Oracle SOA Fault Management Framework
By Kathiravan Udayakumar

SOA Fault Management Framework is a very key framework that requires SOA Architects communities’ attention. It cannot be handover to process specific development community to define the fault management rules and policies as the implication related to this would have wide and critical impacts to SOA Processes running in the SOA Domain.

SOA Faults needs to be handled across all SOA Components and SOA Platforms. Fault Management is complex as the interaction with SOA Components will define the nature of the Fault Management Technique. Oracle has provided an extensive fault management framework to handle faults in a declarative fashion. This chapter will help the readers to understand the Oracle Fault Management Techniques and provides the required guidelines to extend the Oracle framework to achieve sophisticated results from the framework to suite all interaction patterns.

SOA Fault Management components are also required to be integrated to SOA Logger components to log the error messages. Below list of topics will be discussed in detail in this article.

- Exception handling in Oracle SCA Composite Applications
- Oracle SOA Fault Management Framework
  - Oracle SOA Fault Policy Resolution Process
- Extended SOA Fault Management Framework
- Fault Interaction Patterns
- Impacts of Fault Management in SOA Design Patterns
- How to create Custom Fault Actions in Fault Management Framework?

**Exception handling in Oracle SCA Composite Applications**

Exceptional handling is a key component of any computer programming paradigm. Exceptions are raised and propagated as faults in the BPEL context. Error handling in service composite applications can be done using the following list of activities provided by BPEL:

- ...
<table>
<thead>
<tr>
<th>S. No.</th>
<th>Activity name</th>
<th>BPEL icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Catch</td>
<td>![Catch icon]</td>
<td>Catch blocks helps to catch a process specific fault type. System/Business faults and fault message types defined in <a href="#">partner links</a> WSDLs can be handled using this catch block.</td>
</tr>
<tr>
<td>2</td>
<td>CatchAll</td>
<td>![CatchAll icon]</td>
<td>CatchAll blocks added to main or specific block of code can be used to handle any of the faults that are not handled by the catch blocks.</td>
</tr>
<tr>
<td>3</td>
<td>Throw</td>
<td>![ThrowGenericErrorMessage icon]</td>
<td>Throw activity is used to throw the fault/error messages to BPEL engine to show the process/composite state/status as faulted.</td>
</tr>
<tr>
<td>4</td>
<td>Reply</td>
<td>![ReplyFault icon]</td>
<td>Reply activity is used to propagate the fault messages back to a BPEL process.</td>
</tr>
<tr>
<td>5</td>
<td>Terminate</td>
<td>![TerminateTheProcess icon]</td>
<td>This activity helps to terminate a BPEL process.</td>
</tr>
</tbody>
</table>

**System faults and custom faults**

System faults are propagated by the Oracle SCA infra to BPEL when any system level issues are encountered. The list of system faults that are available with SOA Suite 11.1.1.5 are shown in the next screenshot.

The most commonly occurring system faults are remoteFault and bindingFault. Rollback fault is the key fault message that is used in BPEL. Rollback fault message is used to rollback a transaction. Faults propagated to the caller will rollback the transaction.
Custom fault messages that are to be used in BPEL are defined through WSDL. Fault messages defined in the WSDL can be used to propagate the faults to the caller which can be handled in the calling application for error scenarios.

The following listing shows WSDL with a fault message defined in it. These faults messages can be expected by the caller when this service is invoked.

Listing

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SyncBPELProcess"
    targetNamespace="http://xmlns.oracle.com/SOAApplication/SOAProject/SyncBPELProcess"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:client="http://xmlns.oracle.com/SOAApplication/SOAProject/SyncBPELProcess"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">
  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://xmlns.oracle.com/SOAApplication/SOAProject/SyncBPELProcess"
        schemaLocation="xsd/SyncBPELProcess.xsd"/>
    </schema>
  </wsdl:types>
</wsdl:definitions>
```
The previous screenshot shows Fault Chooser with custom fault defined in WSDL. This can be used in choosing the faults to be handled by the catch block.
Oracle SOA Fault Management Framework

Fault management is a declarative environment through which the faults can be managed. Both system and business faults can be managed through fault management framework. Fault management framework provides features to manage both BPEL and Mediator faults. Oracle SOA Suite 11g fault management framework can be leveraged by writing custom fault-policies.xml file to declare possible faults and their associated actions. These policy files can then be associated to composites or components through the fault-bindings.xml file.

The following listing shows a sample fault policy with retry options for remote and binding faults:

Listing 2: Fault policy defined with retry options for remote and binding faults

