent
- Agent Overload
- Agent Connectivity to the Repository

Parameters for the Agent

Always start agents with the parameter `-NAME=` and set the agent's name.

Rationale

This guarantees that the agent name appears in the logs. It is also a requirement if you are using load balancing.

Starting the Agent

Install the agent as a service on Windows hosts. Start the agent at boot time on UNIX. For Windows, the `agentservice.bat` file installs the agent as a service.

For UNIX, see your UNIX administrator.

Rationale

The agent should always be up and running if you want to use it. It must start when the system starts.

Agent Location

If possible, place the agent on the target server. Then, add agents as needed on other systems (for load balancing, to access local flat files, to take advantage of database utilities, and so on).

Rationale

All sources systems send data to the target system. There has to be a physical path from all sources to the target. The target system is the most central location for the agent.

Sizing for the Agent

A practical maximum agent sizing value is 10 concurrent processes per agent.

Rationale

- Take advantage of the distributed architecture to use additional hosts.
- Take advantage of the load balancing capabilities of ODI to use the most available system at any given point in time.

Agent Overload

Agents should never be overloaded, as they do very little. However, when multiple agents are running on the same platform, you must be aware of the type of resources used by each agent:

- Memory — Agents tend to use more memory if you use JDBC connectivity to move data from one system to the other. Each agent uses its own memory space. Ultimately, agents might be competing for memory space. The amount of memory that each agent uses can be configured in the `odiparams.bat` file (on Windows) or the `odiparams.sh` file (on Unix).
- Bandwidth — All agents running on the same host and processing data in parallel potentially compete for bandwidth. Keep this in mind in your choices for agent locations.
- I/O — Multiple agents running on the same host processing large files in parallel are also competing for I/O resources. Consider using multiple platforms in this case.

Agent Connectivity to the Repository

Listener agents should connect to *only* one repository.

Rationale

The process that invokes the agent (designer, operator, or some external scheduler) provides the parameters to connect to the repository. In theory, it is possible to have an agent connect to multiple repositories.

In practice, this is not recommended as the same agent is processing data that most likely is not supposed to be in the same context. Beyond potential issues in the agent itself (potentially updating the wrong repository at times), security considerations should forbid this practice.

JDBC Drivers

This section describes JDBC driver best practices.

This section contains the following topics:

- Version of the JDBC Drivers
- JDBC Driver Types

Version of the JDBC Drivers

Always use the version of the JDBC driver that matches the version of the database.

Rationale

Database vendors update their JDBC drivers with each new release of their databases. Using an old release of the JDBC driver may prevent you from using the latest features of the database.

When using different releases of the same database in the same project, it is usually best to use the latest version of the JDBC driver. Depending on the quality of the drivers provided by the different database vendors, testing might be required to identify the best possible release of the driver.

JDBC Driver Types

JDBC drivers type 4 are always preferred when available (for instance, lite versus OCI for Oracle).

Rationale

JDBC drivers type 4 do not have to go through the client layer of the database. They make a direct connection to the database listener. As such, they are much faster. When type 4 drivers are not available, type 2 and 3 drivers are the next choice (going through a client layer with type 2 or going through a gateway with type 3). Type 1 drivers are the slowest, as they go through an ODBC connection.

Topology

This section describes topology best practices.

This section contains the following topics:

- Definition of the Data Servers
- Connectivity to the Databases
- User Name to Connect to a Database
- Default Schemas

- Creation of a New Technology
- Test Agent Connectivity

Definition of the Data Servers

One data server per database instance (one single login for all schemas).

Rationale

This is a prerequisite for extract, load, and transform (ELT). If we have several data servers, then we have multiple logins as well.

Connectivity to the Databases

When creating a data server, always test the connectivity before creating the schemas. ODI should retrieve the schemas list from the actual database.

Rationale

It is always easier to fix connectivity issues when defining the connectivity parameters rather than moving along and then trying to solve problems that are, in fact, only connectivity issues.

User Name to Connect to a Database

Before defining a new server in ODI, create an ODI user in the database (typically `ODI_TEMP`). Provide as many privileges to this user as needed in their schema.

This user and their schema are used as follows:

- Use this user name to log in to the server.
- Use this user's schema as your work schema for all data schemas on this server.

Rationale

Since the ODI user owns the work schema, it is easier to give the necessary privileges to create and drop staging tables in this schema.

All that is needed after that is the appropriate privileges to read and write data from other source and target schemas.

Default Schemas

When deleting the default schema on a data server, immediately assign a new default data server.

Rationale

Many ODI functions do not work if there is no default schema.

Creation of a New Technology

To create a new technology, duplicate an existing one that is close to the technology to be created.

Rationale

This saves time. This also prevents mistakes.

Test Agent Connectivity

When adding an agent, test the connectivity to the data servers through the agent.

Rationale

This guarantees that the agent can actually connect to the data servers.

Designer

This section describes designer best practices.

This section contains the following topics:

- Markers
- Database Sequence Versus ODI Sequence
- Metadata Project
- Modification of the Knowledge Modules
- Choosing the Appropriate Knowledge Module
- Hardcoded Table Names
- Documentation
- Complex and Reusable Transformations
- Number of Interfaces Per Folder
- Name of the Interfaces
- XML or JMS/XML Synchronization
- Commit Multiple Interfaces Only If the Last One Is Successful
- Variables in a Package
- Contexts in the Interfaces, Knowledge Modules, and Procedures

Markers

- Use progress markers to indicate the level of completion of the different objects.
- Use memos to share notes with other developers on the different objects.
- Use priority markers to indicate which objects have to be developed, investigated, and fixed first.

Rationale

Markers help you with team work by providing a better understanding of the work that remains to be done on the different objects.

Database Sequence Versus ODI Sequence

Use the database sequence whenever possible.

Rationale

ODI provides sequence management, but the sequences can only be properly incremented if the records are processed one by one (row-by-row processing).

To get the best possible performance, ODI always recommends using set processing versus row-by-row processing.

Database sequences allow for set processing.

Metadata Project

Always create a dedicated project for the metadata knowledge modules (journalizing knowledge modules (JKMs), check knowledge modules (CKMs), and service knowledge modules (SKMs)). When using a knowledge module in a model, only use knowledge modules from this metadata project.

Rationale

When generating ODI solutions, this guarantees that dependencies between projects and models do not drag the entire repository into the solution.

Modification of the Knowledge Modules

If a knowledge module delivered with the product is modified, it must be renamed. Changes must be described in the knowledge module description.

Rationale

- End users know that the behavior is different from that of the default knowledge module.
- Maintenance is made easier for the same reason.

Choosing the Appropriate Knowledge Module

To choose the appropriate knowledge module for your interfaces, look at the following elements to make your selection:

- Volume of data (small: JDBC; large: database utilities)
- Technologies
- Available knowledge modules

Do not hesitate and optimize the knowledge modules as needed.

Rationale

The best performance can only be obtained with the knowledge module that best fits each specific environment.

Hardcoded Table Names

There are cases where table names are hardcoded:

- Subqueries
- ODI procedures running queries
- Variable queries

Instead of hardcoding the schema name (and/or owner) of the table, use the `odiRef.getObjectName` substitution method. See *Oracle Data Integrator Substitution Methods Reference* for details on the syntax.

Rationale

As you run your processes in different contexts, the physical schema names (and or owners) may change from host to host. If you let ODI select the physical schema for you, you are ensured of always using a valid name for any given context.

Documentation

Always document models and interfaces in the description field.

Rationale

Beyond the obvious readability of the object, descriptions are gathered in the documentation that ODI generates automatically.

Complex and Reusable Transformations

Whenever you have a complex transformation that is reusable, it is a good practice to make an ODI user function of it.

Rationale

- You write the transformation once only.
- If you have to modify the logic, only modify it once.

Number of Interfaces Per Folder

Try to keep the number of interfaces per folder to a manageable number. Try to create a new folder for each separate action/feature in your project.

Rationale

Version management becomes more difficult as projects and folders grow without control.

Name of the Interfaces

Beyond the naming conventions, think of numbering the interfaces as you create them. For instance:

1. Pop.Customers
2. Pop.Sales

Rationale

This enables you to force the order in which ODI displays the interfaces. When you are ready to put together your package, take the interfaces in the order in which they are listed.

XML or JMS/XML Synchronization

Due to the nature of XML, when you are done loading data into an XML file (or in an enterprise service bus (ESB), using an XML structure), you must issue a `SYNCHRONIZE` command to the XML driver. See the Sunopsis JDBC Driver for XML documentation for more details on this command. It is recommended that you put this command in an ODI procedure.

Rationale

As you work down the hierarchy of the XML file, you add interfaces in your package. By isolating the `SYNCHRONIZE` command in an ODI procedure, you guarantee that the command is always sent after your last interface.

An alternative is to issue this command from within the interface, using an option in the knowledge module. The downside of this alternative is that each time you process a new level down the hierarchy, you have to remove the `SYNCHRONIZE` option from what was previously your last interface, and set the option in the new one. This increases the risk for error.

Commit Multiple Interfaces Only If the Last One Is Successful

If you execute multiple interfaces in a sequence, and only commit after the last interface is successful, always do the commit inside an ODI procedure.

Rationale

The rationale is similar to the XML case described in "XML or JMS/XML Synchronization" on page 7-9. Doing the commit outside of the interfaces guarantees that, if you add additional interfaces in this logic, the commit always happens only after all interfaces are successful. An additional benefit is that anybody looking at the package knows from the procedure that data is not committed within the interfaces without opening any of them.

Variables in a Package

A good practice for variables in a package is to declare the variables.

Rationale

This guarantees that if you use a variable before setting a value or refreshing the value of the variable, it is still instantiated. It is actually mandatory to declare variables whose values are passed as parameters.

Contexts in the Interfaces, Knowledge Modules, and Procedures

Never select or force a context in an interface, knowledge module, or procedure.

Rationale

You want all the elements of your processes to use the current execution context. Forcing a context in any given element forces that element in a context that is not the chosen one. In addition, forcing a context might make debugging more difficult. This is because processes do not necessarily run on the hosts on which you think they run.

Operator

This section describes operator best practices.

Logs Purge

Regularly purge the logs, either manually or through a scheduled job (using the `OdiPurgeLog` tool).

Rationale

Logs are stored in tables. If these tables are growing endlessly, the performance of ODI decreases over time (both to display the logs in the graphical user interface and when the agent generates additional steps).

Operations

This section describes operations best practices.

This section contains the following topics:

- ODI Version Migrations
- Real-Time Scenarios: Definition of a Looping Process

ODI Version Migrations

When migrating ODI from an older release to a newer release, always consider making a database backup of the repositories and migrating the backup.

Rationale

There are several benefits to this approach:

- The old repository is still up and running, and untouched. Production is still live while the repository is being migrated and tested.
- If case issues are identified while testing the new release, there is no need to roll back anything; the old version is still available.
- If the tests of the new release are satisfactory, it is simple to stop the old release and start the new one.

Real-Time Scenarios: Definition of a Looping Process

When defining a process that runs forever, do not design a loop within the scenario. There are two options:

- Define the loop with the scheduler.
- Have the scenario call itself recursively in asynchronous mode.

Rationale

An infinite loop in a scenario generates a scenario that never finishes. This prevents any log purge in the future.

8 Oracle SOA Suite Security

This chapter describes best practices for Oracle SOA Suite security.

This chapter contains the following topic:

- "Integrating with External LDAP Providers"

See Also: *Oracle BPEL Process Manager Administrator's Guide* for additional details about configuring with third-party LDAP servers

Integrating with External LDAP Providers

This section describes how to integrate Oracle BPEL Process Manager users and roles with an external LDAP provider.

Note: This section describes setting up the necessary privileges for the `oc4jadmin` account. The same setup can apply to `bpeladmin` or any user account that requires super-user privileges.

1. Follow the steps documented in section "Creating Necessary Accounts and Granting Necessary Permissions" of the chapter "External LDAP Security Providers" of *Oracle Containers for J2EE Security Guide*. This section discusses steps you must take when using an external LDAP provider to create the necessary accounts and grant necessary permissions.
2. Create the `BPMSystemAdmin` role in the LDAP server.
3. Create the `oc4jadmin` user account in the LDAP server.
4. Add `oc4jadmin` as a member of the `BPMSystemAdmin` role.
5. Go to the *`SOA_ORACLE_HOME`*`/j2ee/home` directory.
6. Specify the following command:

```
java -Xbootclasspath/a:/home/oc4j/bpel/lib/orabpel-boot.jar -jar jazn.jar
 -grantperm oracle.security.jazn.realm.LDAPPrincipal
BPMSystemAdmin com.collaxa.security.ServerPermission server all
```

7. Check the `system-jazn-data.xml` file for the OC4J instance in which the Oracle BPEL Process Manager application is deployed. The following grant statement should be created by the command specified in step 6.

```
<jazn-policy>
   <grant>
      <grantee>
         <principals>
```

```
            <principal>
               <class>oracle.security.jazn.realm.LDAPPrincipal</class>
               <name>BPMSystemAdmin</name>
            </principal>
         </principals>
      </grantee>
      . . .
      <permissions>
         <permission>
            <class>com.collaxa.security.ServerPermission</class>
            <name>server</name>
            <actions>all</actions>
         </permission>
         . . .
      </permissions>
      . . .
   </grant>
   . . .
</jazn-policy>
```

8. Use Oracle Enterprise Manager to configure the third-party LDAP security provider. Security provider configuration occurs for the parent J2EE application `orabpel`, which contains the Oracle BPEL Control Web applications and one other EJB application.
9. Restart OC4J.
10. Test Oracle BPEL Process Manager from Oracle Enterprise Manager.

> **See Also:** *Oracle BPEL Process Manager Administrator's Guide* for additional details about the `BPMSystemAdmin` role

Part II

Oracle SOA Suite Performance

This part describes best practices for Oracle BPEL Process Manager.

This part contains the following chapters:

- Chapter 9, "Best Practices for a JMS-to-Database Scenario"
- Chapter 10, "Oracle BPEL Process Manager Performance"
- Chapter 11, "Oracle Human Workflow Performance"

9

Best Practices for a JMS-to-Database Scenario

This chapter describes the best practices for optimizing the performance of an Oracle BPEL Process Manager application in a Java Message Service (JMS)-to-database scenario. This scenario is derived from an actual proof-of-concept (POC) engagement with a customer.

- Overview
- Requirements
- Functional Design
- Performance Configurations
- Other Recommendations
- Performance Tuning
- Final Numbers

Notes:

- This scenario was performed on Oracle BPEL Process Manager release 10.1.2. However, the concepts described in this chapter are still applicable.
- Proprietary information has been modified to protect the customer's confidentiality. Oracle hopes that you find these best practices adaptable to your own performance configuration scenarios.

Overview

This chapter begins with a description of the functional and performance requirements of the POC in "Requirements" on page 9-2. In "Functional Design" on page 9-2, the optimal logical design and high level architecture to fulfill these requirements are described. "Performance Configurations" on page 9-6 describes the best practices to implement such a design. Guidelines to improve performance are described in "Performance Tuning" on page 9-22. As demonstrated throughout the chapter, some simple tuning can significantly improve performance. "Final Numbers" on page 9-22 documents the actual numbers obtained from performance testing and how these numbers are measured.

Requirements

This section describes functional and performance requirements.

Functional Requirements

The customer has an application that sends JMS messages to a message queue. On average, a message is a few hundred bytes in size. Each of these messages constructs an order list, which contains one to five business orders, averaging two orders per message.

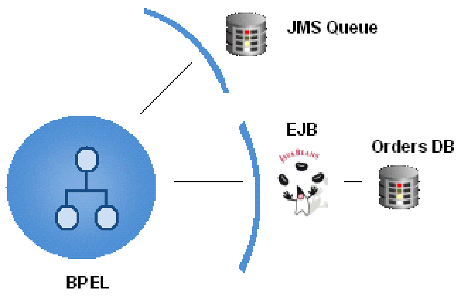
The customer has a session bean (ParseMsgEJB) that parses these messages. Based on the value returned from ParseMsgEJB, a message may be discarded or parsed into an order list. On a general scale, the ratio between the total number of incoming BPEL messages and total number of orders is about 1.7. The ratio between the total number of undiscarded messages and total number of orders is about 2.

The orders are stored in the order table of the destination database. The customer uses entity bean OrderEJB as the logical representation of the rows in the order table. Specifically, one instance of OrderEJB represents one row in the order table. There is a session bean (StoreOrderEJB) that takes an order list and creates (if new) or updates (if pre-existing) OrderEJB.

The application uses Oracle BPEL Process Manager to orchestrate data among the JMS queues and EJBs. In case of parsing errors, it throws an exception.

The functional requirements are shown in Figure 9–1.

Figure 9–1 Functional Requirements



Performance Requirements

The customer measures the throughput by number of orders processed per minute. This number is shown as throughput by orders in Table 9–7 of "Final Numbers" on page 9-22.

The customer expects to see Oracle BPEL Server fully utilize the CPU power of the server computer, and a near linear scalability by adding nodes to a BPEL cluster.

Functional Design

This section describes the functional design of the application.

This section contains the following topics:

- Description

- Process Map
- Activity Map
- Pseudocode

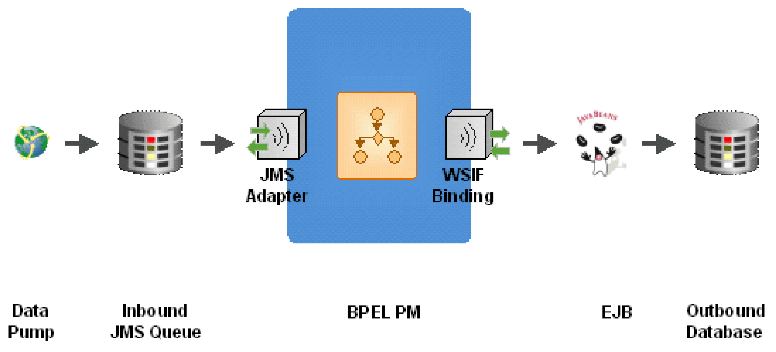
Description

Oracle BPEL Process Manager is the center piece to orchestrate data between the inbound message queue and outbound database. The design details are as follows:

- A Java program (called data pumps) emulates the behavior of the application that sends messages to the inbound queue.
- Oracle AQ provides persistent storage for the inbound queue.
- A resource provider is configured at the same OC4J server as Oracle BPEL Server to be the JMS interface to the AQ.
- An Oracle BPEL Process Manager JMS adapter receives messages from the inbound JMS queue and instantiates BPEL processes. On the outbound side, WSIF binding calls the entity EJBs.
- The BPEL process that orchestrates data between the inbound queue and outbound database works as follows: after receiving an incoming message through the JMS adapter from the inbound queue, the BPEL process calls ParseMsgEJB to parse the message, then calls the session bean StoreOrderEJB to update the outbound database through the Order entity bean.

The logical design is shown in Figure 9–2.

Figure 9–2 Logical Design



Process Map

The BPEL process map is shown in Figure 9–3 (upper part) and Figure 9–4 (lower part).

Figure 9–3 BPEL Process Map (Upper Part)

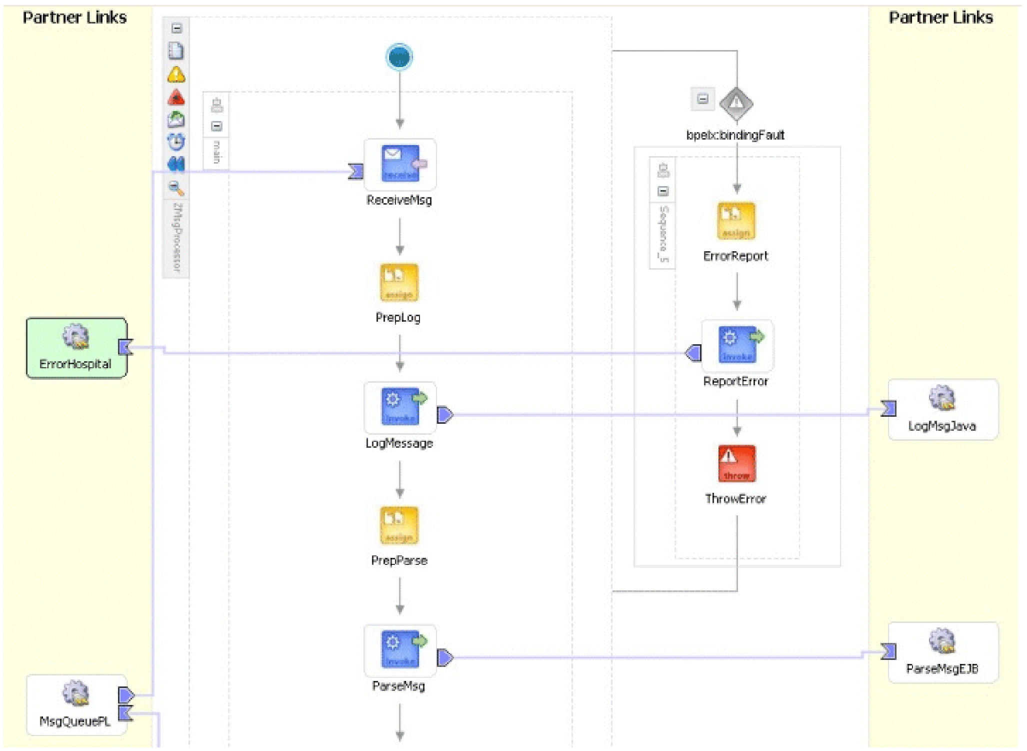
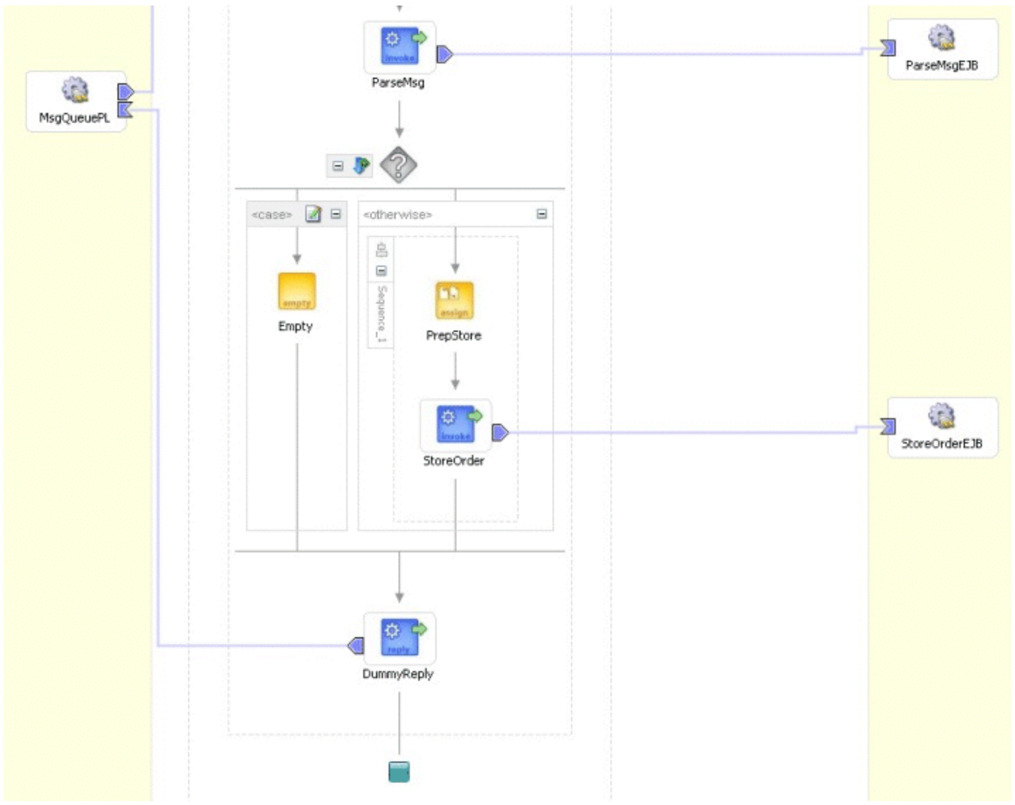


