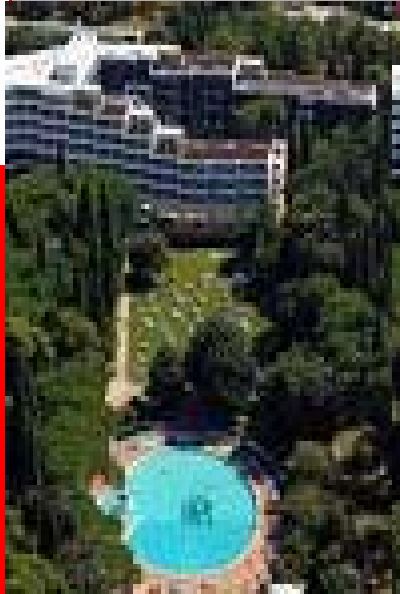


ORACLE®



ORACLE®



Fault Handling in SOA 10.1.3.3/4

Duško Vukmanović
Senior Sales Consultant

Oracle BPEL 10.1.3.3: Fault Handling



Business vs. Technical Faults

- Business fault
 - Defined in service WSDL – **expected to happen**
 - Service designer defines message structure for the fault
 - Part of business state – represents failures in business state, input data inconsistencies, exceptional states

- Technical fault
 - Predefined BPEL faults (remoteFault, bindingFault, etc.) - **unexpected**
 - Infrastructure faults
 - Service down, network outage, connection timeouts
 - Data format errors
 - Inappropriate data structure, corrupted message

BPEL standard faults

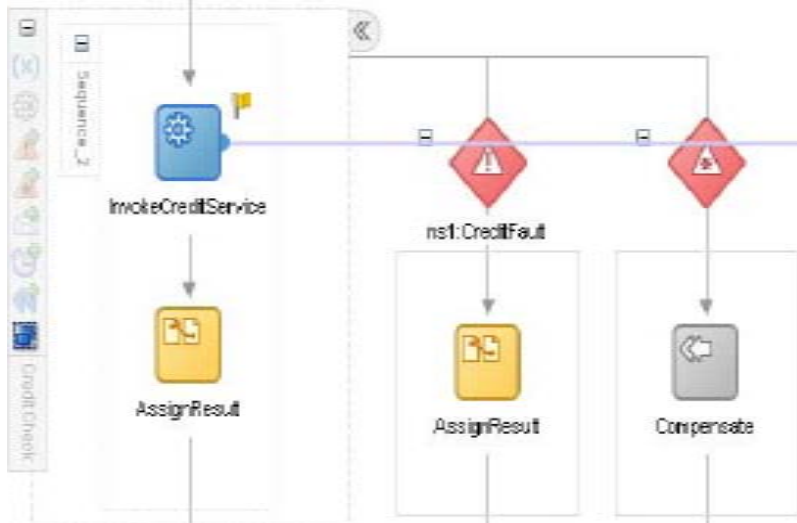
- **BPEL spec defines 11 standard faults in the namespace of "http://schemas.xmlsoap.org/ws/2003/03/business-process/". These 11 standard faults are typeless, meaning they don't have associated messageTypes**
 - selectionFailure
 - conflictingReceive
 - conflictingRequest
 - mismatchedAssignmentFailure
 - joinFailure
 - forcedTermination
 - correlationViolation
 - uninitializedVariable
 - repeatedCompensation
 - invalidReply

Runtime Faults

- **Oracle BPEL PM server introduces a few runtime faults. They are in the namespace of "http://schemas.oracle.com/bpel/extension". These two faults are associated with a collaxa defined messageType "RuntimeFaultMessage".**
 - **BindingFault** -BindingFault is thrown from inside a activity whenever the preparation of the invocation fails. For example, the WSDL of the process fails to be loaded. This kind of fault usually has to be fixed by human intervention.
 - **RemoteFault** - RemoteFault is thrown from inside a activity too. It is thrown because the invocation fails. For example, a SOAP fault is returned by the remote service. RemoteFault can be configured to be retried.