```xml
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy">
  <faultPolicy version="2.0.1" id="SOAFaultPolicy"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:bpelx="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>
          <action ref="custom-retry"/>
        </condition>
      </faultName>
      <faultName
        xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
        name="bpelx:bindingFault">
        <condition>
          <action ref="custom-retry"/>
        </condition>
      </faultName>
    </Conditions>
    <Actions>
      <Action id="custom-retry">
        <retry>
          <retryCount>2</retryCount>
          <retryInterval>4</retryInterval>
          <retryFailureAction ref="custom-recovery"/>
        </retry>
      </Action>
      <Action id="custom-recovery">
        <javaAction>
          <className>custom.soa.extensions.custom.FaultAction</className>
          <defaultAction>ora-human-intervention</defaultAction>
          <propertySet>"Fault"</propertySet>
          <returnValue value="MANUAL">
            <ref>ora-human-intervention"/></ref>
        </javaAction>
      </Action>
      <Action id="ora-human-intervention">
        <humanIntervention/>
      </Action>
    </Actions>
  </faultPolicy>
</faultPolicies>
```
Listing 3: The following listing shows a sample fault policy with re-throw options for remote and binding faults. Fault policy defined with custom re-throw actions for remote and binding faults.

```xml
<faultPolicy version="2.0.1" id="SOAFaultPolicy"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bpelx="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>
        <action ref="custom-rethrow-action"/>
      </condition>
    </faultName>
    <faultName
      xmlns:bpelx=http://schemas.oracle.com/bpel/extension
      name="bpelx:bindingFault">
      <condition>
        <action ref="custom-rethrow-action"/>
      </condition>
    </faultName>
    <faultName
      xmlns:medns="http://schemas.oracle.com/mediator/faults"
      name="medns:mediatorFault">
      <condition>
        <action ref="custom-error-action"/>
      </condition>
    </faultName>
  </Conditions>
  <Actions>
    <Action id="custom-rethrow-action">
      <javaAction
        className="custom.soa.extensions.custom.FaultAction"
        defaultAction="ora-rethrow-fault" propertySet="Fault">
        <returnValue value="Ok" ref="ora-rethrow-fault"/>
      </javaAction>
    </Action>
  </Actions>
</faultPolicy>
</faultPolicies>
```
Fault Management in Mediator

Faults that arise in mediator can be handled using the Fault Policy Framework. Mediator Faults are identified using http://schema.oracle.com/mediator/faults namespace and fault name as mediatorFault from same namespace. Process to define and bind the Fault Policy to a SOA Component is as similar to any SCA Components.

Listing 4: Below XML snippet shows the sample fault Policy for handling Mediator Faults.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies>
  <faultPolicy version="1.0" id="SampleMediatorFaultPolicy">
    <Conditions>
      <faultName xmlns:medns="http://schemas.oracle.com/mediator/faults" name="medns:mediatorFault">
        <condition>
          <test>contains($fault.mediatorErrorCode, "TYPE_DATA")</test>
          <action ref="ora-retry"/>
        </condition>
      </faultName>
    </Conditions>
    <Actions>
      <Action id="ora-retry">
        <retry>
          <retryCount>5</retryCount>
          <retryInterval>6</retryInterval>
          <exponentialBackoff/>
          <retryFailureAction ref="ora-java"/>
```
<retrySuccessAction ref="ora-terminate"/>
</retry>
</Action>
</Actions>
</faultPolicy>
</faultPolicies>