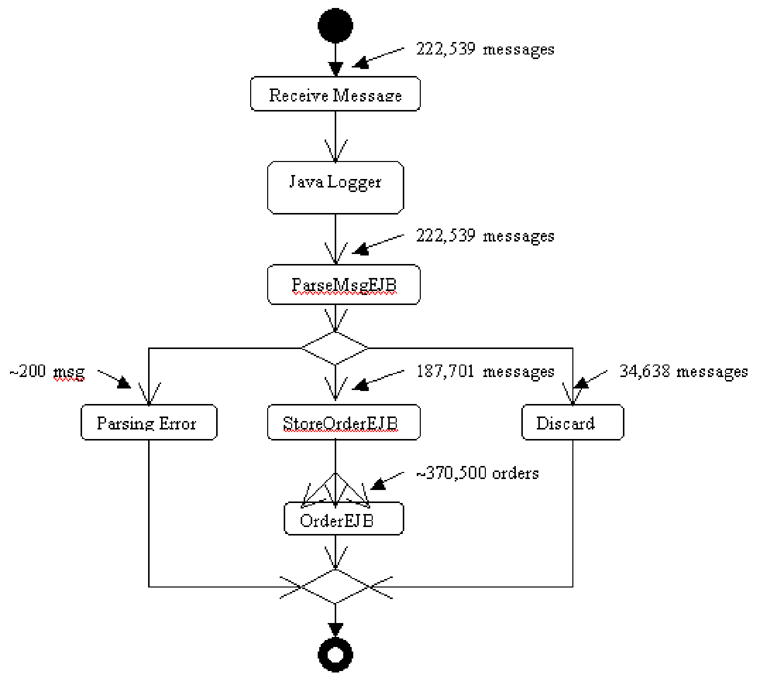
Figure 9–4 BPEL Process Map (Lower Part)



Activity Map

From the perspective of the state of the messages, the BPEL process is illustrated in Figure 9–5. You can see the relationship between incoming messages and business transactions.

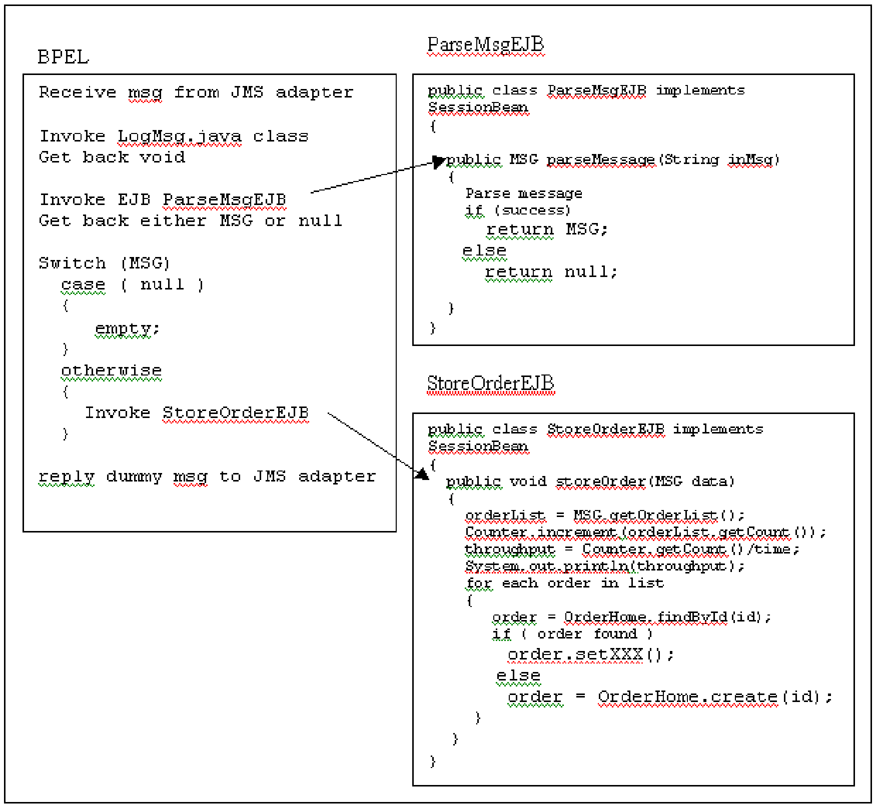
Figure 9–5 Activity Diagram



Pseudocode

The pseudocode shown in Figure 9–6 further explains the implementations of EJBs and the BPEL process:

Figure 9–6 Pseudocode



```
Receive msg from JMS adapter

Invoke LogMsg.java class
Get back void

Invoke EJB ParseMsgEJB
Get back either MSG or null

Switch (MSG)
  case ( null )
  {
     empty;
  }
  otherwise
  {
    Invoke StoreOrderEJB
  }

reply dummy msg to JMS adapter
```

```
public class ParseMsgEJB implements
SessionBean
{

  public MSG parseMessage(String inMsg)
  {
    Parse message
    if (success)
      return MSG;
    else
      return null;

  }
}
```

```
public class StoreOrderEJB implements
SessionBean
{
  public void storeOrder(MSG data)
  {
    orderList = MSG.getOrderList();
    Counter.increment(orderList.getCount());
    throughput = Counter.getCount()/time;
    System.out.println(throughput);
    for each order in list
    {
       order = OrderHome.findById(id);
       if ( order found )
         order.setXXX();
       else
         order = OrderHome.create(id);
    }
  }
}
```

Performance Configurations

This section describes how to configure the various components of this scenario to enhance performance.

This section contains the following topics:

- Architecture
- Optimize the Enqueue Application (Data Pump)
- Optimize the Inbound Queue
- Optimize the JMS Providers
- Optimize the Inbound JMS Adapter
- Optimize the BPEL Process
- Optimize Oracle BPEL Server and OC4J
- Optimize the Outbound Layer
- Create the BPEL Cluster

Architecture

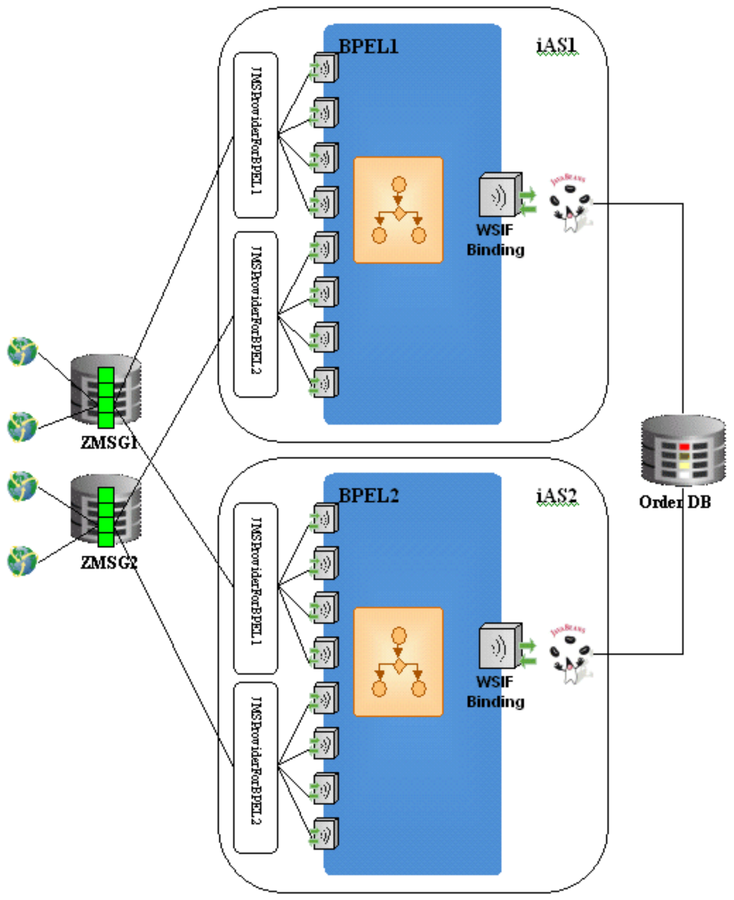
The following system configurations are implemented in order to achieve optimal performance:

- Four threads of data pumps are run from a two-CPU Linux host.

- The inbound AQ queue is installed on a two-node Oracle Real Application Cluster (RAC) database with fast disks.
- A two-node Oracle BPEL Process Manager cluster is run. As a recommended best practice, you do *not* cluster these two Oracle BPEL Process Manager nodes in an Oracle Application Server cluster.
- The JMS providers are configured at the same OC4J container as Oracle BPEL Process Manager.
- Eight instances of the JMS adapter are configured on each Oracle BPEL Process Manager node to receive messages from the two JMS queues. This is to achieve the multithreading effect for both performance and high availability.
- The dehydration database is installed on the same host as the data pumps. This is sufficient to handle the load to the dehydration database, which has been minimized as shown in "Performance Configurations" on page 9-6.
- The EJBs are deployed to the same OC4Js as Oracle BPEL Servers.
- Through WSIF EJB bindings, the BPEL process is configured to talk to the EJBs at the `localhost`.
- The destination database is installed on its own Linux host. This database is not a RAC database.

Figure 9–7 shows this architecture.

Figure 9–7 Architecture



The hardware configurations that implement the architecture are shown in Figure 9–8 and described in Table 9–1.

Figure 9–8 Physical Topology

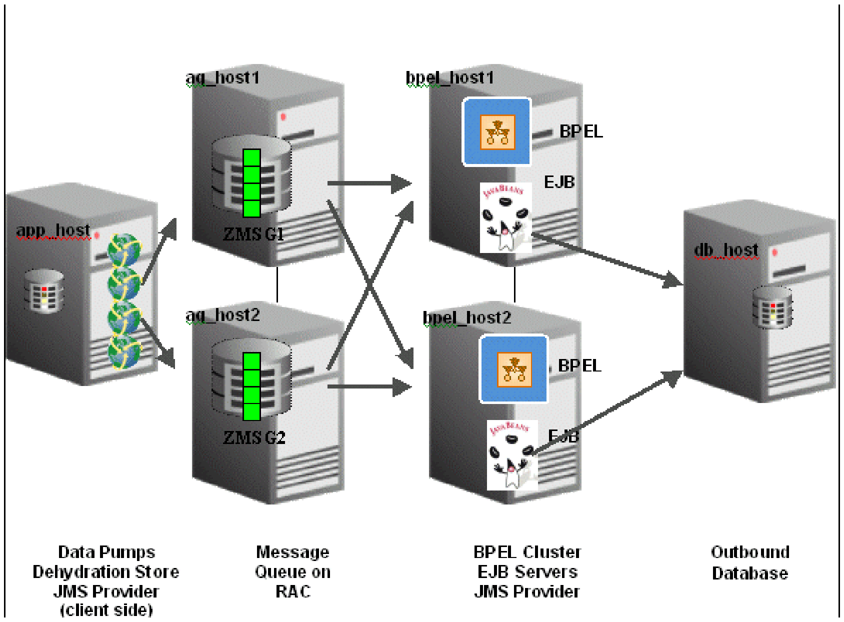


Table 9–1 Hardware Specification

Purpose	Host Name	Specification
▪ Data pump ▪ Dehydration store ▪ JMS providers	App_host	▪ 2 CPU Opteron 283, 2.1 GHz ▪ 4G RAM ▪ Linux 2.4.21-32.ELsmp athlon i386
▪ Oracle Application Server middle tiers 10.1.2.0.0 ▪ Oracle BPEL Servers 10.1.2.0.0 patch 2 (clustered) ▪ EJB Servers	Bpel_host1 Bpel_host2	▪ 2 CPU Opteron 283, 2.1 GHz ▪ 4G RAM ▪ Linux 2.4.21-32.ELsmp athlon i386
Inbound database (Oracle 10.1.0.4) RAC	Aq_host1 Aq_host2	▪ 2-CPU 2.8 GHz Xeons ▪ 1G RAM ▪ Fast Disk ▪ Linux 2.4.21-27.ELsmp i686 i386
Outbound database (Oracle 10.1.0.4)	Db_host	▪ 2 CPU Opteron 283, 2.1 GHz ▪ 4G RAM ▪ Linux 2.4.21-32.ELsmp i386

Optimize the Enqueue Application (Data Pump)

This section describes the objectives and best practices for optimizing the enqueue application.

Objectives

At this step, you try to enqueue messages as fast as possible into the JMS queue. This enables Oracle BPEL Server to always have work to perform.

Best Practices

- Create multiple threads to enqueue.

  In the performance test, four threads of data pumps are run.
- Use thread pooling and connection and session pooling to control the number of concurrent enqueue sessions. This prevents the inbound queue from becoming overloaded.
- Cache the JMS connections and sessions.

  Doing this prevents the overhead of recreating JMS connections and sessions when the application enqueues.
- Batch the commits of enqueue.

  Batching commits improves performance by reducing transaction commit overhead. However, the gains diminish at a point. Also, when setting the enqueue commit size, you must also consider the dequeue commit size. In Oracle BPEL Process Manager 10.1.2, the JMS adapter’s commit size is fixed at 1. In our testing environment, an enqueue commit size of 10 yields the best performance. The optimal batch size for best throughput is dependent on the Java JVM heap size on the client and the system global area (SGA)/log buffers at the database.
- Use JDBC thick driver to optimize the performance for both the application side and Oracle BPEL Process Manager side.

  Note: This scenario was performed on Oracle BPEL Process Manager 10.1.2.0.2. Performance of the JDBC thin driver included with Oracle Application Server has been improved tremendously since version 10.1.3.*x*. Therefore, the performance gain may not be as significant, or even negative, if you are running 10.1.3.*x* Oracle Application Server.

  In performance testing, use of JDBC thick driver did not succeed at the data pump because of configuration issues. However, using thick driver at the dequeue side (that is, JMS adapters) boosted the throughput from 11,000 transaction/minute to 15,000 transactions/minute, or 35%.

Optimize the Inbound Queue

This section describes the objectives and best practices for optimizing the inbound queue.

Objectives

Oracle AQ at the database provides the actual persistent layer for Oracle JMS (OJMS). In performance testing, this layer significantly impacts performance.

A given OJMS AQ destination in a given Oracle database instance can handle a certain level of concurrent access (enqueue and dequeue) to a particular destination before increasing concurrency degrades performance. Therefore, the objective of the AQ optimization is to configure the database and AQ to handle more concurrent threads of mix operations (dequeue and enqueue).

Best Practices

- Install the Oracle AQ queues on a two-node RAC database for higher throughput and availability.
- Stripe the inbound message queue into two queues, and create the two queues on the two-node RAC.

  Alternate the primary and secondary instances for these two queues in the PL/SQL script that creates the queues. For example, use the first node as the primary instance for the first queue and the second node as the primary instance for the second queue. This helps the BPEL process to take advantage of the AQ affinity in a RAC environment for better performance.

  The sample PL/SQL script below creates two queues on the Oracle RAC database, as shown in Table 9–2:

Table 9–2 Relationships among Queues, Queue Tables, and RAC Nodes

Queue Name	Queue Table Name	Primary RAC Node	Secondary RAC Node
ZMSG1_QUEUE	ZMSG1	Aq_host1	Aq_host2
ZMSG2_QUEUE	ZMSG2	Aq_host2	Aq_host1

```
BEGIN
   dbms_aqadm.create_queue_table( queue_table         => 'ZMSG1'
   ,                  queue_payload_type  => 'SYS.AQ$_JMS_TEXT_MESSAGE'
   ,                  sort_list           => 'PRIORITY,ENQ_TIME'
   ,                  comment             => 'JMS ZMSG1 Queue Table'
   ,                  multiple_consumers  => FALSE
   ,                  message_grouping    => DBMS_AQADM.NONE
   ,                  non_repudiation     => DBMS_AQADM.NONE
   ,                  storage_clause      => ''
   ,                  compatible          => '10.0'
   ,                  primary_instance    => '1'
   ,                  secondary_instance  => '2');
   COMMIT;
END;
/
BEGIN
   dbms_aqadm.create_queue_table( queue_table         => 'ZMSG2'
   ,                  queue_payload_type  => 'SYS.AQ$_JMS_TEXT_MESSAGE'
   ,                  sort_list           => 'PRIORITY,ENQ_TIME'
   ,                  comment             => 'JMS ZMSG2 Queue Table'
   ,                  multiple_consumers  => FALSE
   ,                  message_grouping    => DBMS_AQADM.NONE
   ,                  non_repudiation     => DBMS_AQADM.NONE
   ,                  storage_clause      => ''
   ,                  compatible          => '10.0'
   ,                  primary_instance    => '2'
   ,                  secondary_instance  => '1');
   COMMIT;
END;
/

BEGIN
   dbms_aqadm.create_queue( queue_name      => 'ZMSG1_QUEUE'
   ,                       queue_table     => 'ZMSG1'
   ,                       queue_type      => DBMS_AQADM.NORMAL_QUEUE
   ,                       max_retries     => '5'
   ,                       retry_delay     => '0'
   ,                       retention_time  => '0'
   ,                       comment         => 'JMS ZMSG1_QUEUE Queue');
   COMMIT;
END;
/
SHOW ERRORS

BEGIN
   dbms_aqadm.create_queue( queue_name      => 'ZMSG2_QUEUE'
   ,                       queue_table     => 'ZMSG2'
   ,                       queue_type      => DBMS_AQADM.NORMAL_QUEUE
   ,                       max_retries     => '5'
   ,                       retry_delay     => '0'
   ,                       retention_time  => '0'

   ,                       comment         => 'JMS ZMSG2_QUEUE Queue');
   COMMIT;
END;
/
```

- Tune the database parameter `aq_qm_processes`.

  This parameter specifies the number of background processes that remove the dequeued messages from the physical storage. The queue table specified in the queue creation script is merely a view with real tables behind it. Once a message is dequeued, it is not physically removed from those tables immediately. Therefore, the size of the table grows and degrades performance in the case of high volume. Setting this parameter to a value greater than zero helps to reduce the size of the physical table and improve the performance of both enqueue and dequeue operations.

- The following parameters are used for this tuning exercise.

```
open_cursors=1000
processes=1000
db_block_size=8192
db_cache_size=1912602624)
db_file_multi_block_read_count=16
hash_area_size=1024000
job_queue_processes=1
shared_pool_size=402653184
compatible=10.1.0.3.0 (or higher)
log_buffer=6144000 (if feasible, turn logging off altogether)

aq_tm_processes=2 (or more)

gc_defer_time=0
undo_management=AUTO
sort_area_size=1024000
```

- Set the `compatibility` property to `10.0` in the queue creation script.

Optimize the JMS Providers

This section describes the objectives and best practices for optimizing the JMS providers.

Objectives

The JMS provider provides the JMS interface to the physical AQ queue. In OC4J, you configure the JMS provider in the `application.xml` file of the `OC4J_SOA/config` directory.

In the JMS provider configuration, you specify the connection information to the AQ queue, such as the JDBC URL, type of JDBC drivers to use, and so on. The JMS provider configuration also dictates the JNDI names that the JMS adapter refers to later.

The following section describes simple steps to take to improve performance.

Best Practices

- Configure four JMS providers in `application.xml`.

  These four JMS providers differ by two factors:

  – The physical RAC node (node 1 versus node 2)

  – The JDBC driver (OCI versus thin)

Table 9–3 Associations between JMS Providers and RAC Nodes

Provider Name	Used By	Physical Destination
`OracleJMSProviderForDataPump1`	Enqueue application	RAC Node 1
`OracleJMSProviderForDataPump2`	Enqueue application	RAC Node 2
`OracleJMSProviderForBPEL1`	BPEL JMS Adapter	RAC Node 1
`OracleJMSProviderForBPEL2`	BPEL JMS Adapter	RAC Node 2

- Use the RAC node, instead of the RAC load balancing URL, as the JDBC URL. This is to take advantage of AQ affinity.
- Use the thick (that is, OCI) driver for the JMS provider for Oracle BPEL Process Manager.

  You must specify the OCI driver in the JDBC URL in the JMS provider configuration. In performance testing, switching from thin driver to thick driver for the JMS provider for Oracle BPEL Process Manager improved performance 35%.
- Use the thin driver for the JMS provider for the enqueue application (data pump).

  This is done for two reasons:

  – Thick drivers require Oracle SQL client to be installed in the same environment; this may not be convenient for the enqueue application.

  – The enqueue speed is not the bottleneck in this case.
- Sample JMS provider configuration in `application.xml`:

```
<resource-provider name="OracleJmsProviderForDataPump1"
 class="oracle.jms.OjmsContext">
    <property name="url" value="jdbc:oracle:thin:jmsserv/jmsserv@aq_
host1:1522:bpel1"/>
</resource-provider>

<resource-provider name="OracleJmsProviderForDataPump2"
 class="oracle.jms.OjmsContext">
    <property name="url" value="jdbc:oracle:thin:jmsserv/jmsserv@aq_
```

```
host2:1522:bpel2"/>
</resource-provider>

<resource-provider name="OracleJmsProviderForBPEL1"
 class="oracle.jms.OjmsContext">
    <property name="url" value="jdbc:oracle:oci:jmsserv/jmsserv@aq_
host1:1522:bpel1"/>
    <property name="username" value="jmsserv"/>
    <property name="password" value="jmsserv"/>
</resource-provider>

<resource-provider name="OracleJmsProviderForBPEL2"
 class="oracle.jms.OjmsContext">
    <property name="url" value="jdbc:oracle:oci:jmsserv/jmsserv@aq_
host2:1522:bpel2"/>
    <property name="username" value="jmsserv"/>
```

Optimize the Inbound JMS Adapter

This section describes the objectives and best practices for optimizing the inbound JMS adapter.