Handling of Business Faults

- Business faults
 - Handled by business process or service as part of business logic
 - Actions depend on business logic (meaning of the fault), case to case different
 - Rich BPEL support (catch blocks, compensation handlers)



Handling of Technical Faults

- Typical actions from the same set
 - retry, abort, manual intervention
- **Handling in process** => repeated code in all processes, hard to maintain
- **Error Hospital concept** => specialized handling process, requires common code in all processes too
- **Engine level fault handling** => decoupled from processes, transparent, centrally managed

Oracle BPEL Process Manager

Policy Driven Fault Handling (10.1.3.3)

- Designed for technical faults (but can handle business faults as well)
- Engine level
 - No impact on BPEL process design
 - No impact on process in runtime (fault is isolated from process)
- XML-based fault policies
 - Conditions for faults (fault name, XPath on fault content)
 - Set of actions (retry, human intervention, reply scope, rethrow fault, abort, custom Java action)
 - Centrally managed as process ‘aspects’
- Manual resubmission in BPEL Console

XML for Fault Handling Policy

```

<?xml version="1.0" encoding="UTF-8"?>
<faultPolicy version="2.0.1" id="CRM_ServiceFaults"
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">
  <Conditions>
    <faultName xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
name="bpelx:remoteFault">
      <condition>
        <test>$fault.code/code="WSDLReadingError"</test>
        <action ref="ora-human-intervention"/>
      </condition>
      <condition>
        <action ref="ora-retry"/>
      </condition>
    </faultName>
  </Conditions>

```

Policy ID

Fault Name

XPath Expression

Action Reference

XML for Fault Handling Policy

<Actions>

<Action id="ora-retry">

Action ID, referred from Condition

```

<retry>
  <retryCount>3</retryCount>
  <retryInterval>2</retryInterval>
  <exponentialBackoff/>
  <retryFailureAction ref="ora-java"/>
  <retrySuccessAction ref="ora-java"/>
</retry>

```

</Action>

<Action id="ora-rethrow-fault">

Action specification

```

<rethrowFault/>

```

</Action>

<Properties>

```

  <propertySet name="propSet1">
    <property name="server">production</property>
  </propertySet>

```

Properties

```

</Properties>

```

</Actions>

</faultPolicy>

Recovery Actions

- Retry
 - Retries failed operation # times with specified wait time, exponential back off, retry failure action, multiple WSDL locations

- Human Intervention
 - Console based recovery, possibility of changing values of BPEL variables

- Abort process

```

<Action id="ora-retry">
  <Retry>
    <retryCount>3</retryCount>
    <retryInterval>2</retryInterval>
    <exponentialBackoff/>
    <retryFailureAction ref="ora-java"/>
    <retrySuccessAction ref="ora-java"/>
  </Retry>
</Action>

```

```

<Action id="ora-human-intervention">
  <humanIntervention/>
</Action>

```

```

<Action id="ora-terminate">
  <abort/>
</Action>

```

Recovery Actions

- Replay scope
 - Throws replay fault, which triggers replay of enclosing scope
- Rethrow fault
 - Throws fault into the process instance, which can handle it
- Java action – custom Java code called
 - Code can perform additional tasks and select recovery action

```
<Action id="ora-replay-scope">
  <replayScope/>
</Action>
```

```
<Action id="ora-rethrow-fault">
  <rethrowFault/>
</Action>
```

```
<Action id="ora-java">
  <javaAction className="mypackage.myClass"
    defaultAction="ora-terminate"
    propertySet="propSet1" >
    <returnValue value="R_TRM"
      ref="ora-terminate"/>
    <returnValue value="R_THRW"
      ref="ora-rethrow-fault"/>
  </javaAction>
</Action>
```

Java Action

- Implements `IFaultRecoveryJavaClass` interface

```
public interface IFaultRecoveryJavaClass {
    public void handleRetrySuccess(IFaultRecoveryContext ctx );
    public String handleBPELFault(IFaultRecoveryContext ctx );
}
```

- `handleBPELFault` called on fault, return value mapped to an action
- `handleRetrySuccess` called when a retry is successful and the Retry action chains this Java action as retry success action
- Typically handles ‘side tasks’: notifications, fault logging and extended decisions about recovery action
- Executed in EJB context – within BPEL transaction

Policy Bindings

- Domain level
 - Stored in \$ORACLE_HOME/bpel/domains/<domain>/config/fault-bindings.xml
 - Default policy for all processes in a domain:

```
<faultPolicyBindings version="2.0.1">
  <process faultPolicy="ConnectionFaults"/>
</faultPolicyBindings>
```

- Binding for partner link
 - Defined by partner link name or port type, valid across all processes in a domain, which have such partner link

```
<partnerLink faultPolicy="CRM_ServiceFaults">
  <name>creditRatingService</name>
  <portType xmlns:credit="http://services.otn.com">
    credit:CreditRatingService
  </portType>
</partnerLink>
```