**Mediator Error Groups:** Mediator Error Groups can be used in test conditions while defining the Fault Policy. There are five error groups in mediator, TYPE_ALL, TYPE_DATA, TYPE_METADATA, TYPE_FATAL, TYPE_TRANSIENT and TYPE_INTERNAL.
<table>
<thead>
<tr>
<th>S.No</th>
<th>Error Group</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TYPE_DATA</td>
<td>TYPE_DATA_ASSIGN</td>
<td>Error Related to Data Assignment can be identified using this Error Code.</td>
</tr>
<tr>
<td>2</td>
<td>TYPE_DATA</td>
<td>TYPE_DATA_FILTERING</td>
<td>Errors Related to Payload Data Filtering in Mediator can be identified using this Error Code.</td>
</tr>
<tr>
<td>3</td>
<td>TYPE_DATA</td>
<td>TYPE_DATA_TRANSFORMATION</td>
<td>Errors Related to Payload Data Transformation in Mediator can be identified using this Error Code.</td>
</tr>
<tr>
<td>4</td>
<td>TYPE_DATA</td>
<td>TYPE_DATA_VALIDATION</td>
<td>Errors related to Payload Validation in Mediator can be identified using this Error Code.</td>
</tr>
<tr>
<td>5</td>
<td>TYPE_METADATA</td>
<td>TYPE_METADATA_FILTERING</td>
<td>Error related to Meta Data Filtering in Mediator can be identified using this Error Code.</td>
</tr>
<tr>
<td>6</td>
<td>TYPE_METADATA</td>
<td>TYPE_METADATA_TRANSFORMATION</td>
<td>Error related to Metadata Transformation in Mediator can be identified using this Error Code.</td>
</tr>
<tr>
<td>7</td>
<td>TYPE_METADATA</td>
<td>TYPE_METADATA_VALIDATION</td>
<td>Error related to Validation in Mediator can be identified using this Error Code.</td>
</tr>
<tr>
<td>8</td>
<td>TYPE_METADATA</td>
<td>TYPE_METADATA_COMMON</td>
<td>Other Common Errors related handling Metadata can be identified using this Error Code.</td>
</tr>
<tr>
<td>9</td>
<td>TYPE_FATAL</td>
<td>TYPE_FATAL</td>
<td>Error that can be recovered and fault can be identified using this Error Codes specified for their own specific needs</td>
</tr>
<tr>
<td>10</td>
<td>TYPE_FATAL</td>
<td>TYPE_FATAL_DB</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>TYPE_FATAL</td>
<td>TYPE_FATAL_CACHE</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TYPE_FATAL</td>
<td>TYPE_FATAL_ERRORHANDLING</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TYPE_FATAL</td>
<td>TYPE_FATAL_MESSAGING</td>
<td></td>
</tr>
</tbody>
</table>
### Error Codes

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Second Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_FATAL</td>
<td>TYPE_FATAL_TRANSFORMATION</td>
<td>Error related Service Infrastructure can be identified using this Error Code.</td>
</tr>
<tr>
<td>TYPE_TRANSIENT</td>
<td>TYPE_TRANSIENT_MESH</td>
<td>Errors related to JMS such as Enqueue and Dequeue can be identified using this Error Code.</td>
</tr>
<tr>
<td>TYPE_TRANSIENT</td>
<td>TYPE_TRANSIENT_MESSAGE</td>
<td>Errors related to JMS such as Enqueue and Dequeue can be identified using this Error Code.</td>
</tr>
<tr>
<td>TYPE_INTERNAL</td>
<td></td>
<td>All Error related to Mediator internals can be identified using this Error Code.</td>
</tr>
<tr>
<td>TYPE_ALL</td>
<td></td>
<td>Above Specified Errors are grouped under this Error Code</td>
</tr>
</tbody>
</table>

*Fault policies takes higher precedence over catch and catchAll blocks.*

#### Oracle SOA Fault Policy Resolution Process

The following diagram helps you to understand the fault policy resolution process involved in identifying and handling faults:

**Step 1:** Faults generated from the SOA Components will be intercepted by Fault Management Framework.

![Fault Management Process Diagram](image-url)
Step 2: Appropriate Fault Policy Definition will be matched based on the fault policy configuration. If no Policy definitions are available, default fault policy will be used.

Step 3: Fault Action will be recognized from the fault policy definition and execute the required action.

Note: Fault Policy can be local to the composite or global policy; it will take in effect based on the fault binding configuration.

Extending SOA Fault Management Framework
SOA Fault should be handled and managed effectively to establish an effective SOA Architecture. Oracle SOA Fault Management Framework is a declarative environment through which the faults can be managed. Both system and business faults can be managed through fault management framework. Fault management framework provides features to manage both BPEL and Mediator faults. Oracle SOA Suite 11g fault management framework can be leveraged by writing custom fault-policies.xml file to declare possible faults and their associated actions. These policy files can then be associated to composites or components through the fault-bindings.xml file.