Objectives

At this step, you try to perform two steps:

- Maximize the speed at which the JMS adapters move incoming messages from the JMS queue to Oracle BPEL Server.
- Remedy the high availability of the inbound queue layer. This is necessary because at the "Optimize the JMS Providers"stage, when you configure the JMS providers to link to individual RAC nodes, you are no longer using the high availability provided by RAC. With proper configuration at the JMS adapter, you can ensure that the BPEL layer continues to function in case one of the RAC nodes goes down at the AQ layer.

Best Practices

- Define the endpoints for the JMS connection factories in the `oc4j-ra.xml` file.

  The `j2ee/OC4J_SOA/application-deployments/default/JmsAdapter/oc4j-ra.xml` file defines the JNDI name for the JMS connection factory and links it to the JMS provider name previously defined in `application.xml`. This JNDI name is the named referenced by the JMS adapter configuration within Oracle BPEL Process Manager.

```
<connector-factory location="eis/Jms/ZMSG1" connector-name="Jms Adapter">
     <config-property name="connectionFactoryLocation"
       value="java:comp/resource/OracleJmsProviderForBPEL1/
       QueueConnectionFactories/myQCF"/>
     <config-property name="factoryProperties"
       value="java.naming.factory.initial=com.evermind.server.rmi.
       RMIInitialContextFactory;java.naming.provider.url=
       ormi://localhost:3202;java.naming.security.principal=admin;
       java.naming.security.credentials=welcome"/>
     <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE"/>
     <config-property name="isTopic" value="false"/>
     <config-property name="isTransacted" value="true"/>
     <config-property name="username" value="jmsserv"/>
     <config-property name="password" value="jmsserv"/>
```

```
</connector-factory>
<connector-factory location="eis/Jms/ZMSG2" connector-name="Jms Adapter">
     <config-property name="connectionFactoryLocation"
       value="java:comp/resource/OracleJmsProviderForBPEL2/
       QueueConnectionFactories/myQCF"/>
     <config-property name="factoryProperties"
        value="java.naming.factory.initial=com.evermind.server.rmi.
        RMIInitialContextFactory;java.naming.provider.url=ormi:
       //localhost:3202;java.naming.security.principal=admin;java.naming.
       security.credentials=welcome"/>
     <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE"/>
     <config-property name="isTopic" value="false"/>
     <config-property name="isTransacted" value="true"/>
     <config-property name="username" value="jmsserv"/>
     <config-property name="password" value="jmsserv"/>
</connector-factory>
```

The relationship between the JMS connection factories, the JMS providers, and their physical destinations is described in Table 9–4:

Table 9–4 Relationship between JMS Connection Factories, Providers, and Physical Destinations

JMS Connection Factory	JMS Provider	Physical Destination
`eis/Jms/ZMSG1`	`OracleJmsProviderForBPEL1`	RAC Node 1
`eis/Jms/ZMSG2`	`OracleJmsProviderForBPEL2`	RAC Node 2

- Modify the WSDL file of the JMS adapter.

  You must perform the following step in the WSDL file.

  – Parameterize the connection location and queue name in the adapter's WSDL file. Their actual values are later defined in the `bpel.xml` file.

```
<binding name="Consume_Message_binding" type="tns:Consume_Message_ptt">
  <pc:inbound_binding />
  <operation name="Consume_Message">
  <jca:operation
     ActivationSpec="oracle.tip.adapter.jms.inbound.JmsConsumeActivationSpec"
     DestinationName="$JmsQueue"
     UseMessageListener="false"
     PayloadType="TextMessage"
     OpaqueSchema="false" >
  </jca:operation>
  . . .
</binding>
  . . .
<service name="ZMsgQueue">
  <port name="Consume_Message_pt" binding="tns:Consume_Message_binding">
    <jca:address location="$JndiLocation" />
  </port>
</service>
```

- Configure JMS adapter activation agents in the `bpel.xml` file.

  A JMS adapter consists of one or more activation agents. Activation agents are located between the BPEL process and JMS queues. They listen to JMS queues and forward messages to BPEL processes.

Multiple activation agents can be configured for concurrent processing of JMS messages. Moreover, the queue name and destination of each activation agent can be individually configured. This feature enables you to achieve both high throughput and high availability.

Once you define a partner link with the JMS adapter, Oracle JDeveloper creates one `<activationAgent/>` for the adapter in the `bpel.xml` file. You must now manually modify the `bpel.xml` file to create multiple activation agents:

– Define the values for the queue name and destination using the `<property/>` tag. The property names must match the parameter names that are previously declared in the adapter WSDL file (see the last bullet). In this use case, the property name for the queue is `JmsQueue` and the property name for the destination is `JndiLocation`.

– Duplicate the same block of XML code (that is, the `<activationAgent/>` tag) multiple times. In performance testing, eight agents per node on a two-node BPEL cluster yield the most optimal performance (for details, see "Final Numbers" on page 9-22).

– Vary the values of queue name and destination in each of the agent instances (that is, the `<activationAgent/>` tag). To an extreme, queue name and destination can be unique per agent.

Oracle BPEL Process Manager 10.1.3 makes it easier to configure multiple, identical activation agents. This means you do not need to create duplicate entries of `<activationAgent/>` in `bpel.xml`.

You create two tags, one for each set of activation agents, and add the `activationInstances` property to configure four instances of agents in each set.

Table 9–5 Configuration of Multiple Activation Agents

Agent Set	Agent Count	Destination	Queue Name (Truncate Path to Fit Document Space)
1	4	`eis/Jms/ZMSG1`	`java:comp/…/Queues/ZMSG1_QUEUE`
2	4	`eis/Jms/ZMSG2`	`java:comp/…/Queues/ZMSG2_QUEUE`

```
<activationAgents>

  <activationAgent
 className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
 partnerLink="MsgQueuePL">

     <property name="JndiLocation">eis/Jms/ZMSG1</property>
     <property
 name="JmsQueue">java:comp/resource/OracleJmsProviderForBPEL1/Queues/ZMSG1_
QUEUE</property>
        <property name="activationInstances">4</property>
     <property name="portType">Consume_Message_ptt</property>

  </activationAgent>

  <activationAgent
 className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
 partnerLink="MsgQueuePL">
```

```
     <property name="JndiLocation">eis/Jms/ZMSG2</property>
     <property
 name="JmsQueue">java:comp/resource/OracleJmsProviderForBPEL2/Queues/ZMSG2_
QUEUE</property>
        <property name="activationInstances">4</property>
     <property name="portType">Consume_Message_ptt</property>
  </activationAgent>

</activationAgents>
```

Optimize the BPEL Process

This section describes the objectives and best practices for optimizing the BPEL process.

Objectives

The objectives are to optimize the BPEL process for both performance and reliability.

Saving the process state to the dehydration store achieves reliability, specifically the ability to identify and recover from failures. However, doing so has an impact on performance. In contrast, turning off the dehydration boosts performance, but you jeopardize reliability. The following best practices help you satisfy the requirements for both performance and reliability.

Best Practices

- Make the BPEL process *synchronous*.

  If a BPEL process is asynchronous, BPEL messages are persisted to the dehydration store unless the `deliveryPersistPolicy` at `domain.xml` is set to `off`. This is necessary for recovery in the case of errors.

  For a synchronous process, Oracle BPEL Server does not persist the incoming messages to the dehydration store. This is because the caller receives a fault synchronously and therefore can handle the error accordingly.

  If the BPEL process is initiated by a JMS adapter (as in this case), the JMS adapter receives an exception from Oracle BPEL Server in case of a fault. The JMS adapter in turn processes the rejected messages with a configurable rejection handler. The default rejection handler stores the rejected message to a configurable location for later recovery. By default, the location is this directory:

  ```
  IAS_ORACLE_HOME/bpel/domains/domain_name/archive/jca/process_
  name/rejectedMessage
  ```

  Where *`domain_name`* and *`process_name`* depend upon your actual settings. For example, *`domain_name`* is `default` and *`process_name`* is `HelloWorld`.

  You can configure your own rejection handler, storing the messages to an AQ queue or to another BPEL process. For details of how to configure a rejection handler for the JMS adapter, see "Rejection Handlers" on page 3-32 and *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide*.

  If Oracle BPEL Server and the JMS adapters both go down, JMS messages are rolled back to the inbound JMS queue.

  To conclude, making the BPEL process synchronous prevents Oracle BPEL Process Manager from storing all incoming messages to the dehydration store. At the same time, this also provides the ability to store and handle the rejected incoming messages in case of faults. This optimizes both performance and reliability.

To implement this strategy, perform the following steps.

– Modify the WSDL of the JMS adapter (the partner link for initializing the process) so that `portType` performs a synchronous operation. You must first define an empty message type, then modify the `portType` operation to accept the empty message as an output:

```
<message name="EMPTY_msg">
  <part name="empty" element="imp1:empty"/>
</message>
<portType name="Consume_Message_ptt">
    <operation name="Consume_Message">
        <input message="tns:TFMM_MSG_msg"/>
        <output message="tns:EMPTY_msg"/>
    </operation>
</portType>
```

– Modify the BPEL process so that it replies with a dummy message to the JMS adapter.

```
<variable name="dummyReply" messageType="ns1:EMPTY_msg"/>
<reply name="DummyReply" partnerLink="ZMsgQueuePL"
   portType="ns1:Consume_Message_ptt"
   operation="Consume_Message" variable="dummyReply"/>
```

- Set the following performance persistence parameters in `bpel.xml`:

```
<BPELSuitcase>
   . . .
  <BPELProcess>
   . . .
    <configurations>
      <property name="inMemoryOptimization">true</property>
      <property name="completionPersistPolicy">faulted</property>
    </configurations>
    . . .
  </BPELProcess>
</BPELSuitcase>
```

Setting `completionPersistPolicy` to `faulted` means that only the faulted BPEL process instances get their state persisted to the dehydration store and are therefore visible in Oracle BPEL Control. The successful BPEL instances are not dehydrated and therefore do not (and do not need to) appear in Oracle BPEL Control.

- For EJB binding, use `ormi://localhost:port/` as the EJB provider URL at the WSIF binding WSDL.

  Do not use the actual host name. Use `localhost` instead. This simplifies management and deployment: the BPEL process always communicates with the EJB on the same server, and the same ORMI URL can be applicable across multiple nodes of a BPEL cluster.

  The EJB WSIF binding WSDL file looks as follows:

```
  <service name="StoreOrderService">
    <port name="EJBPort" binding="tns:EJBBinding">
      <ejb:address className="gov.ejb.StoreOrderHome" jndiName="StoreOrder"
      initialContextFactory="com.evermind.server.rmi.RMIInitialContextFactory"
```

```
            jndiProviderURL="ormi://localhost:12402/StoreOrderEJB"/>
      </port>
    </service>
```

Optimize Oracle BPEL Server and OC4J

This section describes the objectives and best practices for optimizing Oracle BPEL Server and OC4J.

Objective

Performance testing does not indicate that Oracle BPEL Process Manager requires much tuning because most bottlenecks are at the layers of AQ, JMS providers, and JMS adapters. However, there are still several Oracle BPEL Server settings worth modifying to receive optimal performance.

Best Practices

- Change the logging setting to `fatal` only.

  This minimizes the load on Oracle BPEL Server and the waits on I/O.

- Set `instanceKeyBlockSize` to `100,000`.

  Set this property through **Oracle BPEL Control** > **Manage BPEL Domain** > **Configuration** or manually edit the `domain.xml` file. This property specifies the batch of unique BPEL instance IDs that Oracle BPEL Server retrieves from the dehydration database. A value of `100,000` means that in the lifetime of the OC4J JVM, it makes a trip to the dehydration store for every `100,000` BPEL instances. Increasing this number decreases the frequency at which Oracle BPEL Server visits the dehydration store.

- Optimize the JVM heap.

  The heap size controls the amount of memory the JVM can use. Tuning the JVM heap can ensure that there is enough memory for Oracle BPEL Server to consume. Tune this property in the *SOA_ORACLE_HOME*`/opmn/conf/opmn.xml` file:

  ```
  <process-type id="OC4J_SOA" module-id="OC4J">
       <module-data>
          <category id="start-parameters">
             <data id="java-options" value="-server
   -Djava.security.policy=/scratch/fip/iAS10.1.2Midtier/j2ee/OC4J_
    BPEL/config/java2.policy -Djava.awt.headless=true -Doc4j.userThreads=true
    -Xbootclasspath^/p:/scratch/fip/iAS10.1.2Midtier/integration/orabpel/lib/
    orabpel-boot.jar:/scratch/fip/iAS10.1.2Midtier/integration/orabpel/lib/
    connector15.jar -server -Xmn1228m -XX:+AggressiveHeap -Xms2048m -Xmx2048m
   -Doracle.mdb.fastUndeploy=60
   -Dorabpel.home=/scratch/fip/iAS10.1.2Midtier/integration/orabpel
   -Dhttp.proxySet=true -Dhttp.proxyHost=www-proxy.us.oracle.com
   -Dhttp.proxyPort=80
   -Dhttp.nonProxyHosts=localhost|stacx56.us.oracle.com|*.us.oracle.com|stacx56
  "/>
    <data id="oc4j-options" value="-properties"/>
  </category>
  ```

  An explanation of the settings follows:

  – `-Xmx 2048m` — Set the maximum heap size to 2 GB. Because Oracle BPEL Servers run on dedicated hosts in this scenario, you set the JVM heap size as high as possible. Note that this number is constrained by the operating system's addressable memory space. For 32-bit operating systems, the limit is 4 GB. For Windows 32-bit systems, the limitation is 2 GB (due to operating system limitations).

– `-Xms 2048m` — Set the initial heap size to 2 GB. Sun recommends setting the values for maximum and initial heap size to be equal.

– `-Xmn1228m` — Set the Eden space to be 60% of the maximum heap size. Eden space is a parameter in the JVM garbage collector's generational settings. The garbage collector optimizes collection by classifying objects by how long they live. Most of Oracle BPEL Server’s objects are short-lived. Therefore, they live in the Eden space. Oracle recommends sizing the Eden space to be 60 to 70 percent of the total heap size, as shown in the syntax example above.

– `-XX:+AggressiveHeap` — Set the aggressive heap. This setting is recommended for multiple CPU non-Windows hosts. This property inspects the host resources (size of memory and number of processors) and attempts to set various parameters to be optimal for long-running, memory allocation-intensive jobs. This property has no effect on Windows platforms.

Optimize the Outbound Layer

This section describes the objectives and best practices for optimizing the outbound layer.

Objectives

The EJBs, their bindings, and the destination database represent the outbound layer. The time spent at this layer is included in the total response time of the BPEL process. You try to eliminate the possible bottleneck at this layer.

Best Practices

- WSIF binding (`localhost`)

  As mentioned in "Optimize the BPEL Process" on page 9-16, you use `ormi://localhost:`*`port`*`/`*`application_name`* as the JNDI provider URL in the WSDL file for the EJB binding. This ensures that the BPEL process always communicates with the EJB on the same server. In addition, the same ORMI URL can be applicable across multiple nodes of a BPEL cluster for the purpose of simplifying deployment and management.

- Data-source configuration (maximum connections)

  The container-managed persistence (CMP) EJBs use a data source to connect to the outbound database. Performance of the EJB can be tuned here. Specifically, in the performance testing for this scenario:

  – Set `max-connection` to `80`.

  – Use the JDBC OCI driver

  These two parameters are set in the `j2ee/OC4J_SOA/config/data-sources.xml` file:

```
<data-source name="EJBrmp5"
    class="com.evermind.sql.DriverManagerDataSource"
    connection-driver="oracle.jdbc.OracleDriver"
    location="jdbc/rmp5Core"
    xa-location="jdbc/rmp5XA"
    ejb-location="jdbc/rmp5"
    connection-retry-interval="30"
```

```
max-connect-attempts="10"
min-connections="2"
max-connections="80"
url="jdbc:oracle:oci:@db_host:1521:orcl"
username="scott"
password="tiger"/>
```

- Tune the outbound database.

  Similar to the inbound database, you must set proper values for sessions and processes to ensure that the database can handle a large throughput.

Create the BPEL Cluster

This section describes the objectives and best practices for creating the BPEL cluster.

Objectives

When the CPUs on a single BPEL node are saturated, adding more BPEL nodes and forming a BPEL cluster can improve performance. In the JMS-to-database scenario, this is often accompanied by adding a node to the inbound JMS cluster.

In performance testing, putting a BPEL cluster on top of an Oracle Application Server middle tier cluster was found to be complex and error prone. This changed in the 10.1.3.0.1 release of Oracle BPEL Process Manager and Oracle Application Server. If you are using a pre-10.1.3.0.1 release, the recommended best practice is to avoid the mixture of middle-tier cluster and Oracle BPEL Process Manager Cluster.

See Also: The following documentation for instructions on creating clusters:

- *Oracle Application Server Enterprise Deployment Guide*
- *Oracle Application Server High Availability Guide*

Best Practices

- Add nodes to both the inbound JMS queue cluster and BPEL cluster.

  As described in "Optimize the Inbound Queue" on page 9-9, the inbound JMS queue has a limit on throughput per queue per node. If the following two conditions exist, adding nodes to the BPEL cluster may help improve performance:

  – CPU is saturated on BPEL hosts.

  – A backlog appears at the inbound queue.

  In contrast, if the following conditions exist, consider adding nodes to the inbound JMS queue cluster:

  – CPU on BPEL hosts are underutilized (below 50%).

  – No backlog exists at the inbound queue.

  In fact, you may reach the optimal JMS and AQ and Oracle BPEL Process Manager cluster configurations through multiple iterations of trial and error. For example, you start testing with one JMS and AQ node and one queue and soon reach its limitation. You then add a JMS and AQ node to the RAC. The BPEL hosts immediately saturate the CPU and the messages backlog in the inbound queue appears. You then add an Oracle BPEL Process Manager node and the JMS queue backlog is reduced.

Because creating a two-node JMS and AQ queue cluster was already described in the previous section, you are focused here only on the Oracle BPEL Process Manager cluster.

- Avoid application server (middle tier) clustering.
- If application server clustering is inevitable, perform the Oracle BPEL Process Manager cluster *before* performing an Oracle Application Server middle tier cluster. Here are the exact steps:
  - Install all the middle tier nodes independently without creating a cluster. Oracle recommends installing the Oracle Application Server node first using automatic port assignment and then using its port configuration file for manual port assignment for the installation of subsequent Oracle Application Server nodes. This forces the same port assignment across the cluster to ease load balancer setup and eases the BPEL JMS adapter and EJB binding configuration.
  - Install Oracle BPEL Server on each of these nodes. These Oracle BPEL Servers share the same dehydration store (that is, creating a BPEL cluster)
  - Join these middle tier nodes into the same middle tier cluster.

Other Recommendations

This section describes addition performance recommendations.

Test Maximum Inbound Queue Throughput with Java Programs

You use a Java program to test the maximum throughput that the message queue layer can deliver and use it to guide performance tuning. In performance testing, a Java program was run to enqueue messages to the JMS queue. At the same time, another Java program was run to dequeue messages from the same JMS queue. A maximum throughput of 150 messages per second per queue cluster node was achieved when there were two enqueue threads and four dequeue threads.

Test Maximum Oracle BPEL Process Manager and Adapter Throughput with a Preloaded Queue

It is useful to test the maximum throughput of the Oracle BPEL Process Manager and adapter layer with a preloaded inbound queue; that is, to first fill up the queue and then start Oracle BPEL Server.

You use this simplified environment to tune Oracle BPEL Server (such as JVM heap size). You also use this simplified environment to identify any potential concurrency-related issues at the outbound layer (for example, EJB/TopLink/database). This can prevent you from going in the wrong direction during actual performance tuning. In performance testing, it is at this stage that you discover that the EJB layer experiences locking issues under stress. Some simple tuning of the outbound database can also be conducted with this test. For example, you initially do not plan to make any changes to the database. Towards the end, however, you apply some simple and obvious changes to the database that yield better throughput.

Performance Tuning

Performance tuning guidelines were derived from the performance tests, as documented in Table 9–6. You can follow these guidelines to tune your system for optimal performance.

Table 9–6 Performance Tuning

Behavior	Diagnosis	Solution
■ Backlog appears in the inbound queue ■ CPUs and memory on BPEL hosts are underutilized	The JMS adapters are not sending the messages to Oracle BPEL Process Manager fast enough. In other words, the bottleneck is at the Oracle BPEL Process Manager JMS adapter layer.	Increase the number of JMS agents in `bpel.xml`.
■ No backlog in the inbound queue ■ CPUs and memory on BPEL hosts are underutilized	The bottleneck is at the inbound queue layer.	Two possible solutions: ■ First try to adjust the number of enqueue and dequeue threads and their ratio. ■ Then try to add a node to the inbound message queue cluster.
■ Backlog appears in the inbound message queue ■ CPUs and memory on BPEL hosts are saturated	A single node reaches its limit. In other words, the bottleneck is at the Oracle BPEL Server layer.	Add a BPEL node.

Final Numbers

This section describes throughput details and performance configuration values.

Measurement

This section describes throughput details.

Throughput by Orders

The customer measures the throughput by number of orders processed per minute. This number is shown as throughput by orders in Table 9–7 on page 9-23. As indicated in "Pseudocode" on page 9-5, the StoreOrderEJB calls a Java class that counts the total number of orders and calculates the throughput by dividing the order count by elapsed time.

Throughput by BPEL Instances

The throughput by BPEL instances is also calculated based on the throughput by orders. As shown in Figure 9–5 on page 9-5, among the 222,539 incoming messages, about 200 of them are malformed and throw an exception from Oracle BPEL Process Manager and 34, 638 of them are discarded by design. The remaining 187,701 messages result in 370,500 order creations or updates. Therefore, the ratio between business orders and Oracle BPEL Process Manager instances is as follows:

```
370,500/222,539 = 1.66 orders/message
```

The relationship between throughput by Oracle BPEL Process Manager instance and throughput by orders is as follows:

```
Throughput by BPEL Instance = Throughput By Orders / 1.66
```

Performance Values by Configurations

Table 9–7 shows the performance values based on configuration.