Policy Bindings

- Process level
 - Stored in `bpel.xml` descriptor of the process
 - Default policy for process
 - Policy for partner link (based on partner link name or port type)
- Resolution of policy binding
 - Partner link binding in `bpel.xml`
 - Port type binding in `bpel.xml`
 - Process level binding in `bpel.xml`
 - Partner link level binding specified for the domain.
 - Port type binding specified for the domain.
 - Process level binding at domain level

Human Intervention

Console Recovery

- ‘Frozen’ BPEL instance – corresponding activity in pending state
- Activities tab in BPEL Console:
 - Search for pending activities
 - Select more activities and perform recovery (retry, continue, rethrow, abort, replay)
- Activity detail (after clicking on a pending activity)
 - Examine and change BPEL instance variables
 - Examine the fault information
 - Perform recovery action (retry, continue, rethrow, abort, replay)
 - Start new instance with the same input message

Oracle ESB 10.1.3.3: Fault Handling



ESB Error Types

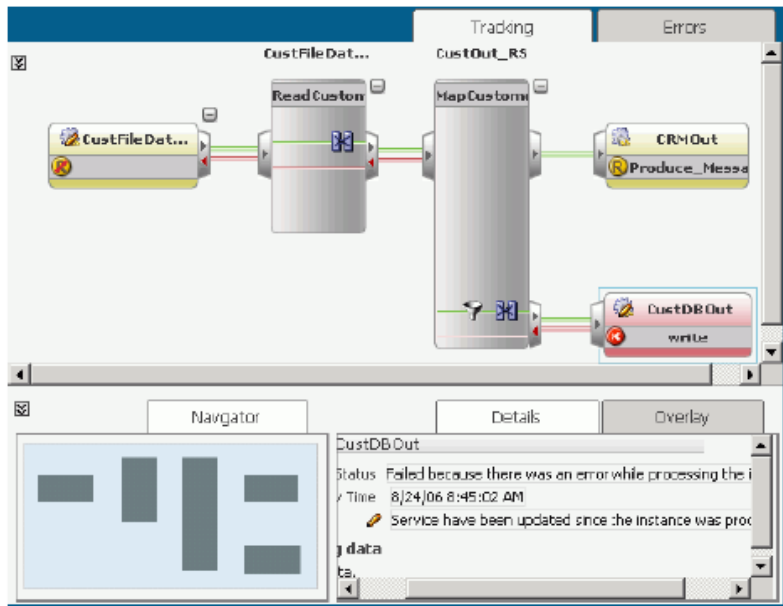
- Business Faults
 - Specified in service WSDL, semantically equal to BPEL business faults
 - Can be routed like response messages
- Retriable errors
 - Connection errors, service downtimes
 - Usually resolved in a relatively short time period
 - Non-retriable errorsA disabled or deleted service or system, out of memory condition, etc.

Fault Handling

Synchronous vs. Asynchronous Execution

- Synchronous execution
 - Transaction is rolled back
 - Error is returned to the caller
 - Caller can retry the operation or handle the error
 - Inbound adapters can retry automatically (configured by endpoint properties)
 - Rejection handlers can be used
- Asynchronous execution
 - Error is sent to ESB Error Hospital
 - Retriable errors can be resubmitted manually from ESB console or programmatically via ESB Client API

ESB Error Hospital



Tracking
Errors

Time	Service:Operation	Message
7/27/06 10:01:44 PM	CustomerData.CustDBOut.write	

Resubmission

To rectify the errors, edit the instance payload, and then resubmit it. The following rules are executed on resubmission.

CustomerData.CustOut_RS.MapCustomerData

{inp1:Custom eSB://ESB_Project CustDBOut::write}

Resubmission Payload Resubmit

```
<inp1:Customer xmlns:inp1="http://xmlns.oracle.com/EsB/CustomProvision">
  <CustomerId>101</CustomerId>
  <Name>Acme International</Name>
  <Description>Accounting Outsourcing Partner</Description>
  <Profile>
    <Status>Active</Status>
    <CreditRating>5</CreditRating>
    <CreditTerms>30n4</CreditTerms>
    <PreferredCurrency>USD</PreferredCurrency>
    <ActivityRating>2</ActivityRating>
    <AccountRep>
      <Title>VP Finance</Title>
    </AccountRep>
  </Profile>
</inp1:Customer>
```