Table 9–7 Final Performance Numbers

Test Title	Test Configuration	Throughput by Orders (order/min)	Throughput by BPEL Instances (Instance/Minute)
Java test	1. Four data pump threads 2. Two inbound queues on two AQ nodes 3. Ten threads of Java dequeuer program in place of Oracle BPEL Process Manager	33664 order/min	20280 instance/min (338 instance/sec)
Preloaded queue test	1. Two inbound queues on two AQ nodes, preloaded with messages 2. Two BPEL nodes 3. Eight activation agents per BPEL node	20300 order/min	12228 instance/min (203 instance/sec)
Single AQ node test	1. Four data pump threads 2. One inbound queue on one AQ node 3. Two BPEL nodes 4. Five activation agents per BPEL node	15000 order/min	9036.14 instance/min (150 instance/sec)
Two AQ node test	1. Four data pump threads 2. Two inbound queues on two AQ nodes 3. Two BPEL nodes 4. Ten activation agents per BPEL node	19000 order/min	11445 instance/min (191 instance/sec)

10

Oracle BPEL Process Manager Performance

This chapter describes performance and tuning best practices for Oracle BPEL Process Manager.

This chapter contains the following topics:

- Performance Concepts and Terminology
- Using the File Adapter to Process Large File Sizes
- Configuring Your Environment to Optimally Process Large Payloads
- Benchmark and Architecture
- Clustering, Load Balancing, Failover, and Recovery
- Stability and Memory
- Performance Tuning
- Transactions Related
- Dehydration Store Management
- Load Throttling

Note: The Oracle BPEL Process Manager Performance Tuning chapter of *Oracle Application Server Performance Guide* for details about setting process, domain, OC4J, JVM, and Oracle database properties that impact Oracle BPEL Process Manager performance.

Performance Concepts and Terminology

This section provides an overview of Oracle BPEL Process Manager performance concepts and terminology.

This section contains the following topics:

- Breakpoint Activities
- Idempotent Activities
- Durable Versus Transient Processes
- One-Way Invocation Versus Two-way Invocation (Synchronous And Asynchronous Processes)
- Dehydration
- Delivery Service

- Threads and Transactions

> **See Also:** *Oracle Application Server Performance Guide* for additional details about Oracle BPEL Process Manager performance concepts and terminology

Breakpoint Activities

A breakpoint activity is an activity in which Oracle BPEL Server must wait for the next incoming messages or for a timer expiration. Breakpoint activities are as follows:

- Receive
- Pick (which contains onMessage and onAlarm branch conditions)
- Wait

Idempotent Activities

An idempotent activity is an activity that can be retried (for example, an assign activity). A nonidempotent activity is an activity that can happen only once (for example, an invocation to a hotel booking Web service). Oracle BPEL Server saves the instance after a nonidempotent activity.

An invoke activity is idempotent by default in Oracle BPEL Process Manager, but can be configured. All breakpoint activities are always nonidempotent. All other activities are idempotent.

Durable Versus Transient Processes

There are two types of BPEL processes:

- Transient — a process that does not incur any midprocess breakpoint and idempotent activities. In other words, a transient process never stops and waits for any events in the middle of the process. Transient processes are typically short-lived. The CreditRatingService sample included with Oracle BPEL Process Manager is a typical transient process.
- Durable — a process that has one or more midprocess breakpoint or idempotent activities. In other words, a durable process must stop and wait for some events in the middle of the process. Such events can be the arrival of a message or expiration of a timer. Durable processes are typically long-lived. The LoanFlow sample included with Oracle BPEL Process Manager is a typical durable process.

To determine the process type, examine your process flow. If there are any breakpoint activities, such as receive, wait, or pick, then the process is durable. Otherwise, the process is transient.

One-Way Invocation Versus Two-way Invocation (Synchronous And Asynchronous Processes)

A BPEL process's receive activity can receive either of the following invocation types from partners:

- One-way invocation — this type is a request-only operation. Oracle BPEL Server immediately returns to the calling partner after receiving the message. The BPEL process may or may not use an invoke activity to send a callback message to the same partner sometime later in the process. This action depends on the functional requirements. A one-way invocation is also called an asynchronous invocation.

The message associated with the one-way invocation is called an asynchronous message. If the initial receive activity of a BPEL process is receiving a one-way invocation from its partner, this process is usually called an asynchronous process. The HelloWorld sample included with Oracle BPEL Process Manager is an example of an asynchronous process.

- Two-way invocation — this type is a request and response operation and must have both inbound and outbound messages. After receiving the message, the BPEL process must use the reply activity to send a reply to the caller. The partner blocks until it gets a response from the BPEL process. A two-way invocation is also called a synchronous invocation. The message associated with the two-way invocation is called a synchronous message. If the initial receive activity of a BPEL process is receiving a two-way invocation from its partner, this process is usually called a synchronous process. The CreditRatingService sample included with Oracle BPEL Process Manager is an example of a synchronous process.

One-way or two-way invocations are the interface to the BPEL process. The operations are defined in the WSDL file. The receive activity of a BPEL process for these two operations also differs in the `.bpel` file. Table 10–1 describes these invocation types.

Table 10–1 One-Way and Two-Way Invocations

Use	One-Way Invocation	Two-Way Invocation

WSDL file definition	`<operation name="oneway">`
	` <input message="in"/>`
	`</operation>`
		`<operation name="twoway">`
		` <input message="in"/>`
		` <output message="out"/>`
		`</operation>`

Variable declarations in BPEL activities	`<receive operation="oneway"`
	` variable="in"/>`
		`<receive operation="twoway"`
		` variable="in"/>`
		`...`
		`<reply operation="twoway"`
		` variable="out"/>`

Through-delivery service	The request is saved in the delivery service. The caller thread does not block until the message is delivered to the targeted instance.	The request is delivered into Oracle BPEL Server and the targeted BPEL instance. The caller thread is blocked until the response is ready.

The terms asynchronous process and synchronous process only refer to the operation types of the initial receive activity of a BPEL process. Because a BPEL process can have multiple receive activities, it can have both one-way and two-way operations in one process. Therefore, it is legitimate for an asynchronous process to have receive and reply activities in the middle of the process to handle a two-way invocation. It is also legitimate for a synchronous process to continue to receive a one-way invocation from another partner after it provides the first partner with a two-way reply. In other words, an asynchronous process can be transient and a synchronous process can be durable.

Dehydration

Over the life cycle of a BPEL instance, the instance with its current state of execution may be saved in a database. When a BPEL instance is saved to a database, the instance is also known as being dehydrated. The database where the BPEL instance is saved is called a dehydration store.

Once a BPEL instance is dehydrated, Oracle BPEL Server can off load it from the memory of Oracle BPEL Server. When a certain event occurs, such as the arrival of a message or the expiration of a timer, Oracle BPEL Server locates and loads the pertinent BPEL instance from the dehydration store back into the memory of Oracle BPEL Server and resumes the execution of the process instance.

Dehydrating BPEL instances offers reliability. If Oracle BPEL Server crashes in the middle of executing a process, the instance can be recovered automatically, programmatically, or manually from the dehydrated states. When Oracle BPEL Server resumes the execution of the process instance, it resumes from the last dehydration point, which is the last state of the instance that Oracle BPEL Server saves to the dehydration store.

There are three cases in which dehydration occurs:

- When the BPEL instance encounters a midprocess breakpoint activity (not including the initial receive)

  That is where an existing BPEL instance must wait for an event, which can be either a timer expiration or message arrival. When the event occurs (the alarm expires or the message arrives), the instance is loaded from the dehydration store and execution is resumed. This type of dehydration occurs only in durable processes, which have midprocess breakpoint activities. A transient process does not have any midprocess breakpoint activities.

- When the BPEL instance encounters a nonidempotent activity

  When Oracle BPEL Server recovers after a crash, it retries the activities in the process instance. However, it should only retry the idempotent activities. Therefore, when Oracle BPEL Server encounters a nonidempotent activity, it dehydrates it. This enables Oracle BPEL Server to memorize that this activity was performed once and is not performed again when Oracle BPEL Server recovers from a crash.

- When the BPEL instance finishes

  At the end of the BPEL process, Oracle BPEL Server saves the process instance to the dehydration store, unless you explicitly configure it not to do so. This happens to both durable and transient processes. For transient processes, the end of the process is the only point where the process instance is saved. This is because a transient process does not have any midprocess breakpoint activities and nonidempotent activities where the inflight dehydration can occur.

When and how the dehydration occurs differs based on the process types:

- Transient process — Oracle BPEL Server dehydrates the process instance only once at the end of the process. When a host crashes in the middle of running the process instance, the instances are not visible from Oracle BPEL Control.
- Durable process — Oracle BPEL Server dehydrates the process instance inflight at all midprocess breakpoint and idempotent activities, plus the end of the process. When the server crashes, this process instance appears in Oracle BPEL Control up to the last dehydration point (breakpoint activity) once the server restarts. If the server crashes before the process instance reaches the first midprocess breakpoint activity, the instance is not visible in Oracle BPEL Control after the server restarts.

Delivery Service

Oracle BPEL Process Manager uses the delivery service to handle incoming messages and correlate them to, or instantiate, the appropriate process instances.

For asynchronous messages, Oracle BPEL Server maintains a delivery queue at the dehydration store database to persist incoming asynchronous messages. Once the asynchronous messages arrive at Oracle BPEL Server, the delivery service immediately saves them to the delivery queue in the dehydration store. Therefore, once a client sends an asynchronous BPEL message and does not receive an exception, the client is guaranteed that Oracle BPEL Server has reliability received the message. If Oracle BPEL Server crashes before the delivery service saves the incoming messages to the delivery queue, the partner gets an exception. If Oracle BPEL Server crashes after the delivery service saves the incoming message, but before reaching a dehydration point (for example, a midprocess breakpoint activity as with a durable process or the end of the process as with a transient process), the message is rolled back to the delivery queue. Automatic, manual, or programmatic recovery can be performed from the saved messages. Sometimes saving the messages to the delivery queue is also known as dehydration.

For synchronous incoming messages, Oracle BPEL Server never saves them to the dehydration store. The partner blocks until getting a reply from Oracle BPEL Process Manager. If Oracle BPEL Server crashes before replying to the partner, the partner who sent this message to Oracle BPEL Process Manager gets an exception or timeout and can handle it accordingly.

Threads and Transactions

Whenever Oracle BPEL Server dehydrates messages or process states, a transaction is committed. In some scenarios, such as a synchronous transient process, one thread is created to dehydrate the instance in one transaction. In other scenarios, such as an asynchronous durable process, multiple threads are created to run the process instance and dehydrate it inflight in multiple transactions. Understanding how the threads and transactions work in Oracle BPEL Server helps you select the proper performance tuning settings.

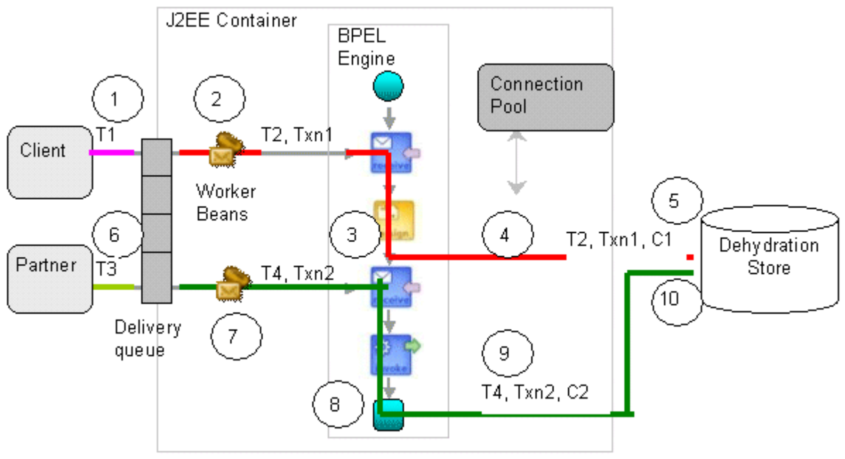
Oracle BPEL Server's threading and transaction actions can be illustrated by the following two scenarios.

- Threading and Transactions for an Asynchronous Durable Process
- Threading and Transactions for a Synchronous Transient Process

Threading and Transactions for an Asynchronous Durable Process

Figure 10–1 shows the threading and transaction actions involved in an asynchronous durable process.

Figure 10–1 Threading and Transactions for an Asynchronous Durable Process

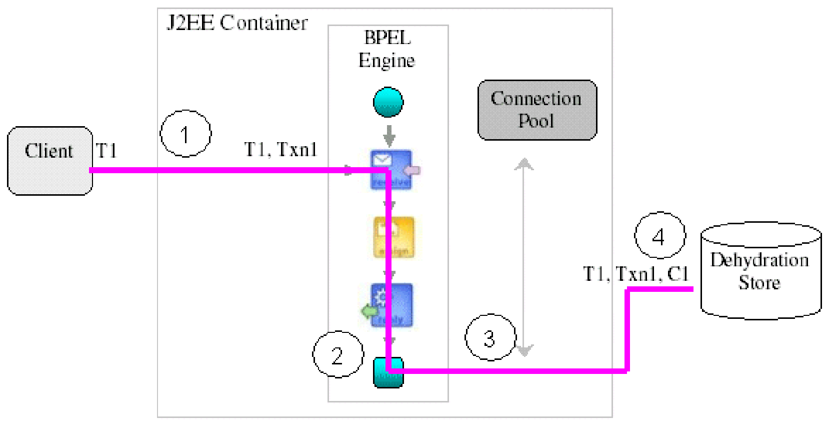


1. Caller thread T1 receives an incoming message and saves it to the internal delivery queue. Thread T1 is then released by Oracle BPEL Server and the caller can continue its own processing.
2. Inside the server, a message driven bean called WorkerBean monitors the queue for invocation requests. The WorkerBean picks up the message from the delivery queue, spawns a new thread (T2), and starts a Java Transaction API (JTA) transaction (Txn1) to create and run the instance.
3. Thread T2 encounters a midprocess breakpoint activity (receive) and decides to dehydrate the process instances.
4. Thread T2 retrieves a database connection (C1) from the data source connection pool.
5. Thread T2 saves the process instance to the dehydration database, which ends the transaction (Txn1).
6. Caller thread T3 intercepts an invocation from the partner and saves the incoming message to the delivery queue (dehydration point).
7. The Oracle BPEL Server WorkerBean picks up the message from the delivery queue, spawns a new thread (T4), and starts a JTA transaction (Txn2) to continue the instance.
8. Thread T4 encounters the end of the process and must dehydrate the process instance.
9. Thread T4 retrieves a database connection (C2) from the data source connection pool.
10. Thread T4 saves the process instance to the dehydration database, which ends the transaction (Txn2).

Threading and Transactions for a Synchronous Transient Process

Figure 10–2 shows the threading and transaction actions involved in a synchronous transient process.

Figure 10–2 Threading and Transactions for a Synchronous Transient Process



1. Caller thread T1 receives a message and starts a JTA transaction (Txn1) to create and run the instance.

2. Thread T1 encounters the end of the process and must dehydrate the process instance.
3. Thread T1 retrieves a database connection (C1) from the data source connection pool.
4. Thread T1 saves the process instance to the dehydration database, which ends the transaction (Txn1).

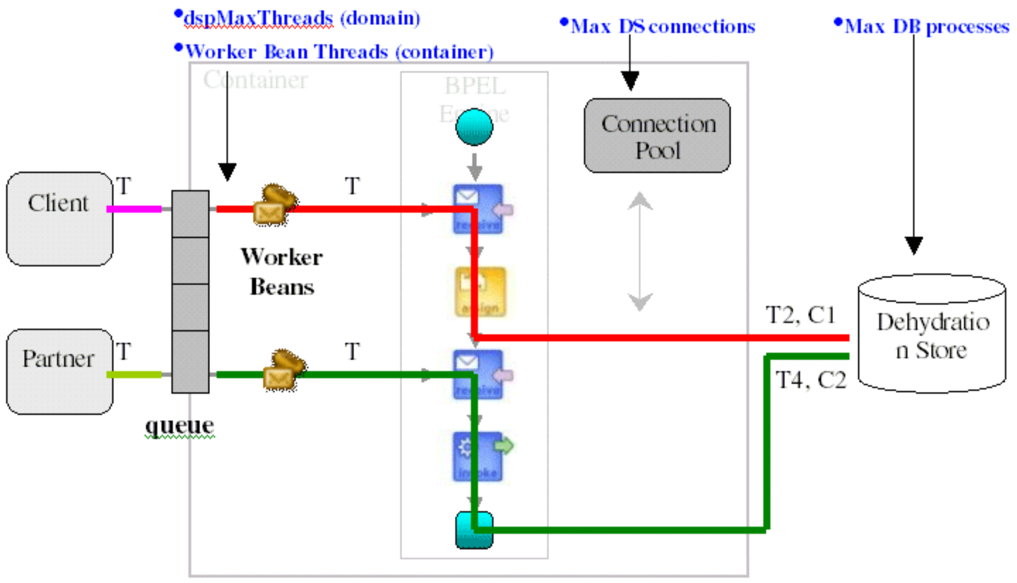
Relationship Among Some Performance Settings

From reviewing the Oracle BPEL Process Manager performance settings in *Oracle Application Server Performance Guide*, you can derive some important relationships that can be used to properly set threading and connection pooling parameters. These parameters include:

- `dspMaxThreads` (domain level)
- WorkerBean threads (OC4J container level)
- Maximum data source connections (OC4J container level)
- Maximum database connections (database level)

Figure 10–3 provides an overview.

Figure 10–3 Relationship Among Some Performance Settings



Using the File Adapter to Process Large File Sizes

This section describes the best practices for using the file adapter to process large file sizes.

This section contains the following topics:

- Overview
- Use Case and Requirements
- Initial Design and Problems

- Diagnosis
- Best Practices
- Final Numbers

See Also: *Oracle SOA Suite New Features* for information about processing large payloads in release 10.1.3.3:

`http://www.oracle.com/technology/products/ias/bpel/pdf/10133technotes.pdf`

Overview

The best practices described in this section are derived from a real world application. A medical service company created a BPEL process to send medical records from batch files to an Oracle AQ message queue. With the initial design and configuration, the company encountered serious performance problems with large file sizes (1.8M). After applying some simple best practices, the total elapsed time to process the same file was reduced from 1.5 hours to 2 minutes.

The use case and the requirements are first outlined. The original design and configuration that resulted in poor performance are then described. The best practices used to achieve the dramatic performance improvements are then illustrated.

Note: Performance numbers described in this chapter were achieved on the hardware configuration at the customer site. Depending upon your hardware configuration, your numbers can vary.

Use Case and Requirements

The customer needs a BPEL process to receive data from an inbound file, parse it into multiple records of patients, and enqueue the records into an AQ message queue for a downstream application to process further. The inbound file contains multiple lines of fixed length. These lines are of different types, as signified by the first two characters at the beginning of the lines.

Initial Design and Problems

The initial design of the BPEL process reads in the whole file as one message. Even though one inbound file contains multiple patient records, only one BPEL process instance is initiated per inbound file and one message contains all the patient records.

After reading the whole file as one message, the BPEL process traverses the message element by element through a while loop to programmatically construct patient records. After forming a patient record, the BPEL process transforms the data to the input variable of the outbound AQ adapter and enqueues the data to the outbound AQ message queue.

However, upon testing this process with a source file of about 1.8 MB with around 240 records, the following behaviors are observed:

- The BPEL process instance does not consistently appear in Oracle BPEL Control.
- It takes the BPEL process 1.5 hours to complete on a 4 GB RAM Solaris host.
- The process either runs out of memory or inconsistently stops with a transaction timeout.

Diagnosis

Two major problems with the initial design were identified:

- Multiple lines and records within the inbound file are processed sequentially. For an 1.8M incoming file that contains 2000 lines, it takes 2000 sequential iterations to finish processing the file.
- Unnecessarily large volumes of data are saved to the dehydration database. At the end of the while loop, an assign activity updates a field of the input variable, which stores the entire incoming file. As the default configuration dictates, Oracle BPEL Server saves the updated input variable into the audit trail tables of the dehydration database. For an 1.8M incoming file containing 2000 lines or 240 records, the Oracle BPEL Process Manager process saves a 1.8M payload to the audit trail tables 2000 times. This severely slows performance.

Best Practices

The following best practices are applied to address the performance problems. As a result, the performance dramatically improves.

- Debatch the Incoming File with a Proper Native Schema
- Set the Message Publish Size to a Proper Value
- Avoid Unnecessary Updates of Large Size Payloads

Debatch the Incoming File with a Proper Native Schema

The file adapter is responsible for detecting the arrival of the inbound file at the designated directory. It also passes the inbound file to a translator that uses a native schema file to translate the native format of the inbound file to XML messages. When doing so, there is an option on whether to debatch the data in the inbound file.

Debatching means parsing the data in the inbound file into multiple messages and publishing the messages to Oracle BPEL Server to initiate the BPEL process. Doing so enables Oracle BPEL Server to process multiple messages concurrently and improve the overall throughput. The original design did not debatch the data.

To perform debatching, you must first understand the structure of the inbound file, then accordingly define a native schema that translates the inbound file into multiple XML messages to send to Oracle BPEL Process Manager.

Structure of the Inbound Data File The inbound file contains lines of a fixed length that form records of multiple types:

- One header record at the beginning of the file
- One trailer record at the end of the file
- One or more patient records between the header record and trailer record

The header record consists of one line at the beginning of the file, signified by the first two characters: `H<space>`.

Similarly, the trailer record consists of one line at the end of the file, signified by the first two characters: `T<space>`.

Patient records are formed by multiple lines of different types between the header and trailing records. Each of these lines starts with a two-character code that signifies the type of line. For example, `P<space>` means `Patient Details`, `C<space>` means `Charge`, `DC` means `Diagnosis Code`, and so on. The details of the record types definitions are shown in Table 10–2:

Table 10–2 Details of the Record Types Definitions

Record Type	Type Code
Patient Demographics Record	`P<space>`
Insurance 1 Data Record	`M1`
Insurance 2 Data Record	`M2`
Insurance 3 Data Record	`M3`
Guarantor Data Record	`G<space>`
Charge Record	`C<space>`
Query Response Record	`Q<space>`
Diagnosis Code Record	`DC`

Lines of different types form a patient record. Each patient record begins with the patient detail line and ends with a diagnosis code line. Lines are in deterministic order by types. In other words, the `G` type line is always followed by a `C` type line, and so on. Among these line types, the `C` type is the only type of line that may repeat.

The following syntax shows a sample inbound file.