Microsoft Internet Explorer

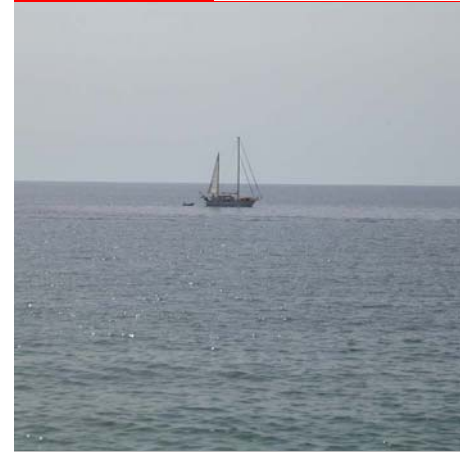
Resubmitted Successfully

OK

Time	Instance ID	Status
2:30:14 AM	9A8223C0379011D8BF206B344644428D	
2:30:10 AM	90682AA0379011D8BF206B344644428D	
2:33:10 PM	9376D030374511D8BF206B344644428D	
2:33:08 PM	928F0840374511D8BF206B344644428D	
2:39:34 PM	16631130373E11D8BF206B344644428D	
2:39:34 PM	166C2DD0373E11D8BF206B344644428D	
2:39:33 PM	15FB88C0373E11D8BF206B344644428D	
2:39:33 PM	15C73840373E11D8BF206B344644428D	

- Error (Resubmittable)
- Error
- Faulted
- Error & Faulted
- Error (Rejection) & Faulted
- No Faults or Errors
- Error (Resubmittable) & Faulted

Oracle Consulting: Fault Handling Extensions



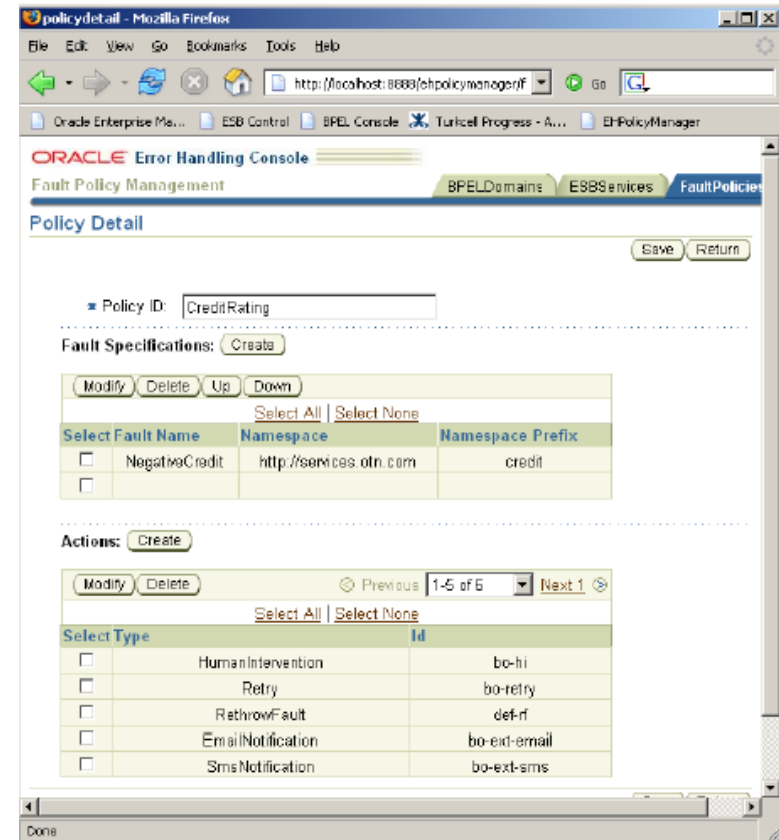
Main Requirements

- Unified technical errors handling in BPEL and ESB
 - Based on fault handling policies in BPEL
 - Automatic retry in ESB Error Hospital
 - Ability to resubmit large amounts of failed instances
 - GUI for maintaining fault handling policies
- Notifications
 - E-mail, SMS and Web Service channels, aggregated by thresholds
- Fault logging
 - Logging of all faults from BPEL and ESB into database
 - GUI for searching the error logs
- Metadata propagation for BPEL instances