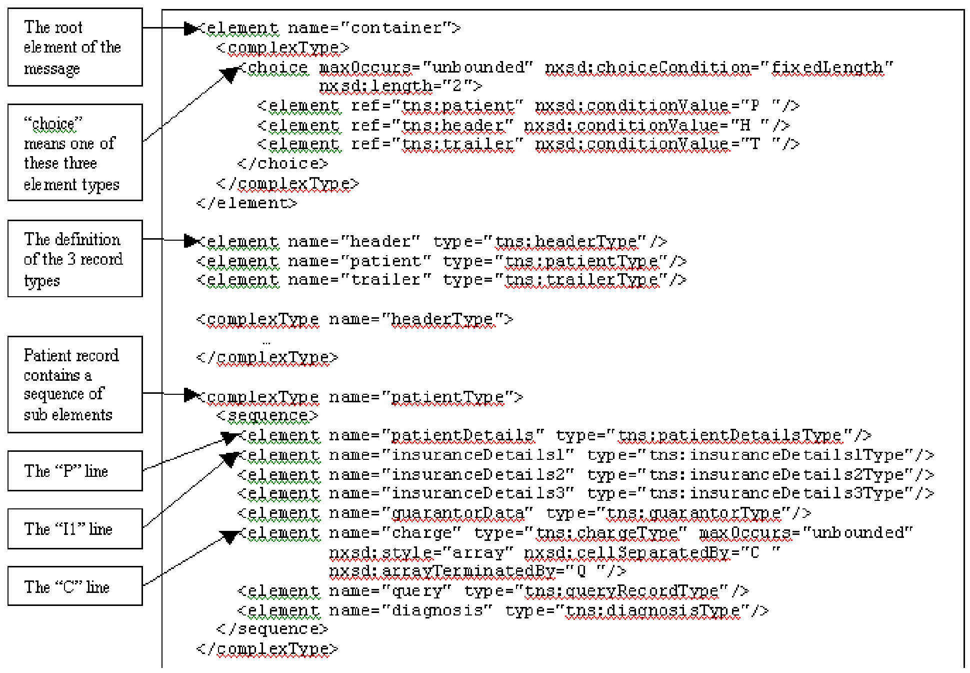
```
H 43432345United Memorial Hospital
P V 074781324        I 390451783243432301EVANS, TERRY
M1074781324        ACMHMO   ACME HEALTH HMO             PO BOX 12234
M2074781324
M3074781324
G 074781324        EVANS, TERRY             1564 North 23RD AVE
C V 074781324        390451786423459    EVANS, TERRY
Q 074781324
DC074781324        445.2 STOMACH DISORDER
P V 924001233        I   2467873280032102JONES,JOSEPH H
M1924001233        ACMPPO        ACME PPO                   P O BOX 13319
M2924001233
M3924001233
G 924001233        JONES,JOSEPH H                    100 2ND STR
C V 924001233      39045178420757                JONES,JOSEPH H
C 924001233        39045178423459                JONES,JOSEPH H
Q 924001233
DC924001233        201.0  STOMACH CARDIA
T
```

where:

- The line beginning with `H 43432345` is the header.
- Lines `P V 074781324` through `DC074781324` represent the first patient.
- Lines `P V 924001233` through `DC924001233` represent the second patient.
- The line consisting of `T` represents the trailer.

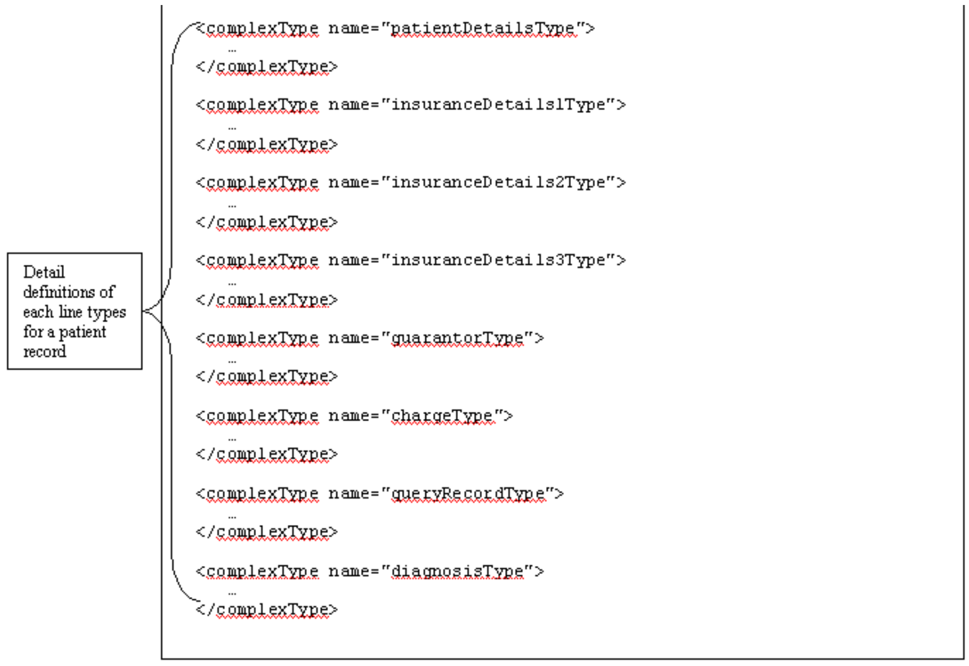
Define the Native Schema File Based on the knowledge of the inbound file structure, a native schema file is created to translate the inbound file to XML messages, as shown in Figure 10–4 and Figure 10–5.

Figure 10–4 Native Schema File (Upper Part)


The root element of the message
"choice" means one of these three element types
The definition of the 3 record types
Patient record contains a sequence of sub elements
The "P" line
The "I1" line
The "C" line

```
<element name="container">
  <complexType>
    <choice maxOccurs="unbounded" nxsd:choiceCondition="fixedLength"
            nxsd:length="2">
      <element ref="tns:patient" nxsd:conditionValue="P "/>
      <element ref="tns:header" nxsd:conditionValue="H "/>
      <element ref="tns:trailer" nxsd:conditionValue="T "/>
    </choice>
  </complexType>
</element>

<element name="header" type="tns:headerType"/>
<element name="patient" type="tns:patientType"/>
<element name="trailer" type="tns:trailerType"/>

<complexType name="headerType">
  ...
</complexType>

<complexType name="patientType">
  <sequence>
    <element name="patientDetails" type="tns:patientDetailsType"/>
    <element name="insuranceDetails1" type="tns:insuranceDetails1Type"/>
    <element name="insuranceDetails2" type="tns:insuranceDetails2Type"/>
    <element name="insuranceDetails3" type="tns:insuranceDetails3Type"/>
    <element name="guarantorData" type="tns:guarantorType"/>
    <element name="charge" type="tns:chargeType" maxOccurs="unbounded"
             nxsd:style="array" nxsd:cellSeparatedBy="C "
             nxsd:arrayTerminatedBy="Q "/>
    <element name="query" type="tns:queryRecordType"/>
    <element name="diagnosis" type="tns:diagnosisType"/>
  </sequence>
</complexType>
```

Figure 10–5 Native Schema File (Lower Part)



```
<complexType name="patientDetailsType">
  ...
</complexType>

<complexType name="insuranceDetails1Type">
  ...
</complexType>

<complexType name="insuranceDetails2Type">
  ...
</complexType>

<complexType name="insuranceDetails3Type">
  ...
</complexType>

<complexType name="guarantorType">
  ...
</complexType>

<complexType name="chargeType">
  ...
</complexType>

<complexType name="queryRecordType">
  ...
</complexType>

<complexType name="diagnosisType">
  ...
</complexType>
```

Sample XML Message At runtime, the file adapter calls the native schema to translate the data in the inbound file into one the following three types of messages:

- Header record — The message contains the `<header/>` element as the only child element of the root element. The following XML code shows one header record.

```
<part name="container">
  <container>
```

```
      <header>
        <C2>43432345</C2>
        <C3>United Memorial Hospital </C3>
        <C4/>
        <C5/>
        <C6/>
        <C7/>
        <C8/>
      </header>
    </container>
  </part>
```

- Patient record — The message contains the `<patient/>` element as the only child of the root element. And the patient element contains children elements such as `<patientDetails/>`, `<insuranceDetails1/>`, `<insuranceDetails2/>`, `<insuranceDetails3/>`, and `<charge/>` that reflect the various line types that form the patient record. The following XML code shows one patient record.

```
  <part name="container">
    <container>
      <patient>
        <patientDetails>
          <patientRecordType>VI </patientRecordType>
          <patientAccountPrefix>34</patientAccountPrefix>
          <patientAccountNumber>4325611 I </patientAccountNumber>
          <patientStatus> 24</patientStatus>
          <patientRestData>67873280032102JONES,JOSEPH H 100 2ND STR
 GREENVILLE FL45326 M1294545932 1194323627534509234USA COOK
46H34.1322Steinfort,
 James SMITH V32342FLM09345169 3005023561150751 201.0 VI.8N 64560311 M
 JONES,MARY 100 2nd Str GREENVILLE FL45326 SP
 12843430200934563744UNK,UNVAILABLE . UNKNOWN ZH10004 UNK 1904599345 REXRTRXV .
 UNKNOWN ZH10004 8858345122ACME PPO 21 35050234 ESOPHAGOGASTRECTOMY
 </patientRestData>
        </patientDetails>
        <insuranceDetails1>
          <insurance1RecordType>M1</insurance1RecordType>
          <insurance1AccountNumber>924001233 </insurance1AccountNumber>
          <insurance1Mnemonic>ACMPPO </insurance1Mnemonic>
          <insurance1Name>ACME PPO </insurance1Name>
          <insurance1RestData>P O BOX 13319 COLUMBUS KY904143561 USA
 3499816490 JONES,MARY SP 64560311 59834471 56621 54527345619003
 G43570566709 31089453JONES,MARY 100 2nd Str GREENVILLE FL45326
 F30200873</insurance1RestData>
        </insuranceDetails1>
        <insuranceDetails2>
          <insurance2RecordType>M2</insurance2RecordType>
          <insurance2AccountNumber>924001233 </insurance2AccountNumber>
          <insurance2Mnemonic/>
          <insurance2Name/>
          <insurance2RestData> 64560311 </insurance2RestData>
        </insuranceDetails2>
        <insuranceDetails3>
          <insurance3RecordType>M3</insurance3RecordType>
          <insurance3AccountNumber>924001233 </insurance3AccountNumber>
          <insurance3Mnemonic/>
          <insurance3Name/>
          <insurance3RestData> 64560311 </insurance3RestData>
        </insuranceDetails3>
        <guarantorData>
```

```
          <guarantorRecordType>G </guarantorRecordType>
          <guarantorAccountNumber>924001233 </guarantorAccountNumber>
          <guarantorName>JONES,JOSEPH H </guarantorName>
          <guarantorStreet>100 2nd Str </guarantorStreet>
          <guarantorStreet2/>
          <guarantorCity>GREENVILLE </guarantorCity>
          <guarantorState>FL</guarantorState>
          <guarantorZip>45326 </guarantorZip>
          <guarantorCountry1>COOK </guarantorCountry1>
          <guarantorCountry2>USA </guarantorCountry2>
          <guarantorPhone>2548974341 </guarantorPhone>
          <guarantorEmpName>REXRTRXV </guarantorEmpName>
          <guarantorEmpAddress1>. </guarantorEmpAddress1>
          <guarantorEmpAddress2/>
          <guarantorEmpCity>UNKNOWN </guarantorEmpCity>
          <guarantorEmpState>DD</guarantorEmpState>
          <guarantorEmpZip>32105 </guarantorEmpZip>
          <guarantorEmpPhone>3428286759</guarantorEmpPhone>
          <guarantorSSNum>659349034</guarantorSSNum>
          <guarantorRelationship>FA </guarantorRelationship>
          <guarantorUnitNum>64560311 </guarantorUnitNum>
          <guarantorFiller/>
          <guarantorEOR/>
        </guarantorData>
        <charge>
          <chargeAccountPrefix>VI </chargeAccountPrefix>
          <chargeAccountNumber>924001233 </chargeAccountNumber>
          <chargeServiceDate>39045178</chargeServiceDate>
          <chargeRestData>420757 JONES,JOSEPH H SMITH Steinfort, James 9992
Williams Avenue, Orlando, FL45326 E45871 FLFX9834532 78678 346 2 64560311
MMX.P 78678 METABOLIC PANEL </chargeRestData>
        </charge>
        <charge>
          <chargeAccountPrefix>VI </chargeAccountPrefix>
          <chargeAccountNumber>924001233 </chargeAccountNumber>
          <chargeServiceDate>39045178</chargeServiceDate>
          <chargeRestData>423459 JONES,JOSEPH H SMITH Steinfort, James 9992
Williams Avenue, Orlando, FL45326 E45871 FLFX9834532 75435 346 2 64560311
MMX.P 75435 CBC PLATELET AUTO DIFF </chargeRestData>
        </charge>
        <query>
          <queryType>34</queryType>
          <queryAccountNumber>4325611 </queryAccountNumber>
          <query1/>
          <query2/>
          <queryRestData/>
        </query>
        <diagnosis>
          <diagnosisRecordType>DC</diagnosisRecordType>
          <diagnosisAccountNumber>924001233 </diagnosisAccountNumber>
          <diagnosisCode1>201.0 </diagnosisCode1>
          <diagnosisName1>STOMACH CARDIA </diagnosisName1>
          <diagnosisRestData>8943 AIRWAY OBSTRUCT 657.23
RESPIRATORY FAILURE 672.3 PNEUMONIA 625.7 EMPYEMA
671.5 PLEURAL EFFUSION 334.0 VENTRIC TACHYCARD 561.0
HEART FAILURE NOS 345.9 KLEBSIELLA INFECTION </diagnosisRestData>
        </diagnosis>
      </patient>
    </container>
  </part>
```

- Trailer record — The message contains the `<trailer/>` element as the only child element of the root element. Because the `T` line in the inbound file does not contain any data, the children elements of `<trailer/>` contain no data either.

  The following XML code shows one trailer record.

  ```
  <part name="container">
    <container>
      <trailer>
        <footer/>
      </trailer>
    </container>
  </part>
  ```

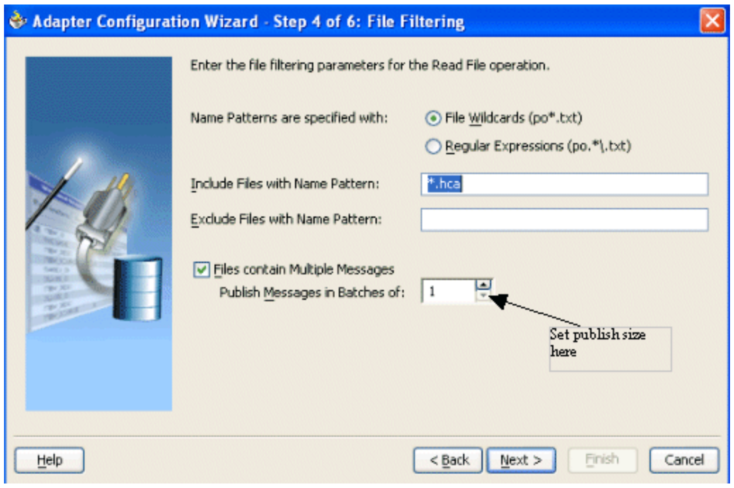
Set the Message Publish Size to a Proper Value

The publish size defines the communication between the adapter and Oracle BPEL Server. The adapter behaves as an Oracle BPEL Process Manager client. Once it parses the file into one or more messages, it has an option to group the messages together before publishing to Oracle BPEL Server. A publish size of `5` means the file adapter groups five records together and publishes it to Oracle BPEL Server. Though these five messages are grouped and sent to Oracle BPEL Process Manager in one batch, Oracle BPEL Server still treats these as five messages.

Try several different values and test the performance before identifying the optimal value. However, this parameter must be set to at least `1` for debatching to function.

Figure 10–6 shows where you set the publish size in the file adapter selection of the Adapter Configuration Wizard.

Figure 10–6 Setting Publish Size



Enter the file filtering parameters for the Read File operation.

Name Patterns are specified with: [✓ File Wildcards (po*.txt)] Regular Expressions (po.*\.txt)

Include Files with Name Pattern: *.hca

Exclude Files with Name Pattern:

[✓ Files contain Multiple Messages]

Publish Messages in Batches of: 1

Avoid Unnecessary Updates of Large Size Payloads

Because debatching only handles one record (either patient, header, or trailer) per Oracle BPEL Process Manager instance, a variable for tracking the number of patients is no longer required. For the same reason, the size of the payload dramatically decreases. Both eliminate the problem of saving a large size payload to the dehydration database multiple times.

Final Numbers

After applying these best practices, the 1.8M inbound file that previously took 1.5 hours to process now only takes 2 minutes.

In Oracle BPEL Control, 246 Oracle BPEL Process Manager instances are shown. That includes 242 patient records, 1 header record, and 1 trailer record.

Configuring Your Environment to Optimally Process Large Payloads

This section describes how to configure your environment to optimally process large payloads.

This section contains the following topics:

- Functional Requirements
- Performance Configuration

See Also: *Oracle SOA Suite New Feature* for information about processing large payloads in release 10.1.3.3:

`http://www.oracle.com/technology/products/ias/bpel/pdf/10133technotes.pdf`

Functional Requirements

A customer has a J2EE application to construct an XML payload and initiate a BPEL process through the Oracle BPEL Process Manager Java API. The customer must configure their environment to optimally process large payloads.

Performance Configuration

This section describes how to configure your environment to optimize performance.

This section contains the following topics:

- Optimize the Message Sending Application
- Optimize the BPEL Server
- Create a BPEL Cluster
- Optimize the BPEL Process
- Optimize the Dehydration Store Database

Optimize the Message Sending Application

Employ a thread pooling mechanism to throttle the number of concurrent requests sent to Oracle BPEL Process Manager.

Optimize the BPEL Server

- Increase the JVM heap size.

  With a large payload (1M or above) and high concurrency, the response time for each request decreases. One reason for this is that multiple requests are competing for memory. Therefore, increasing the JVM heap size becomes a viable method for improving performance.

  The maximum JVM heap size to which to increase depends on the size of the actual physical memory, the CPU/operating system.

The physical size of the CPU is an obvious limitation. In many Linux systems, the addressable memory is also limited by CPU ability, in addition to the physical memory size. For example, in a 32-bit Linux system, each CPU can only claim up to 2-Gigabytes of addressable memory from the physical memory, even though the physical memory may be bigger. On Solaris, the size of the addressable memory is the same as the size of the physical memory.

To increase the heap size for the OC4J instance on which Oracle BPEL Process Manager runs, you must modify an XML block similar to the following in the *IAS_ORACLE_HOME*/opmn/conf/opmn.xml file:

```
  <process-type id="OC4J_SOA" module-id="OC4J">
     <module-data>
        <category id="start-parameters">
           <data id="java-options" value="-server
 -Djava.security.policy=/scratch/fip/iAS10.1.2Midtier/j2ee/OC4J_
BPEL/config/java2.policy -Djava.awt.headless=true -Doc4j.userThreads=true
-Xbootclasspath^/p:/scratch/fip/iAS10.1.2Midtier/integration/orabpel/lib/orabpe
l-boot.jar:/scratch/fip/iAS10.1.2Midtier/integration/orabpel/lib/connector15.ja
r
-server -Xms256M -Xmx1024M -Doracle.mdb.fastUndeploy=60
-Dorabpel.home=/scratch/fip/iAS10.1.2Midtier/integration/orabpel
-Dhttp.proxySet=true -Dhttp.proxyHost=www-proxy.us.oracle.com
-Dhttp.proxyPort=80
-Dhttp.nonProxyHosts=localhost|stacx56.us.oracle.com|*.us.oracle.com|stacx56"/>
<data id="oc4j-options" value="-properties"/>
        </category>
. . .
. . .
```

- Modify the syncMaxWaitTime value in the domain.xml file.

  If the server is overloaded, it throws timeout errors to the client (application) when it takes longer than the value of syncMaxWaitTime to process the request.

  Depending on the requirements of the application, you may choose to increase or decrease the value of syncMaxWaitTime.

  Set this parameter through **Oracle BPEL Control** > **Manage BPEL Domain** > **Configurations**, or edit the domain.xml file. The latter action requires you to restart Oracle BPEL Server.

- Increase the transaction-timeout value in the server.xml file.

  For a synchronous BPEL process, a transaction is started once the process is initiated. This transaction ends when the process instance finishes. This prevents Oracle BPEL Server from erring out when it takes a long time to save data to the dehydration store due to high concurrency.

- Set the audit level value to minimum.

  When the audit level is set to minimum or off, the audit trail saves the types of events in the BPEL process to the dehydration store, but not the actual payload. Therefore, it saves some bandwidth between Oracle BPEL Server and the dehydration store database.

  Set the value of the **auditLevel** performance property under **Oracle BPEL Control** > **Manage BPEL Domain** > **Configurations**.

Create a BPEL Cluster

When CPU and memory are highly utilized (for example, above 70%), consider creating a BPEL cluster by adding another node.

Optimize the BPEL Process

This section describes the objectives and best practices for optimizing the BPEL process.

Objectives When the size of the payload is large, the parsing, copying, and transforming of payload data becomes very expensive, as is saving the data to the dehydration store database.

Best Practices

- Use Oracle XDK as parser

  If you use Oracle's internal parser, or XDK, Oracle BPEL Server can yield better performance in XML transformations, especially in the case of large payloads. Internal testing shows that the average response time for the transformation of the 1M payload is reduced from 20 seconds to 16 seconds in the case of a heavy load.

  In the 10.1.3 release of Oracle BPEL Process Manager, XDK is used as the default XML parser.

  For the 10.1.2 release, please follow these steps to enable the Oracle BPEL Process Manager transformation to use Oracle XDK.

  – Edit the `TjenesteService.bpel` file to add the namespace definition for the `xdk` prefix:

  ```
  <process name="TjenesteService"
  targetNamespace="http://xmlns.ic.dsb.dk/TjenesteService"
  . . .
  xmlns:ora="http://schemas.oracle.com/xpath/extension"
  xmlns:xdk="http://schemas.oracle.com/bpel/extension/xpath/function/xdk">
  ```

  – Edit the `TjenesteService.bpel` to replace the XPath function `ora:processXSLT()` with `xdk:processXSLT()`. This makes Oracle XDK the XML parser for transformations.

  ```
  <copy>
     <from expression="xdk:processXSLT('trftjeneste2ZZHRLTD2_
  TJENESTE01.xsl',bpws:getVariableData('inputVariable','tjenesteListe'))"/>
     <to variable="inpZZHRLTD2_TJENESTE01" part="input_ZZHRLTD2_TJENESTE01"/>
  </copy>
  ```

- If using 10.1.2, edit the `trftjeneste2ZZHRLTD2_TJENESTE01.xsl` file to work around a known bug in the XDK parser.