BPEL and ESB Extensions

Fault Handling Policies Management GUI

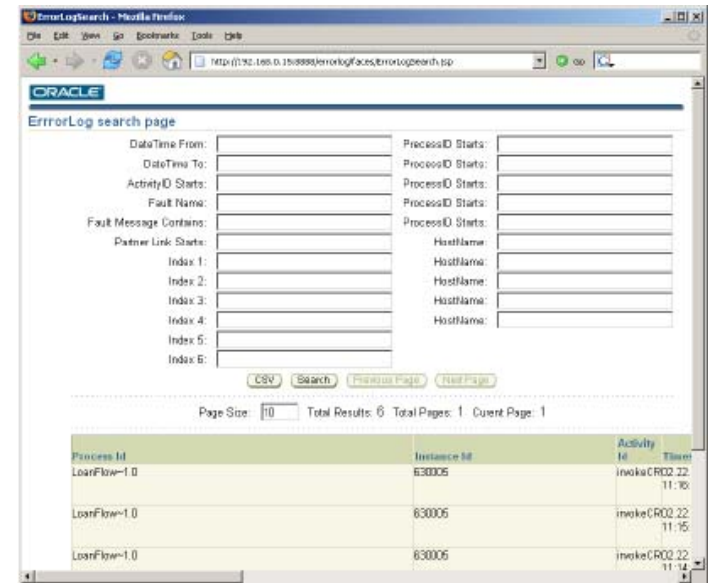
- WEB based GUI for maintaining fault-handling policies with all extended actions (notifications and WS call).
- Provides editing capabilities for policies (create, modify, delete)
- Maintains associations of policies to ESB services and BPEL Processes
- Uses Oracle ADF with EJB 3.0 and JPA persistence, exports XML policies for BPEL and ESB



BPEL and ESB Extensions

Notifications and Logging

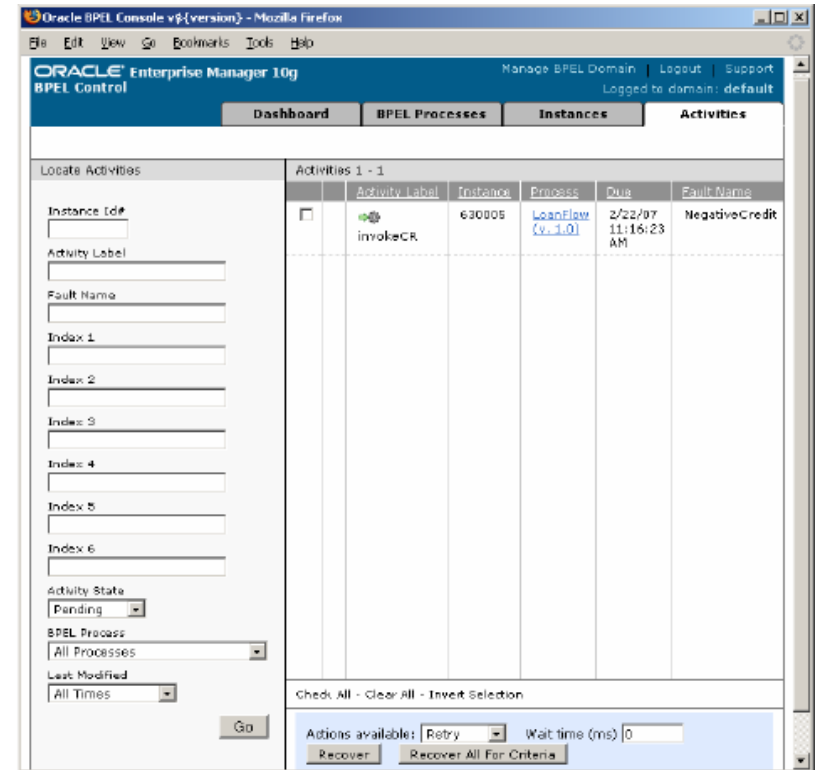
- Configurable e-mail, SMS and WS call notifications Threshold for number of errors and recipients can be specified in fault-handling policy
- WS call can be configured in the policy – another extension point
- BPEL notification service is used to fire notifications
- Implemented as Java Action in BPEL policy
- Database error logging
 - All faults are logged into a database table
 - Simple GUI for searching



BPEL Extensions

Enhanced BPEL Manual Recovery

- Extensions of BPEL Console (JSP pages and a new EJB module)
- Extended search criteria for BPEL instances and activities queries
- Enables recovering all BPEL instances matching specified search criteria
- The wait time can protect the system from being overloaded when big number of instances is being resolved.