  – Replace the XPath function `format-string()` with `formatString()`.

  – Replace the XPath function `current-data()` with `currentDate()`.

  For example, change the following:

  ```
  <xsl:value-of
  select="orcl:format-string(&quot;{0}{1}{2}&quot;,
   substring(xp20:current-date ate(),1.0,4.0),
     substring(xp20:current-date(),6.0,2.0),
    substring(xp20:current-date(),9.0,2.0))"/>
  ```

to:

```
<xsl:value-of
select="orcl:formatString(&quot;{0}{1}{2}&quot;,
 substring(xp20:currentDate(),1.0,4.0),
   substring(xp20:currentDate(),6.0,2.0),
 substring(xp20:currentDate(),9.0,2.0))"/>
```

- Set performance properties in `bpel.xml`.
  - `completionPersistPolicy=faulted` and `inMemoryOptimization=on`

    Setting `completionPersistPolicy` to `faulted` allows Oracle BPEL Server to only save the states of faulted process instances to the dehydration store. This property is used in conjunction with the `inMemoryOptimization` property, which must be set to `on` for this case.

    In internal testing, the response time under a heavy load improved 20% after setting both properties to the proposed values.

    For example:

```
<BPELSuitcase>
  <BPELProcess id="TjenesteService" src="TjenesteService.bpel">
   <partnerLinkBindings>
   . . .
   </partnerLinkBindings>
   <configurations>
     <property name="inMemoryOptimization">true</property>
     <property name="completionPersistPolicy">faulted</property>
   </configurations>
   </BPELProcess>
</BPELSuitcase>
```

Optimize the Dehydration Store Database

This section describes the objectives and best practices for optimizing the dehydration store database.

Objectives With a large payload, the dehydration store database can fill up quickly. On the other hand, saving the large payload to the database (if this is needed) is also costly.

Best Practices

- Increase the tablespace size to prepare room for the dehydration of a large payload. For example:

```
alter tablespace orabpel add datafile
 '/oracle/product/oradata/orcl/orabpel02.dbf'
size 1024M
autoextend on
next 512M
maxsize 12040M;
```

- Recreate the `orabpel` schema to fix the percentage increase for tables.
  - Acquire the latest `domain_oracle.ddl` from Oracle Support Services.
  - Run the script as user `orabpel` against the dehydration store database.

Benchmark and Architecture

This section describes frequently asked questions about Oracle BPEL Process Manager benchmarks and architecture.

This section contains the following topics:

- What is a Typical Throughput (Transaction Rate) for Oracle BPEL Server?
- What is the Maximum Number of Activities in a BPEL Process?
- Is Oracle BPEL Process Manager Designed to Support Synchronous Processes?

What is a Typical Throughput (Transaction Rate) for Oracle BPEL Server?

A typical throughput is 100 instances per second. This throughput is based on the following setup:

- A typical modern 2-CPU computer with 2G of memory
- A simple 2-3 step asynchronous BPEL process without transformations
- Messages less than 100K in size

What is the Maximum Number of Activities in a BPEL Process?

Up to 10,000 activities have been tested. In addition, audit trail features (like lazy loading in Oracle BPEL Control, compressing large while loops, and so on) can handle large audit trails. Realistically, however, once a process gets into the thousands of activities, it is best to split this many activities into multiple processes.

The `*SOA_ORACLE_HOME*\bpel\samples\tutorials\701.LargeProcesses` sample enables you to test this feature. Specify values between 7 (1600 activities) and 10 (3000+ activities) to see instances with several thousand activities.

For processes that do not have dynamic activities (for example, looping, flowN, and so on), hundreds of activities is probably a realistic limit from a developer manageability perspective.

Is Oracle BPEL Process Manager Designed to Support Synchronous Processes?

Oracle BPEL Process Manager is designed to support both synchronous and asynchronous processes. The difference between synchronous process and asynchronous processes are only the interfaces (one with output and another without).

Clustering, Load Balancing, Failover, and Recovery

This section describes frequently asked questions about clustering, load balancing, failover, and recovery.

This section contains the following topics:

- Do I Use Oracle Application Server Clustering or Oracle BPEL Process Manager Clustering?
- How Does a Oracle BPEL Process Manager Cluster Work with DCM/OPMN?
- I Have Problems Clustering Oracle BPEL Process Manager on Top of an Oracle Application Server Cluster
- Can I Cluster an Oracle BPEL Process Manager Process to Two Domains of the Same Oracle BPEL Server, Instead of Two Oracle BPEL Servers?

- Can I Add New Nodes to Oracle BPEL Process Manager Clustering without Shutting Down the Dehydration Database and other Oracle BPEL Process Manager Nodes?
- Must I Trade Performance with Reliability?
- When a BPEL Node Crashes, How Do the Other Nodes Get the Instance and Continue
- How Do I Perform a Manual Recovery?
- Where Do Incoming Messages That Do Not Invoke BPEL Processes or Display in Oracle BPEL Control Go?
- What Causes the Asynchronous Invoke or Callback Messages to Permanently Remain at the Recovery page of Oracle BPEL Control?
- Do Synchronous Messages Display in the Recovery Page?

Do I Use Oracle Application Server Clustering or Oracle BPEL Process Manager Clustering?

Oracle Application Server clustering does not cluster Oracle BPEL Process Manager. Oracle Application Server cluster and Oracle Application Server clustering provide different ways to implement clustering.

How Does a Oracle BPEL Process Manager Cluster Work with DCM/OPMN?

Oracle Application Server clustering does not cluster Oracle BPEL Process Manager. You must use Oracle BPEL Process Manager deployment tools instead of DCM/OPMN to deploy Oracle BPEL Process Manager processes. You must separately deploy your Oracle BPEL Process Manager process to each node of the Oracle BPEL Process Manager cluster.

I Have Problems Clustering Oracle BPEL Process Manager on Top of an Oracle Application Server Cluster

See the *Oracle Application Server Enterprise Deployment Guide* for instructions.

Can I Cluster an Oracle BPEL Process Manager Process to Two Domains of the Same Oracle BPEL Server, Instead of Two Oracle BPEL Servers?

No. The full path of a BPEL process includes the domain name, process name, and version extension. Processes differ on any one of these three elements and are considered different. Therefore, they cannot be clustered.

Can I Add New Nodes to Oracle BPEL Process Manager Clustering without Shutting Down the Dehydration Database and other Oracle BPEL Process Manager Nodes?

Yes. You only need to configure the new node to point to the existing dehydration database.

In 10.1.0.2, you must copy the BPEL suitcase file (the jar file under `orabpel/domains/`*`domain_name`*`/deploy`) from the existing nodes to the new node.

In all versions, you must ensure that the process design does not include hard coded values (such as host name) that are only suitable for a particular node.

Must I Trade Performance with Reliability?

In most cases, you only trade monitorability and trackability with performance. You do not need to trade reliability with performance.

In the case of a transient process, dehydration only occurs after the process completes. That means there is only one dehydration point, and it is after the process is finished. Therefore, dehydrating a transient process does not help reliability. That is why transient processes are good candidates for setting `inMemeoryOptimization=true` and turning off or reducing the `completionPersistPolicy` value. By doing so, you are only losing monitorability, not reliability.

In the case of durable processes, dehydration occurs inflight in the middle of the process. You *must* have dehydration on. You are *not* allowed to do the following:

- Set `inMemoryOptimization=true`
- Set `completionPersistPolicy=false`

In other words, you are *not* allowed to trade reliability with performance.

There is one small case where you can trade performance with reliability. That is when you set `deliveryPersistPolicy=on` or `off` for two-way invocations (asynchronous process). If the server crashes, all the unprocessed messages are lost.

When a BPEL Node Crashes, How Do the Other Nodes Get the Instance and Continue

It depends on the location of the BPEL process.

- If the BPEL process instance is waiting at the midprocess receive activity when the BPEL node crashes, the process instance is not in an active JTA transaction. The process instance remains at the dehydrated state until the message it is waiting to receive arrives. During this moment, no BPEL nodes need to pick up this instance. Upon the arrival of the message, Oracle BPEL Server on the remaining node retrieves the instance from the dehydration store and continues to process it.
- If the BPEL process is in the middle of an active JTA transaction when the server crashes, the transaction gets rolled back. If this is an asynchronous invocation, the message gets rolled back to the dehydration store with the state unchanged.

  In this case, a manual recovery must be performed. See "How Do I Perform a Manual Recovery?" on page 10-21.

If this is a synchronous invocation, the client receives errors and is responsible for resubmitting the message.

If the process instance is waiting at the wait activity, then you must update the timer table of one of the nodes so that the wait activity awakens on that node. Otherwise, even after the timer expires, none of the nodes complete the wait activity until a manual recovery is performed.

How Do I Perform a Manual Recovery?

Go to Oracle BPEL Control, click the **Process** tab, and go to the **Perform Manual Recovery** link in the lower left corner. Click this link to display pending instances that require recovery. The pending instances are categorized by invoke messages, callback, and activity, as a reflection of various pending situations.

Where Do Incoming Messages That Do Not Invoke BPEL Processes or Display in Oracle BPEL Control Go?

For an asynchronous BPEL process, these messages do not disappear. Instead, they can be found and recovered by going to Oracle BPEL Control and select **Process** > **Perform Manual Recovery**. The disappeared messages display under the **Invoke** tab. You can recover these messages from this Oracle BPEL Control page.

What Causes the Asynchronous Invoke or Callback Messages to Permanently Remain at the Recovery page of Oracle BPEL Control?

Oracle BPEL Process Manager includes a delivery service that intercepts incoming messages. Once it intercepts an incoming message, the delivery service does two things:

- Places a very short JMS message in the in-memory queue (`jms/collaxa/BPELWorkerQueue`) in OC4J JMS. The small JMS message triggers the WorkerBean in the downstream step.
- Saves the Oracle BPEL Process Manager message to tables in the dehydration store. The `invoke_message` and `invoke_message_bin` tables are for invoking messages (messages that instantiate processes), whereas the `dlv_message` and `dlv_message_bin` tables are for callback messages. For more information about the dehydration store schema, see *Oracle Application Server Performance Guide*.

Downstream, Oracle BPEL Server has an MDB called WorkerBean that listens on `jms/collaxa/BPELWorkerQueue`. Once triggered, the WorkerBean use the information in the small JMS message to load the full Oracle BPEL Process Manager messages from tables in the dehydration store. In the same thread, Oracle BPEL Server instantiates or continues the Oracle BPEL Process Manager instance. By the time Oracle BPEL Server finishes processing this message (that is, it reaches the end of the process or reaches the first dehydration point), Oracle BPEL Server updates the message tables to mark the message as `HANDLED`.

The recovery page shows those invoke or callback messages that are *not* in the `HANDLED` state. If you click fast enough, you see messages come and go on the recovery page. Seeing messages on the recovery page does not necessarily mean there are problems, unless the messages remain there permanently. They remain there permanently because there are no additional JMS messages to trigger the WorkerBean to process the unhandled BPEL messages in the `invoke_message` and `dlv_message` tables. The following scenarios can cause BPEL messages to remain in unresolved states:

- The server shuts down or crashes before it finishes processing the BPEL message. When the server restarts again, the in-memory JMS message associated with the invoke or callback message is already gone.
- Oracle BPEL Server cannot finish processing the message before reaching the timeout dictated by the transaction timeout value specified in the `server.xml` file. In this case, it rolls back the message to the message table. However, the JMS message associated with the invoke messages is already consumed.

In these cases, no more events (that is, JMS messages) trigger Oracle BPEL Server to process the BPEL messages in the `invoke_message` or `dlv_message` tables. Therefore, a manual recovery is performed.

Do Synchronous Messages Display in the Recovery Page?

Oracle BPEL Process Manager does not save synchronous processes to the `invoke_message` or `dlv_message` table. Therefore, they never appear in the recovery page of Oracle BPEL Control. In the case of a server shutdown, crash, or transaction time out, errors or exceptions are returned to the caller. The caller can decide the action to take.

If the caller is an adapter (for example, the JMS adapter), the rejected message is handled by a configurable rejection handler. The default rejection handler puts the rejected messages into a file under the `domain/`*`domain_name`* directory.

> **See Also:** The following documentation for details about rejection handlers:
>
> - "Rejection Handlers" on page 3-32
> - *Oracle Application Server Adapter for Files, FTP, Databases, and Enterprise Messaging User's Guide*

Stability and Memory

This section describes frequently asked questions about stability and memory.

What Causes an Out-of-Memory Error?

This may be caused by several possibilities, or a combination of them:

- JVM heap size is too small

  Modify the JVM heap size setting at *`ORACLE_HOME`*`/opmn/conf/opmn.xml`.

- Overthreading

  If the BPEL process is asynchronous, reduce the `dspMaxThread` parameter (from Oracle BPEL Control under **Manage BPEL Domain** -> **Configuration**). The default value is `100`. As a rule, do not increase this number unless the CPU and memory are underutilized. Indeed, thread counts between `40 - 60` for single CPU installations are typically recommended. There is no hard limit on the `dspMaxThread` value, but a value greater than `250` is usually considered high.

  If the BPEL process is synchronous, the client and partner calling this BPEL process controls the concurrency. Reduce the threads.

- Payload is too large

  A payload greater than 1MB is considered large. You may visibly see an impact on performance. A payload greater than 10 MB must be treated very carefully. If the payload is large, then reduce the concurrency (thread counts) to reduce memory consumption. On the other hand, there is no hard limit of the size of payload. It depends on what you do with the payload.

Performance Tuning

This section describes frequently asked questions about performance.

This section contains the following topics:

- Is a Dehydrated BPEL Process Instance Using Oracle BPEL Process Manager Resources?
- Do Synchronous Processes Dehydrate?

- Can I Improve Asynchronous Process Performance by Disabling Dehydration?
- Does Disabling completetionPersistPolicy Cause Out-Of-Memory Errors Because Everything Remains in Memory Instead of Being Dehydrated?

Is a Dehydrated BPEL Process Instance Using Oracle BPEL Process Manager Resources?

No. A dehydrated BPEL process instance is not using any system resources of Oracle BPEL Process Manager.

Do Synchronous Processes Dehydrate?

By default, all BPEL processes are dehydrated regardless of whether they are synchronous or asynchronous. However, many synchronous processes do not need to be durable. You can turn off the dehydration store by setting `inMemoryOptimizaiton` to `true` and `completetionPersistPolicy` to `faulted` or `off`.

Can I Improve Asynchronous Process Performance by Disabling Dehydration?

Yes, you can turn off the dehydration for asynchronous process, as long as the process is transient. A transient process is a process without midprocess breakpoint activities such as receive, wait, and pick.

Oracle BPEL Server always dehydrates a durable process and ignores the settings of the persistence properties (such as `inMemoryOptimization` or `completionPersistPolicy`).

However, Oracle BPEL Server always saves invocation messages of asynchronous processes to the `invoke_message` table. This cannot be changed. The performance tuning property `deliveryPersistPolicy` is deprecated in Oracle BPEL Process Manager 10.1.3.1.

Does Disabling completetionPersistPolicy Cause Out-Of-Memory Errors Because Everything Remains in Memory Instead of Being Dehydrated?

It does not make a difference. A transient process is in memory until the end of the process, regardless of the `completionPersistPolicy` setting.

Transactions Related

This section describes frequently asked questions about transactions.

If Top Level Processes Invoke Second Level Subprocesses, Can the Main Process Propagate a JTA Transaction Context to the Subprocesses?

Yes. All subprocesses are executed in the same JTA transaction as the invoke activity of the parent process as long as both of the following are true:

- All subprocesses are synchronous.
- All subprocesses are co-located in the same Oracle BPEL Server.

Dehydration Store Management

This section describes frequently asked questions about dehydration store management.

This section contains the following topics:

- The Dehydration Database is Running Out of Tablespace. Is it Safe to Clean Up the Dehydration Store?
- How to I Clean Up the Dehydration Store?
- What Should I Do If I Receive Error ORA-01691 in the domain.log File?
- What Should I Do If I Receive Error ORA-05002 in the domain.log File?

The Dehydration Database is Running Out of Tablespace. Is it Safe to Clean Up the Dehydration Store?

It is safe as long as you do not need those persisted incoming messages.

How to I Clean Up the Dehydration Store?

See `http://www.oracle.com/technology/pub/articles/bpel_cookbook/blanvalet.html` for details about purging or archiving BPEL dehydration stores.

What Should I Do If I Receive Error ORA-01691 in the domain.log File?

```
<2005-12-14 16:05:59,168> <ERROR> <default.collaxa.cube>
 <BaseCubeSessionBean::logError>
Error while invoking bean "delivery": Cannot update lob column.
The process domain was unable to update the lob column "7" in the datastore.
The exception reported is: ORA-01691: unable to extend lob segment ORABPEL.SYS_
LOB0000069158C00007$$ by 128 in tablespace ORABPEL
```

You can apply one or both of the following options:

- Increase `orabpel` tablespace size by running a script similar to the following. Note that in 10.1.3, this script is already included in the default `IRCA` tool of Oracle BPEL Process Manager.

  ```
  alter tablespace orabpel add datafile
   '/oracle/product/oradata/orcl/orabpel02.dbf'
  size 1024M
  autoextend on
  next 512M
  maxsize 12040M;
  ```

- Clean up the dehydration store (see "How to I Clean Up the Dehydration Store?" on page 10-25).

What Should I Do If I Receive Error ORA-05002 in the domain.log File?

```
<2005-12-20 17:34:42,139> <ERROR> <default.collaxa.cube.engine.dispatch>
 <BaseScheduledWorker::process>
 Failed to handle dispatch message ... exception ORABPEL-05002

Message handle error.
An exception occurred while attempting to process the message
 "com.collaxa.cube.engine.dispatch.message.invoke.InvokeInstanceMessage";
```

```
the exception is: Transaction was rolled back: timed out;
nested exception is: java.rmi.RemoteException:
No Exception - originate from:java.lang.Exception: No Exception - originate from:;
 nested exception is:
        java.lang.Exception: No Exception - originate from:
```

Possible causes and solutions are as follows:

- Poor performance of the dehydration database

  If you are using Oracle Lite as the dehydration store, move to Oracle 9*i* or 10*g*. If Oracle 9*i* or 10*g* is already in use, check the database parameters `process` and `session` to make sure they can handle the expected throughput.

- OC4J has too few available connections to the dehydration database

  Increase the `maxConnection` value of the `BPELServerDataSource` at the *`BPEL_HOME`*`/integration/orabpel/system/appserver/oc4j/j2ee/home/config/data-sources.xml` (for the developer installation) or `j2ee/OC4J_SOA/config/data-sources.xml` (for the middle-tier installation).

- Message size is too big

  There are two ways to resolve this problem:

  – Increase the transaction timeout at *`BPEL_HOME`*`/integration/orabpel/system/appserver/oc4j/j2ee/home/config/server.xml` (for the developer installation) or `j2ee/OC4J_SOA/config/server.xml` (for the middle-tier installation).

  – Decrease the **auditLevel** value in Oracle BPEL Control, which is accessible from **Manage BPEL Domain** > **Configurations**. This reduces the amount of data saved to the dehydration store.

Load Throttling

This section describes frequently asked questions about load throttling.

How Do I Throttle the Load in BPEL Processes?

You can only throttle the load of asynchronous processes. This is done by tuning the `dspMaxThread` property. This is a domain level tuning parameter and impacts the performance of all the asynchronous receive activities in the same BPEL domain.

For asynchronous processes, the Oracle BPEL Process Manager delivery service saves the invocation message to the `invoke_message` table to achieve reliable delivery of messages. In a separate thread, the WorkerBean fetches the invocation message from the table and processes it. The `dspMaxThread` property controls the size of the pool from which the WorkerBean acquires the thread to fetch and process the invocation message.

Once again, this throttling only works for asynchronous invocations (asynchronous processes).

See Also: The following documentation for additional details about `dspMaxThread`:

- Chapter 1, "Oracle BPEL Process Manager"
- *Oracle Application Server Performance Guide*

11

Oracle Human Workflow Performance

This chapter describes how to tune Oracle Human Workflow for optimal performance.

This chapter contains the following topics:

- Minimizing Client Response Time
- Improving Server Performance
- Completing Workflows Faster

Minimizing Client Response Time

Since workflow client applications are interactive, it is important to have good response time at the client. Some of the factors that affect the response time include service call overhead, querying time to determine the set of qualifying tasks for the request, and the amount of additional information to be retrieved for each qualifying task. This section discusses the various options available to address these factors:

- Choose the Right Workflow Service Client
- Narrow Qualifying Tasks Using Precise Filters
- Retrieve Subset of Qualifying Tasks (Paging)
- Fetch Only the Information That Is Needed for a Qualifying Task
- Reduce the Number of Return Query Columns
- Tune the Identity Provider
- Tune the Database

Choose the Right Workflow Service Client

Workflow services support two major types of clients: SOAP and EJB clients. EJB clients can be further separated into local EJB clients and remote EJB clients.

If the client application is based on .Net technologies, then only the SOAP workflow services can be used. However, if the client application is based on J2EE technology, then some trade-offs determine which client to use. They are listed below in increasing amount of overhead.

- Local client — This has the least amount of overhead among the different clients since there is no remote method invocation (RMI) involved. Always consider this option first. However, this option is suitable only if the client application is running on the same JVM as the workflow services (soa-infra application). Also, deploy the client application as a child of the soa-infra application.

- Remote client — This is the next best option in terms of overhead. The additional overhead in comparison with the local client is related to RMI and marshalling and unmarshalling of method arguments.
- SOAP client — While this option is preferred for standardization based on Web services, the XML serialization and deserialization increases the overhead compared to remote clients.

Narrow Qualifying Tasks Using Precise Filters

This is the most important factor in improving response time. When a task list is retrieved, the query should be as precise as possible so the maximum filtering can be done at the database level. When the inbox view is requested for a user, the tasks are filtered mainly based on whether they are assigned to the current user or to the groups the to which user belongs. By specifying additional predicate filters on the inbox view, the overall response time for the query can be reduced since a lesser number of tasks qualify. Alternatively, you can define views by specifying predicate filters and the overall response time for such views is reduced since a lesser number of tasks qualify. All predicates passed to the query APIs or defined in the views are directly pushed to the database level SQL queries and the database optimizer uses the best indexes to create an optimal execution plan. The additional filters can be based on task attributes or promoted flex fields. For example, instead of listing all purchase order approval tasks, views can be defined to present tasks to the user based on priority, date, category, or amount range.

For example, to retrieve all assigned tasks for a user with `priority = 1`, you can use the following API call:

```
Predicate pred = new Predicate(TableConstants.WFTASK_STATE_COLUMN,
                               Predicate.OP_EQ,
                               IWorkflowConstants.TASK_STATE_ASSIGNED);

pred.addClause(Predicate.AND,
               TableConstants.WFTASK_PRIORITY_COLUMN,
               Predicate.OP_EQ,
               1);

List tasks = querySvc.queryTasks(ctx,
                                 queryColumns,
                                 null,
                                 ITaskQueryService.ASSIGNMENT_FILTER_MY,
                                 null,
                                 pred,
                                 null,
                                 startRow,
                                 endRow);
```

Retrieve Subset of Qualifying Tasks (Paging)

Once the task list is narrowed down to meet a specific criteria as discussed in "Narrow Qualifying Tasks Using Precise Filters" on page 11-2, the next level of filtering is based on how many tasks to present to the user. You want to avoid fetching too many rows, which not only increases the query time but also increases the application process time and the amount of data returned to the client. The query API has paging parameters that control the number of qualifying rows returned to the user and the start row.

For example, in the `queryTasks` method:

```
List tasks = querySvc.queryTasks(ctx,
                                  queryColumns,
                                  null,
                                  ITaskQueryService.ASSIGNMENT_FILTER_MY,
                                  null,
                                  pred,
                                  null,
                                  startRow,
                                  endRow);
```

Set the `startRow` and `endRow` parameters to reasonable numbers (for example, `0` and `20`) to limit the number of return matching records.

Fetch Only the Information That Is Needed for a Qualifying Task

When using the `queryTask` service, it is recommended that you reduce the amount of optional information retrieved for each task returned in the list. This reduces the overhead from additional SQL queries and Java logic.

For example, in the following `queryTasks` method, only the group actions information is retrieved. Although you can also retrieve attachment and payload information directly in the listing, it is expensive.

```
List optionalInfo = new ArrayList();
optionalInfo.add(ITaskQueryService.TASK_ACTIONS_TYPE_GROUP_ACTIONS);

// optionalInfo.add("Attachments");
// optionalInfo.add("Payload");

List tasks = querySvc.queryTasks(ctx,
                                  queryColumns,
                                  optionalInfo,
                                  ITaskQueryService.ASSIGNMENT_FILTER_MY,
                                  null,
                                  pred,
                                  null,
                                  startRow,
                                  endRow);
```

In rare cases where the entire payload is needed, the payload information can be requested. Only some of the payload fields are typically needed for displaying the task list. For example, for purchase order tasks, the purchase order amount may be a column that needs to be displayed. Rather than fetching the payload as additional information and then retrieving the amount using an XPath expression and displaying it in the listing, by mapping the amount column from the payload to a flex field, the flex field can be directly retrieved during SQL querying. This significantly reduces the processing time.

Similarly for attachments, if the name of the attachment is to be displayed in the listing and the document itself is stored in an external repository, then it may be better to capture the attachment name in the payload and also map it to a flex field so that processing overhead is minimized. While constructing the listing information, the link to the attachment can be constructed by fetching the appropriate flex field.

Reduce the Number of Return Query Columns

When using the `queryTask` service, it is recommended that you reduce the number of query columns to improve the SQL time. Also, try to use the common columns because they are most likely indexed. Therefore, the SQL can execute faster.

For example, in the following `queryTasks` method, only the `TASKNUMBER` and `TITLE` columns are returned:

```
List queryColumns = new ArrayList();
queryColumns.add("TASKNUMBER");
queryColumns.add("TITLE");
...
List tasks = querySvc.queryTasks(ctx,
                                 queryColumns,
                                 null,
                                 ITaskQueryService.ASSIGNMENT_FILTER_MY,
                                 null,
                                 pred,
                                 null,
                                 startRow,
                                 endRow);
```

Tune the Identity Provider

The workflow service uses information from the identity provider in constructing the SQL query to determine the tasks qualifying for a user based on his or her role and group membership. The identity provider is also queried for determining role information to determine privileges of a user when fetching the details of a task and determining which actions the user can perform on a task. There are a few ways to speed up requests made to the identity provider.

- Set the search base in the identity configuration file to node(s) as specific as possible. Ideally, you should populate workflow-related groups under a single node to minimize traversal for search and lookup. This is not always possible; for example, you may need to use existing groups and grant membership to groups located in other nodes. If it is possible to specify filters that narrow down the nodes to be searched, then you should specify them in the identity configuration file.
- Index all critical attributes (such as dn, cn, and so on) in the identity provider. This ensures that when a search or lookup is done, only a subset of the nodes is traversed instead of a full tree traversal.
- Use an identity provider that supports caching. Not all LDAP providers support caching. However, Oracle Internet Directory supports caching and. Therefore, most of the lookup and search queries are faster.

Tune the Database

The workflow schema is shipped with several indexes defined on the most important columns for all the tables. Based on the type of request, different SQL queries are generated to fetch the task list for a user. The database optimizer evaluates the cost of different plan alternatives (for example, full table scan and access table by index) and decides on a plan that is lower in cost. For the optimizer to work correctly, the index statistics should be current at all times. Therefore, as with any database usage, it is important to ensure that the database statistics are updated at regular intervals and other tunable parameters such as memory, table space, and partitions are used effectively to get maximum performance.

Improving Server Performance

Server performance essentially determines the scalability of the system under heavily loaded conditions. "Minimizing Client Response Time" on page 11-1 lists a number of ways in which client response times can be minimized by fetching the right amount of information and reducing querying overhead. These techniques also reduce the database and service logic overhead at the server and therefore improve server performance. In addition, a few other recommendations can be made to improve server performance. This section describes these recommendations.

- Archive Completed Instances Periodically
- Use Java Callbacks Instead of BPEL Callbacks
- Suppress Logging and Debugging Messages
- Minimize Notification Overhead
- Cluster Nodes

Archive Completed Instances Periodically

The database scalability of a system is largely dependent on the amount of data in the system. Since business processes and workflows are temporal in nature, once they are processed they are not queried frequently. Having many completed instances in the system slows down the system. Therefore, it is recommended that an archival scheme be used to periodically (for example, quarterly, or annually) move completed instances to another system that can be used to query historical data.

Archiving must be done carefully to avoid orphan task instances.

Use Java Callbacks Instead of BPEL Callbacks

The workflow callback functionality can be used to query or update external systems after any significant workflow event, such as assignment or completion of a task. While this functionality is very useful, it has to be used wisely, keeping in mind the overhead involved.

When performance is critical, check if it is sufficient to update the external system once after the task is completed instead of doing so after every workflow event. In this case, instead of using a callback, the service can be invoked once after the completion of the task. If a callback cannot be avoided, then consider using a Java callback instead of a BPEL callback. Java callbacks do not have the overhead associated with a BPEL callback since the callback method is executed in the same thread. In contrast, a BPEL callback has the additional overhead of sending a message to the BPEL engine that in turn much be correlated so that it is delivered to the correct process instance. Also, the workflow service has to be called by Oracle BPEL Server after the invocation of the service.

Suppress Logging and Debugging Messages

When debugging is enabled, the workflow service generates many logging messages to trace parameters passed through various calls and monitors execution paths. This is useful during the prototype and testing phase of a system. However, this can generate significant overhead. If performance is critical, set the log level to a higher level to reduce the amount of messages that must be logged. You can lower the log level if an issue that requires extensive debugging is discovered.

In case of integrating with workflow services, the following log4j properties can be included in the classpath to tune the log level.

```
log4j.rootCategory=warn;

log4j.category.oracle.bpel.services=warn;
```

Minimize Notification Overhead

Notifications are useful in quickly alerting users that they have a task on which to work. In environments where most approvals happen through an e-mail, actionable notifications are really useful. This also implies that there is not much load in terms of worklist usage. However, if most users interact through the worklist, and notifications serve a secondary purpose, then in a performance-critical environment, notifications must be used judiciously. Consider minimizing the notification to just alert a user when a task is assigned instead of sending out notifications for each workflow event. Also, if the task content is also mailed in the notification, there is significant overhead. To minimize this overhead, consider making the notifications secure. In this case, only a link to the task is sent in the notification and not the task content itself.

Cluster Nodes

All workflow instances and state information are stored in the dehydration database. Therefore, workflow services are stateless. This means that they can be used concurrently on a cluster of nodes. When performance is critical and a highly scalable system is needed, a clustered environment can be used for supporting workflow.

Completing Workflows Faster

The time it takes for a workflow to complete is highly dependent on the routing type specified for the workflow. Nevertheless, the workflow functionality provides some options to use to speed up the completion of workflows as discussed in this section.

- Use Workflow Reports to Monitor Progress
- Specify Escalation Rules
- Specify User and Group Rules for Automated Assignment
- Use Task Views to Prioritize Work

Use Workflow Reports to Monitor Progress

A number of workflow reports (and corresponding views) are available that make it easier to monitor and proactively fix problems so that workflows complete faster.

- The unattended tasks report provides a list of group tasks that need attention since they have not yet been acquired by any user to work on.
- The task cycle time report provides an idea of how much time it takes for a particular type of workflow to complete.
- The task productivity report indicates the inflow and outflow of tasks for different users.
- The assignee time distribution report provides a detailed drill down of the time spent by each user during the task life cycle. This includes the time the task was idle waiting for it to be picked up by a user.

All of these reports can be used effectively to fix problems. By checking unattended tasks reports, you can assign tasks that have been in the queue for a long time to specific users. By monitoring cycle time and other statistics, you can add staff to groups that are overloaded or take a longer time to complete. Therefore, reports can be used effectively to ensure workflows complete faster.

Specify Escalation Rules

To ensure that tasks do not get stuck at any user, you can specify escalation rules. For example, you can move a task to a manager if a certain amount of time passes without any action being taken on the task. Custom escalation rules can also be inserted if the task must be escalated to some other user based on alternative routing logic. By specifying proper escalation rules, you can reduce workflow completion times.

Specify User and Group Rules for Automated Assignment

Instead of manually reassigning tasks to other users or members of a group, you can use user and group rules to perform automated reassignments. This ensures that workflows get timely attention. For example, a user can set up a user rule such that workflows of a specific type and matching a certain filter criteria are automatically reassigned to another user in a specified time window. Similarly, a group rule can be used to automatically reassign workflows to a member of the group based on different routing criteria such as round robin, most productive, and so on. Therefore, rules can help to significantly reduce workflow waiting time, which results in faster workflow completion.

Use Task Views to Prioritize Work

Your inbox can contain tasks of various types with various due dates. You must manually sift through the tasks or sort them to determine which one to work on next. Instead, by creating task views where tasks are filtered based on due dates or priority, you can get work prioritized automatically. This enables you to focus on completing tasks instead of wasting time deciding which tasks to work on. This also results in faster completion of workflows.

Index

Numerics

10.1.3.3 release enhancements
  documentation, 3-1

A

acknowledgment types, 5-2
activation agents
  initiating a BPEL process instance on a fixed interval, 1-31
activities
  maximum number in a BPEL process, 10-19
adapter framework
  adapter SDK, 3-31
  changing inbound adapter parameters at runtime, 3-27
  dynamic BPEL process routing support, 3-34
  enabling debug logging, 3-37
  inbound message flow multiplexing, 3-29
  rejectedMessageHandlers configuration, 3-32
  scheduling endpoint activation, 3-29
  throttling message flow, 3-27
  tuning the JCA connection cache, 3-30
  using JGroups configuration for clustering, 3-28
adapter SDK, 3-31
adapters
  connecting to a protocol or application for which Oracle does not have an adapter, 2-3
  supported with Oracle ESB, 2-3
agents
  connectivity to the Oracle Data Integrator repository, 7-4
  overloading in Oracle Data Integrator, 7-4
  parameters for Oracle Data Integrator, 7-3
  sizing for Oracle Data Integrator, 7-4
  starting for Oracle Data Integrator, 7-4
  testing connectivity to the data servers in Oracle Data Integrator, 7-7
ant command
  registering an Oracle ESB project from the command line, 2-9
AQ adapter
  configuring with Oracle BAM, 6-2
  replacing a design time schema with a runtime schema, 3-2
ascontrol_admin role
  description, 2-7
ascontrol_appadmin role
  description, 2-7
ascontrol_monitor role
  description, 2-7
asynchronous durable processes, 10-5
attachments
  in workflows, 4-10
automatic retry
  configuring in Oracle ESB, 2-9

B

B2B
  See Oracle B2B
BAM
  See Oracle BAM
basicHeaders property
  description, 1-39
basicPassword property
  description, 1-39
basicUsername property
  description, 1-39
BPEL flow
  forcing a rollback, 1-5
BPEL Locator API
  using, 1-20
BPEL node crashes
  how other nodes get the instance and continue, 10-21
BPEL performance
  concepts and terminology, 10-1
BPEL process configuration
  nonblocking invoke, 1-16
  not seeing synchronous requests time out, 1-14
  setting the syncMaxWaitTime property, 1-14
BPEL processes
  maximum number of activities, 10-19
  throttling the load, 10-26
BPEL subprocesses
  incurring overhead when calling, 1-27
bpelx:rollback
  forcing a rollback, 1-6, 3-4, 3-5
breakpoint activities
  dehydration occurrences, 10-4

description, 10-2

C

callbackBindings property
description, 1-39
callbacks
in workflows, 4-8
chaining
complex routing patterns in workflows, 4-8
clustering
adding new nodes to Oracle BPEL Process Manager clustering without shutting down the dehydration database and other nodes, 10-20
an Oracle BPEL Process Manager process to two domains of the same Oracle BPEL Server, 10-20
balancing performance and reliability, 10-21
configuration JGroups, 3-28
how other nodes get the instance and continue after BPEL node crashes, 10-21
Oracle BPEL Process Manager clustering on top of Oracle Application Server clustering, 10-20
performing manual recoveries, 10-21
use of DCM and OPMN in Oracle BPEL Process Manager clustering, 10-20
using Oracle Application Server clustering or Oracle BPEL Process Manager clustering, 10-20
color schemas
beautifying reports in Oracle BAM, 6-5
completionPersistLevel property
description, 1-37
completionPersistPolicy property
description, 1-37
impact of disabling on memory errors, 10-24
setting, 1-9
configuration properties
deployment descriptor, 1-37
connection pooling
FTP adapter, 3-21
connections
to a protocol or application for which Oracle does not have an adapter, 2-3
constructs
Oracle B2B interoperability with Oracle ESB and Oracle BPEL Process Manager, 5-6
contentType property
description, 1-39
contexts
in Oracle Data Integrator, 7-10
correlation property
description, 1-39

D

daemons
limitation on implementing using Oracle BPEL Process Manager, 1-30
data servers
definitions for Oracle Data Integrator, 7-6
data source connections
relationship with other performance settings, 10-7
data sources
setting up pure SQL with dynamic routing in Oracle ESB, 2-34
database adapter
configuring private ownership, 3-11
delete polling strategy, 3-10
exception misclassifications, 3-8
exceptions to look for, 3-13
explicit schema name references, 3-9
finding the root exception, 3-12
foreign key and primary key constraints, 3-10
inbound pooling gets stuck and cannot close connection exceptions, 3-6
journaling disabled, 3-13
logical delete polling strategy, 3-11
missing instances, 3-8
missing query by example support in 10.1.3, 3-9
not automatically retrying invokes after remote faults, 3-6
primary key constraints with merge, 3-11
printing thread on log messages, 3-12
scenarios where the adapter cannot survive database restarts, 3-6
setting up pure SQL with dynamic routing in Oracle ESB, 2-37
setting xADataSourceName, 3-3
tuning documentation, 3-12
XA configuration, 3-3
database connections
relationship with other performance settings, 10-7
database schemas
for Oracle Data Integrator repositories, 7-1
database sequences
versus ODI sequence in Oracle Data Integrator, 7-7
databases
scenarios under which databases are required for Oracle ESB, 2-5
setting up pure SQL with dynamic routing in Oracle ESB, 2-33
supported as a storage backend for Oracle ESB, 2-2
testing connectivity for Oracle Data Integrator, 7-6
DBMS_SCHEDULER
automating purging of ESB instance data, 2-11
debatching
notifications when debatching is done, 3-18
processing large files when using the file adapter, 3-16, 3-17
defaultInput property
description, 1-37
dehydration
adding new nodes to Oracle BPEL Process

Manager clustering without shutting down the dehydration database, 10-20
BPEL process instances use of Oracle BPEL Process Manager resources, 10-24
cleaning up the database, 10-25
cleaning up the database when running out of tablespace, 10-25
description, 10-3
disabling to improve asynchronous process performance, 10-24
impact of process types, 10-4
managing the dehydration store, 10-25
occurrences of, 10-4
of synchronous processes, 10-24
optimizing the dehydration store database, 10-18
delete polling strategy
in database adapter, 3-10
delivery service
description, 10-4
deliveryPersistPolicy property
description, 1-37
deployment
recommendations for Oracle BAM, 6-7
resolving a process lock timeout error, 1-22
deployment descriptor properties
configuration properties, 1-37
partnerLinkBinding properties, 1-39
documentation
of models and interfaces in Oracle Data Integrator, 7-9
domain-value mapping
performance in Oracle ESB, 2-18
dspInvokeAllocFactor property
description, 1-3
throttling requests, 1-3
dspMaxThreads property
description, 1-2, 1-3
relationship to ReceiverThreads, 1-1
relationship with other performance settings, 10-7
throttling requests, 1-3
durable processes, 10-5
dehydration occurrences, 10-4
description, 10-2
dynamic partner links
support in Oracle ESB, 2-14
dynamic routing
using pure SQL with, 2-32
with pure SQL in Oracle ESB, 2-32

E

e-commerce
acknowledgment types, 5-2
components, 5-2
description, 5-1
misconceptions, 5-2
Eden garbage collection
specifying an optimal value for ReceiverThreads, 1-2
e-mail notifications
sending from Oracle ESB, 2-6
embedded Java
using with BPEL, 1-36
endpoint activation
scheduling, 3-29
Enterprise Link
using Oracle BAM without, 6-8
error hospital
Oracle ESB, 2-15
error instances
resubmitting multiple error instances at the same time in Oracle ESB, 2-9
errors
cause of out-of-memory errors, 10-23
conditions that cannot be handled by a fault handler, 1-28
diagnosing in Oracle B2B, 5-8
Oracle B2B checklist, 5-10
Oracle B2B inbound direction errors, 5-9
Oracle B2B outbound direction errors, 5-9
Oracle B2B typical errors, 5-9, 5-11
resolving a process lock timeout error, 1-22
resolving error ORA-01691 in domain.log, 10-25
resolving error ORA-05002 in the domain.log file, 10-25
ESB systems
benefits of, 2-6
esb_config.ini file
runtime server global settings, 2-7
escalations
in workflows, 4-9
expiration events
scheduling with Quartz, 1-24
external data sources
configuring BAM to read data from, 6-4
external tables
reading into Oracle BAM using Enterprise Link plans, 6-4
external URLs
calling through Oracle BAM, 6-6
external Web services
calling through Oracle BAM, 6-6

F

fault handler
error conditions that cannot be handled, 1-28
file adapter
accessing Windows network folders, 3-16
appending an existing file, 3-17
avoiding unnecessary updates of large size payloads, 10-14
controlling the size of a rejected message, 3-17
creating notifications indicating that debatching is done, 3-18
debatching large XML files, 3-17
guaranteeing message order, 3-19
high availability, 3-19
losing messages when using the time-pattern in

the outbound partner link file name, 3-18
processing large file sizes, 10-7
processing large files with debatching, 3-16, 10-9
reading the file name of the generated file from a write operation in BPEL, 3-17
setting the message publish size to a proper value, 10-14
thread configuration, 3-19
filters
support in Oracle ESB, 2-8
firewall
creating an ESB-to-ESB bridge across a firewall, 2-3
flow logic
error conditions that cannot be handled with a fault handler, 1-28
receiving empty and zero node selection failure faults, 1-28
foreign key
database adapter constraints, 3-10
FTP adapter
connection pooling, 3-21
includes same functionality as file adapter, 3-21
limitation on support for ACTIVE connections mode, 3-21
partner link for an invoke operation to retry if connection-related errors exist, 3-21
fullWSAddressing property
description, 1-39

G

GetActiveUnitOfWork property
description, 3-4
global transactions
configuring the JMS adapter, 3-22
creating a JMS dequeue operation when using the JMS adapter, 3-22

H

high availability
handling a RAC node failover with Oracle BPEL Server, 1-21
setting up the file adapter, 3-19
support in Oracle ESB, 2-8
HTTP
handling inbound HTTP POSTs from Oracle ESB, 2-4
HTTP_PASSWORD property
setting, 2-9
HTTP_USERNAME property
setting, 2-9
httpAccept property
description, 1-40
httpContentType property
description, 1-40
httpKeepAlive property
description, 1-40
httpPassword property
description, 1-40
httpUsername property
description, 1-40

I

IBM WebSphere MQ messages
configuring with Oracle BAM, 6-2
idempotent activities
description, 10-2
incoming data
collecting and parsing in Oracle BAM, 6-6
initializeVariables property
description, 1-37
in-memory table
storing scheduled jobs, 1-24
inMemoryOptimization property
description, 1-38
setting, 1-9
installation
adapters installation failure, 3-2
changing the Oracle ESB creation schema to use a custom tablespace, 2-28
of Oracle ESB as a standalone component, 2-12
instance data
purging in Oracle ESB, 2-10
instances
instance of an invoked process not displaying in Oracle BPEL Control, 1-8
per folder in Oracle Data Integrator, 7-9
interface numbering
in Oracle Data Integrator, 7-9
interfaces
committing multiple interfaces in Oracle Data Integrator, 7-10
invocations
descriptions of, 10-3
one-way versus two-way, 10-2
invoke messages
sequence of events in the delivery of, 1-18
invoke_message table
querying for invoke messages and activities, 1-18
InvokerBean MDB
description, 1-4
IP_MESSAGE_TYPE header
fields of, 5-6
in Oracle B2B, 5-6

J

Java
using embedded Java with BPEL, 1-36
JCA connection cache
tuning, 3-30
JCA headers
propagating the file name from input to output file adapter in Oracle ESB, 2-12
JDBC drivers
supported types for Oracle Data Integrator, 7-5
supported versions for Oracle Data