The screenshot shows the Oracle BPEL Console v8.1.0.0 interface in Mozilla Firefox. The main window displays the 'BPEL Control' section with tabs for Dashboard, BPEL Processes, Instances, and Activities. The 'Activities' tab is active, showing a table of activities. On the left, there are search filters for Instance Id#, Activity Label, Fault Name, and six Index fields. The 'Activity State' is set to 'Pending', 'BPEL Process' is 'All Processes', and 'Last Modified' is 'All Times'. A 'Go' button is at the bottom of the filters. The table shows one activity: 'invokeCR' with Instance '630005', Process 'LoanFlow (v. 1.0)', Due date '2/22/07 11:16:23 AM', and Fault Name 'NegativeCredit'. At the bottom, there are buttons for 'Recover' and 'Recover All For Criteria', along with a 'Wait time (ms)' field set to 0.

Activity Label	Instance	Process	Due	Fault Name
invokeCR	630005	LoanFlow (v. 1.0)	2/22/07 11:16:23 AM	NegativeCredit

BPEL Extensions

Metadata Propagation

- Implemented as a message interceptor in BPEL

```

<Part name="payload:(http://xmlns.oracle.com/LoanProcess)LoanProcessProcessRequest">↓
  <Field>↓
    <XPath>{/ns1:LoanProcessProcessRequest/ns1:SSID},{ns1=http://xmlns.oracle.com/Loa
    <HeaderField>instanceIndex1</HeaderField>↓
  </Field>↓
  <Field>↓
    <XPath>{/ns1:LoanProcessProcessRequest/ns1:userID},{ns1=http://xmlns.oracle.com/
    <HeaderField>instanceIndex2</HeaderField>↓
  </Field>↓
</Part>↓
  
```

ORACLE Enterprise Manager 11g
BPEL Control

Dashboard | BPEL Processes | Instances

Title: Instance #420008 of LoanProcess
 Reference Id: 420008
 BPEL Process: [LoanProcess \(v. 1.0\)](#)
 Last Modified: 3/13/08 4:17:21
 State: open.running
 Priority: 0

Additional Information
 Conversation Id: d3464ecc8cb88134;-4d7931e3;118a87cb109;-7f77
 Creator: null
 Metadata: null
 Status: CreditCheck

Index
 Index #1: 0235435
 Index #2: marcel

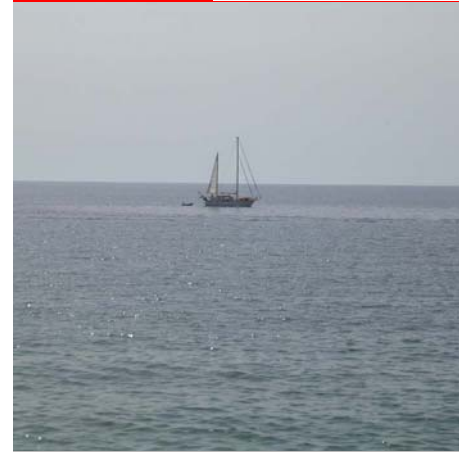
```

<ns1:LoanProcessProcessRequest>
  <ns1:userID>marcel</ns1:userID>
  <ns1:SSID>0235435</ns1:SSID>
  <ns1:loanAmount>100000</ns1:loanAmount>
</ns1:LoanProcessProcessRequest>
  
```

10.1.3.4 New Features – Fault Handling

- **Fault Policy Changes Do Not Require a Restart of Oracle BPEL Server**
 - http://host.port/BPELConsole/domain_name/doReloadFaultPolicy.jsp
- **Fault Management Behavior When the Number of Instance Retries is Exceeded**
 - when you configure a fault policy to recover instances with the ora-retry action and the number of specified instance retries is exceeded - The instance is marked as **open.faulted** (in-flight state) once the number of instance retries is exceeded. The instance remains active.

Fault Handling **Demonstration**



Summary

- Externalizes handling of technical failures from processes
 - Network outages, service downtimes
- Can be centrally managed by operational teams
- Manual intervention – more control over fault recovery
 - Faulted processes are not ‘lost’, can be retried
- ICCC extensions
 - Partial port for ESB (policy based automatic retry on Error Hospital)
 - Fault logging and e-mail/SMS/WS notifications
 - GUIs (error log, policy management, ESB bulk resubmit)

Fault\Compensation Code Examples: \$SOA_HOME\bpel\samples\...

- Returning a sync fault to client:
 - \utils\CreditRatingService
- Catching sync faults
 - \demos\LoanDemo, \references\Catch
- Catching sync faults and async fault patterns:
 - \tutorials\107.Exceptions
- Throwing faults internally:
 - \references\Throw
- XA-style transactions:
 - demos\BankTransferDemo\BankTransferFlow
- Compensation:
 - demos\BankTransferDemo\BankTransferFlowWithCompensation



ORACLE IS THE INFORMATION COMPANY

ORACLE®