Integrator, 7-5
JGroups
configuring for clustering, 3-28
JMS adapter
configuring for global transaction participation, 3-22
creating a JMS dequeue operation in a global transaction, 3-22
Oracle ESB, 2-18
using with the WebLogic JMS provider, 3-25
XA configuration, 3-22
JMS providers
use with Oracle ESB, 2-2
JMS-to-database scenario
activity map of design, 9-5
architecture of configurations, 9-6
creating the BPEL cluster, 9-20
design description, 9-3
functional requirements, 9-2
implementations of EJBs and the BPEL process, 9-5
optimizing Oracle BPEL Server and OC4J, 9-18
optimizing performance, 9-1
optimizing the BPEL process, 9-16
optimizing the enqueue application, 9-8
optimizing the inbound JMS adapter, 9-13
optimizing the inbound queue, 9-9
optimizing the JMS providers, 9-11
optimizing the outbound layer, 9-19
performance requirements, 9-2
performance tuning, 9-22
performance values by configurations, 9-23
process map of design, 9-3
testing maximum inbound queue throughput, 9-21
testing maximum Oracle BPEL Process Manager and adapter throughput with a preloaded queue, 9-21
throughput by BPEL instances, 9-22
throughput by orders, 9-22
jms.xml file
editing for the OC4J AS JMS provider, 2-19
JTA transactions
including the BPEL subprocess in, 1-6
main process propagating a JTA transaction context to a subprocess, 10-24
transaction boundaries in a flow, 1-4

K

keepGlobalVariables property
description, 1-38
knowledge modules
Oracle Data Integrator, 7-8
selecting in Oracle Data Integrator, 7-8

L

LDAP
authentication and authorization for Oracle BAM, 6-7
LDAP providers
integrating with external providers, 8-1
literal assign, 1-29
loads
throttling, 10-26
loadSchema property
description, 1-38
location property
description, 1-40
lock starvation
simulating, 1-22
lock timeout errors
resolving, 1-23
logging
enabling debug logging, 3-37
in Oracle ESB, 2-27
logical delete polling strategy
in database adapter, 3-11
logs purge
in Oracle Data Integrator, 7-10
logTopLinkAll property
setting, 3-12
looping process definition
in Oracle Data Integrator, 7-11

M

manual recoveries
performing, 10-21
manual recovery page
accessing instance from, 3-8
displaying an invoke message and activity, 1-18
markers
Oracle Data Integrator, 7-7
MaxRaiseSize property
setting a large value, 3-11
MDBs
different types, 1-4
memory
causes of out-of-memory errors, 10-23
impact of disabling the completionPersistPolicy property, 10-24
message ordering
support in Oracle ESB, 2-8
messages
disabling tracking messages in Oracle ESB, 2-10
guaranteeing message order when using the file adapter, 3-19
inbound message flow multiplexing, 3-29
not displaying in Oracle BPEL Control, 10-22
remaining permanently on the recovery page of Oracle BPEL Control, 10-22
throttling message flow, 3-27
messaging headers
support for setting in Oracle ESB, 2-8
metadata project
Oracle Data Integrator, 7-8
methodology
handling large XML documents with BPEL, 1-34

limitations on implementing a daemon using Oracle BPEL Process Manager, 1-30
modeling a two stage real-time response process, 1-35
using embedded Java with BPEL, 1-36
migration
in Oracle Data Integrator, 7-11
Oracle BAM, 6-7

N

namespace prefixes
in Oracle ESB, 2-15
.NET clients
consuming Oracle ESB services from, 2-3
noAlterWSDL property
description, 1-38
node selection failure faults
receiving, 1-28
nonblocking invoke
description, 1-16
nonBlockingInvoke property
description, 1-40
nonidempotent activities
dehydration occurrences, 10-4
NormalizedMessage header
specifying the participate property in, 1-5
notifications
in workflows, 4-9

O

OC4J
configuring version 10.1.3 with Oracle BAM, 6-2
OC4J JMS
Oracle ESB, 2-18
ODI
See Oracle Data Integrator
ODI sequence
versus database sequence in Oracle Data Integrator, 7-7
ODI_TEMP
user name for connecting to a database, 7-6
onAlarm branch
not expiring as expected, 1-26
once-and-only-once processing
in Oracle ESB, 2-14
one-way invocations
description, 10-2
operations
in Oracle Data Integrator, 7-11
operators
in Oracle Data Integrator, 7-10
optimizeVariableCopy property
description, 1-38
Oracle B2B
adapter service, 5-7
associating an inbound document to a customer, 5-6
description, 5-3
diagnosing errors, 5-8
error check list, 5-10
identifying a trading partner, 5-8
inbound direction errors, 5-9
interoperability constructs, 5-6
IP_MESSAGE_TYPE header, 5-6
outbound direction errors, 5-9
outbound purchase order use case, 5-4
place in Oracle SOA Suite, 5-3
required vertical data, 5-7
typical errors, 5-9, 5-11
typical topology, 5-4
Oracle BAM
beautifying reports with color schemas, 6-5
building a simple report, 6-5
calling external URLs, 6-6
calling external Web services, 6-6
collecting and parsing incoming data, 6-6
configuring a plan to run continuously, 6-7
configuring BAM to read data from external data sources, 6-4
configuring Oracle Enterprise Manager with, 6-4
configuring with IBM WebSphere MQ messages, 6-2
configuring with OC4J 10.1.3, 6-2
configuring with the AQ adapter, 6-2
creating a custom HTML body in a report for custom display, 6-5
deployment recommendations, 6-7
designing a report showing the ratio of two aggregated values, 6-5
designing BPEL JMS sensors to publish events in BAM, 6-3
designing BPEL sensors for BAM sensor actions, 6-3
designing BPEL sensors to send events to BAM, 6-3
designing predictive alerts, 6-6
LDAP authentication and authorization, 6-7
migration, 6-7
reading external tables into Oracle BAM using Enterprise Link plans, 6-4
repository maintenance, 6-8
support with Oracle ESB, 2-7
using without Enterprise Link, 6-8
view types, 6-5
Oracle BPEL Process Manager
handling large XML documents, 1-34
limitations on implementing a daemon, 1-30
semantics of calls to and from Oracle ESB, 1-10
Oracle Data Integrator
agent connectivity to the repository, 7-4
agent overload, 7-4
agent parameters, 7-3
agent sizing, 7-4
agent startup, 7-4
committing multiple interfaces, 7-10
contexts, 7-10
data server definitions, 7-6
database sequence versus ODI sequence, 7-7

default schemas, 7-6
documenting models and interfaces, 7-9
hardcoded table names, 7-8
ID of the work repositories, 7-2
instances per folder, 7-9
interface numbering, 7-9
JDBC driver types, 7-5
JDBC driver versions, 7-5
knowledge module modification, 7-8
knowledge module selection, 7-8
logs purge, 7-10
looping process definition, 7-11
markers, 7-7
metadata project, 7-8
migrating, 7-11
name of the work repositories, 7-2
new technology creation, 7-6
number of database schemas for the repositories, 7-1
number of work repositories, 7-2
operations, 7-11
operators, 7-10
remote connectivity to repositories, 7-3
repositories, 7-1
repositories in different environments, 7-2
repository types, 7-2
supported databases for repositories, 7-3
testing agent connectivity to the data servers, 7-7
testing database connectivity, 7-6
topology, 7-5
transformations, 7-9
user name for connecting to a database, 7-6
variables in a package, 7-10
XML or JMS/XML synchronization, 7-9

Oracle Enterprise Manager
configuring with Oracle BAM, 6-4

Oracle ESB
automatic retry configuration, 2-9
calling an ESB routing service from ADF, 2-3
changing the Oracle ESB creation schema to use a custom tablespace, 2-28
consuming from .NET clients, 2-3
creating a bridge to another ESB across a firewall, 2-3
creating the DefaultSystem, 2-27
creating the routing service to use pure SQL with dynamic routing in, 2-43
data source setup for using pure SQL with dynamic routing, 2-34
database adapter setup for using pure SQL with dynamic routing, 2-37
database setup for using pure SQL with dynamic routing, 2-33
databases supported as a storage backend, 2-2
databases supported on a Windows client workstation, 2-2
DefaultSystem group in missing, 2-25
deploying using pure SQL with dynamic routing, 2-46
disabling tracking messages, 2-10
displaying BPEL processes in Oracle ESB Control, 2-12
enabling end-to-end tracking between ESB and BPEL, 2-24
error hospital, 2-15
filtering support, 2-8
fixing a registration error, 2-27
fixing IOException errors, 2-28
guaranteeing once-and-only-once processing, 2-14
handling inbound HTTP posts, 2-4
handling multiple BPEL versions when using an ESB SOAP service to call BPEL, 2-13
high availability support, 2-8
inability to register projects or connect to Oracle ESB Control, 2-24
inconsistent data in Oracle ESB Control, 2-25
installing as a standalone component, 2-12
JMS adapter, 2-18
limitation on versioning support, 2-21
limitations on inbound e-mail adapter support, 2-12
logging facilities, 2-27
management API support, 2-7
managing the timeout duration for a synchronous request-response from a SOAP invocation service, 2-7
maximizing performance, 2-18
maximum number of sessions exceeded, 2-23
messaging ordering support, 2-8
modeling in Oracle JDeveloper to use pure SQL with dynamic routing, 2-40
no namespace prefix, 2-15
number of listeners feature, 2-9
OC4J JMS, 2-18
optimization of service communication in the same Java container, 2-7
Oracle ESB Control not displaying services or instance tracking information, 2-26
overriding a WSIF invocation of a BPEL service, 2-13
passing security credentials when invoking an external Web service, 2-9
performance speed, 2-17
pointing at alternative locations for a WSDL of a SOAP service endpoint, 2-13
propagating the file name from input to output file adapter using JCA headers, 2-12
pure SQL with dynamic routing in, 2-32
purging instance data, 2-10
registering a project from the command line with ant, 2-9
reporting problems, 2-22
resubmitting multiple error instances, 2-9
results of when two asynchronous services fail, 2-6
retiring processes, 2-21
role in JTA transaction rollbacks, 1-6
RPC style SOAP services not supported, 2-4
running in third-party containers, 2-2

scenarios under which databases are required, 2-5
semantics of calls to and from Oracle BPEL Process Manager, 1-10
sending e-mail notifications, 2-6
separator for different SOAP endpoints, 2-14
setting message headers, 2-8
sizing, 2-19
support for dynamic partner links, 2-14
support for Oracle BAM, 2-7
support for rejectedMessageHandlers property, 2-8
supported adapters, 2-3
supported operating systems, 2-2
supported XML document sizes, 2-8
transaction semantics, 2-29
transaction support, 2-4
use with third-party JMS providers, 2-2
user accounts and roles, 2-7
using custom XPath functions, 2-14
using pure SQl with dynamic routing, 2-32
using reset.bat and reset.sh scripts, 2-12
Web services SOAP/HTTP, 2-18
workarounds for support of multipart messages or binary attachments, 2-6
WSDL known issues, 2-10
WSIF support, 2-4
Oracle ESB Control
default URL, 2-24
displaying BPEL processes, 2-12
not displaying services or instance tracking information, 2-26
Oracle JDeveloper
setting up pure SQL with dynamic routing in Oracle ESB, 2-40
orion-ejb-jar.xml file
changing all values of transaction-timeout on all EJBs, 1-10
server transaction setting specified in, 1-20
setting threading values in, 1-2
throttling requests, 1-3
OutboundRetryCount property
setting, 2-8
OutboundRetryEnabled property
setting, 2-8
OutboundRetryInterval property
setting, 2-8

P

partnerLinkBinding properties
deployment descriptor, 1-39
payloads
processing large payloads, 10-15
performance
concepts and terminology, 10-1
disabling dehydration to improve asynchronous process performance, 10-24
incurring overhead when calling a BPEL subprocess, 1-27
maximizing Oracle ESB performance, 2-18
memory impact of disabling the completetionPersistPolicy property, 10-24
optimizing in a JMS-to-database scenario, 9-1
relationship of some performance settings, 10-7
speed of Oracle ESB, 2-17
trading with reliability, 10-21
tuning Quartz performance, 1-25
plans
configuring to run continuously in Oracle BAM, 6-7
predictive alerts
designing in Oracle BAM, 6-6
preferredPort property
description, 1-40
primary key
database adapter constraints, 3-10
processes
durable versus transient, 10-2
retiring in Oracle ESB, 2-21
pure SQL
using with dynamic routing, 2-32
with dynamic routing in Oracle ESB, 2-32
pw property
description, 1-39

Q

Quartz
actions used for, 1-26
tuning performance, 1-25
using to schedule expiration events, 1-24

R

RAC
handling a RAC node failover with Oracle BPEL Server, 1-21
ReceiverThreads property
description, 1-1
optimal value based on Eden garbage collection, 1-2
relationship to dspMaxThreads, 1-1
tuning to maximize throughput and minimize JVM thread context switching, 1-4
recovery
displaying an invoke message and activity in the manual recovery page, 1-18
implementing an automated recovery mechanism, 1-20
messages not displaying in Oracle BPEL Control, 10-22
messages remaining permanently on the recovery page of Oracle BPEL Control, 10-22
performing manual recoveries, 10-21
synchronous messages displaying in the recovery page, 10-23
rejectedMessageHandlers property
configuring, 3-32
support in Oracle ESB, 2-8

relaxTypeChecking property
  description, 1-38
relaxXPathQName property
  description, 1-38
reliability
  trading with performance, 10-21
reports
  building in Oracle BAM, 6-5
  creating a custom HTML body for custom display in Oracle BAM, 6-5
  designing to show the ratio of two aggregated values in Oracle BAM, 6-5
repositories
  ID of Oracle Data Integrator work repository, 7-2
  in different Oracle Data Integrator environments, 7-2
  maintaining in Oracle BAM, 6-8
  name of Oracle Data Integrator work repository, 7-2
  number of Oracle Data Integrator work repositories, 7-2
  Oracle Data Integrator, 7-1
  remotely connecting to with Oracle Data Integrator, 7-3
  supported databases for Oracle Data Integrator, 7-3
  types for Oracle Data Integrator, 7-2
requests
  throttling, 1-3
reset.bat script
  using, 2-12
reset.sh script
  using, 2-12
resource bundles
  in workflows, 4-9
response processes
  modeling a two stage real-time response process, 1-35
RetryCount property
  setting, 2-7, 2-10
RetryInterval property
  setting, 2-7, 2-10
retryInterval property
  description, 1-40
retryMaxCount property
  description, 1-40
role property
  description, 1-39
roles
  in Oracle ESB, 2-7
rollbacks
  forcing a rollback in a BPEL flow, 1-5
  forcing with bpelx:rollback, 1-6
  in XA configurations, 3-4, 3-23
  invoking a partner link that marks the JTA transaction for rollback only, 1-6
  reasons for transaction rollbacks, 1-8
routing services
  in workflow, 4-7
  setting up pure SQL with dynamic routing in Oracle ESB, 2-43
RPC style SOAP services
  not supported in Oracle ESB, 2-4

S

scheduling
  jobs stored in in-memory table, 1-24
  using Quartz to schedule expiration events, 1-24
schemas
  default schema for Oracle Data Integrator, 7-6
security
  integrating with external LDAP providers, 8-1
security credentials
  passing when invoking an internal Web service in Oracle ESB, 2-9
sendXSIType property
  description, 1-40
sensorActionLocation property
  description, 1-38
sensorLocation property
  description, 1-38
sensors
  configuring BPEL sensors to send events to Oracle BAM, 6-3
  designing BPEL JMS sensors to publish events in BAM, 6-3
  designing BPEL sensors for BAM sensor actions, 6-3
serviceProperties property
  description, 1-40
sizing
  Oracle ESB, 2-19
skipDeepCopy property
  description, 1-27
SLACompletionTime property
  description, 1-39
synchronization
  in Oracle Data Integrator, 7-9
synchronous processes
  dehydration, 10-24
  supported in Oracle BPEL Process Manager, 10-19
synchronous requests
  not seeing them time out, 1-14
synchronous transient processes, 10-6
  receiving a transaction timeout if invoking, 1-9
syncMaxWaitTime property
  description, 1-14

T

table names
  hardcoded in Oracle Data Integrator, 7-8
testIntroduction property
  description, 1-38
third-party containers
  certified with Oracle ESB, 2-2
threading
  configuration for the file adapter, 3-19

controlling the number for executing invocation messages, 1-3
created to dehydrate a transaction instance, 10-5
for a synchronous transient process, 10-6
for an asynchronous durable process, 10-5
MDB responsibilities in Oracle BPEL Server, 1-4
relationship between ReceiverThreads and dspMaxThreads, 1-1
throttling requests, 1-3
WorkerBean MDB, 1-2

threading and transactions for, 10-5, 10-6
threading and transactions for a synchronous transient process, 10-6
threading and transactions for an asynchronous durable process, 10-5
throughput
typical transaction rate for Oracle BPEL Server, 10-19
timeout errors
resolving a process lock timeout error, 1-22
resolving lock timeout errors, 1-23
timeout property
description, 1-40
topology
for Oracle Data Integrator, 7-5
trading partners
order of identification, 5-8
transaction boundaries
in a flow, 1-4
transaction property
description, 1-39
transaction rollbacks
reasons for rollbacks, 1-8
transaction semantics
in Oracle ESB, 2-29
transaction timeouts
modifying timeout settings for Oracle BPEL Server, 1-9
reasons for a transaction rollback, 1-8
receiving if invoking a synchronous transient process, 1-9
transaction=participate property
applying as a partner link property for a synchronous BPEL subprocess invocation, 1-6
applying as a process property, 1-7
global rollbacks, 3-4
impact on BPEL transactionality, 1-6
passed in the NormalizedMessage used to invoke the process, 1-11
setting in XA configurations, 3-5, 3-23
specifying in the NormalizedMessage header, 1-5
transactions
committed when dehydration occurs, 10-5
for a synchronous transient process, 10-6
for an asynchronous durable process, 10-5
forcing a rollback in a BPEL flow, 1-5
impact of transaction=participate property on BPEL transactionality, 1-6
instance of an invoked process not displaying in Oracle BPEL Control, 1-8
main process propagating a JTA transaction context to a subprocess, 10-24
modifying transaction timeout settings for Oracle BPEL Server, 1-9
receiving a transaction timeout if invoking a synchronous transient process, 1-9
semantics with Oracle ESB and Oracle BPEL Process Manager calls, 1-10
transaction boundaries in a flow, 1-4
types of support in Oracle ESB, 2-4
transaction-timeout attribute
changing all values on all EJBs, 1-10
transformations
in Oracle Data Integrator, 7-9
transient processes, 10-6
dehydration occurrences, 10-4
description, 10-2
troubleshooting
creating the DefaultSystem, 2-27
DefaultSystem group in missing, 2-25
enabling end-to-end tracking between ESB and BPEL, 2-24
exceptions to look for when using the database adapter, 3-13
finding the root exception when using the database adapter, 3-12
fixing IOException errors, 2-28
fixing the registration error in Oracle ESB, 2-27
inability to register projects or connect to Oracle ESB Control, 2-24
inconsistent data in Oracle ESB Control, 2-25
journaling disabled when using the database adapter, 3-13
maximum number of sessions exceeded, 2-23
not displaying services or instance tracking information in Oracle ESB Control, 2-26
reporting problems in Oracle ESB, 2-22
Web service tools to use, 2-23
tuning
database adapter documentation, 3-12
two-way invocations
description, 10-2

U

URLs
calling external URLs through Oracle BAM, 6-6
user accounts
in Oracle ESB, 2-7
user property
description, 1-39
UseServerTimeout property
setting to improve OC4J AS JMS performance, 2-18

V

validateXML property
description, 1-41

variables
  in a package in Oracle Data Integrator, 7-10
vertical data
  required in Oracle B2B, 5-7
view types
  Oracle BAM, 6-5

W

wait activity
  not expiring as expected, 1-26
Web services
  calling external services through Oracle BAM, 6-6
  troubleshooting tools, 2-23
Web services SOAP/HTTP
  Oracle ESB, 2-18
Windows client workstation
  databases supported with Oracle ESB, 2-2
Windows network folders
  accessing when using the file adapter, 3-16
work_item table
  querying for invoke messages and activities, 1-18
WorkerBean MDB
  decision to request an MDB, 1-2
  description, 1-2, 1-3, 1-4
  relationship with other performance settings, 10-7
workflow services
  archiving completed instances periodically, 11-5
  attachments, 4-10
  callbacks, 4-8
  chaining complex routing patterns, 4-8
  choosing the right workflow service client, 11-1
  clustering nodes, 11-6
  completing workflow faster, 11-6
  configuring the API reference, 4-1
  escalations, 4-9
  fetching the information required for a qualifying task, 11-3
  improving server performance, 11-4
  Java interface for exposing methods, 4-2
  minimizing client response time, 11-1
  minimizing the notification overhead, 11-6
  narrowing qualifying tasks using precise filters, 11-2
  notifications, 4-9
  printing payload contents with the API reference, 4-6
  reducing the number of return query columns, 11-3
  resource bundles, 4-9
  retrieving subsets of qualifying tasks, 11-2
  routing capabilities, 4-7
  specifying escalation rules, 11-7
  specifying user and group rules for automated assignment, 11-7
  suppressing logging and debugging messages, 11-5
  tuning the database, 11-4
  tuning the identity provider, 11-4
  using Java callbacks instead of BPEL callbacks, 11-5
  using task views to prioritize work, 11-7
  using the API reference, 4-1
  using the main() method of the API reference, 4-3
  using workflow reports to monitor progress, 11-6
wsdlLocation property
  description, 1-41
wsdlRuntimeLocation property
  description, 1-41
WSDLs
  known issues with Oracle ESB, 2-10
WSIFs
  overriding a WSIF invocation of a BPEL service in Oracle ESB, 2-13
  supported in Oracle ESB, 2-4
wsseHeaders property
  description, 1-41
wssePassword property
  description, 1-41
wsseUsername property
  description, 1-41

X

XA configuration
  both invokes in the same global transaction, 3-4, 3-23
  database adapter, 3-3
  failure causes a rollback, 3-4, 3-23
  guaranteeing both invokes use the same SQL connection, 3-5
  JMS adapter, 3-22
  making a database adapter receive and invoke part of the same XA transaction, 3-5
  receiving a CCI local transaction commit error, 3-24
xADataSourceName
  setting, 3-3
XML documents
  handling large documents, 1-34
  sizes supported by Oracle ESB, 2-8
XPath functions
  using custom functions in Oracle ESB, 2-14
xpathValidation property
  description, 1-39
XSLT Mapper
  using custom functions in Oracle ESB, 2-14