SOA Fault Management Framework is a combination of SOA Fault Handler Components, tools and guidelines. SOA Fault Handling Components will provide web services to handle and log faults with bunch of SOA Fault Policies designed to handles faults declaratively. The below design block shows the Extended SOA Framework characteristics and features provided by it.

Framework Components
SOA Fault Framework should be supported by three major components they are SOA Fault Handlers, SOA Fault Policy definitions (supported in Oracle SOA Products) and SOA Error Code and Action Configuration Repository and SOA Notification Framework
<table>
<thead>
<tr>
<th>Components</th>
<th>Component Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA Fault Handler</td>
<td>SOA Fault Handler should be able to handle faults that are generated from SOA Web Services. There can be three different versions of SOA Fault Handlers to support different patterns of Fault Handling. Asynchronous; Synchronous and One-Way Fault Handler. SOA Fault Handler</td>
</tr>
<tr>
<td>SOA Fault Policies</td>
<td>SOA Fault Policies are policies that are to be made available for the SOA Components to subscribe. SOA Fault Policies will define the fault handling mechanism for SOA Faults that occur in the SOA environments. Based on the Process type the fault policies could be provided.</td>
</tr>
<tr>
<td>SOA Error Codes and Action Configuration</td>
<td>SOA Error Codes and Action Configuration Repository should be in place to configure the list of enterprise level technical, business exception error code to handle the errors for the configured actions.</td>
</tr>
<tr>
<td>SOA Notification Framework</td>
<td>SOA Notification Framework is essential component to be integrated with SOA Fault Management Framework through Service calls and SOA Fault Policies.</td>
</tr>
</tbody>
</table>

**Features:**

- SOA Faults Handler should be made available in different interaction patterns; they are synchronous, asynchronous and one-way to support different patterns of SOA Faults Handling mechanism in different scenarios.
- SOA Faults Handler should be exposed as different interfaces through Java, XPath, WebServices, DB Calls and etc.
- SOA Fault should log the faults through the SOA Logger Framework to trace the faults and error messages through logs.
- SOA Fault Management Framework should define an Enterprise level fault definitions to provide enterprise level fault message format for different service components to interact.
• SOA Fault Management Framework should provide error code and action configuration Mechanism. This will be integrated with notification services and be integrated with Fault Policy factory.

• SOA Fault Management Framework should provide SOA Fault Policies to handle the faults through policy definitions.

• SOA Fault Management Framework should be able to connect to the notification service to send the email messages, console messages and SMS on critical alerts.

• SOA Fault Management Framework could be integrated with IT Service Desk if required to create SOA Middleware incident tickets and notify critical errors,

• SOA Fault Management Framework should be able to recover the messages through fault message recovery framework (In built Oracle SOA Fault Recovery Mechanism) can be implemented.

• SOA Fault Management Framework should be made extendable to handle custom and business faults.

Implementation Techniques:
Below snapshot shows the technical implementation view of SOA Fault Management Framework. Clients would access the SOA Fault Management Implementation through different wrapper interfaces that are exposed. Wrappers should push the Fault messages to Fault Management Queue. SOA Fault Management BPEL Component would dequeue the messages from Fault Management Queue and complete the logging process based on the Fault Management configuration. This can be achieved through Fault Policies and custom actions that perform the job that are intent to be executed by the wrappers implementation shown in the below screenshot.
### SOA Fault Handler

SOA Fault Handler component can be implemented using BPEL Process. If required a version of OSB wrapper service can be implemented to handle the faults.

### SOA Fault Policies

SOA Fault Policies can be defined using the Oracle SOA Fault Management framework, it is recommended to provide three types of fault policies by default one for Synchronous BPEL Services, Asynchronous BPEL Services and other one for Mediators.

### SOA Error Codes and Action Configuration

SOA Error Code and Action Configurations can be accomplished using ADF or Oracle Apex based system.

### SOA Notification Framework

SOA Notification Framework can be implemented by integrating the existing services or using UMS layer of Oracle SOA Suite.

### SOA Fault Header Details:

Below XML Schema Definition shows SOA Fault Definition. SOAFaultHandler Service could use this as input to the service to expose the Fault Message.

![XML Schema Diagram](image)

### Code Reference:

**Listing 5: SOA Fault definition (XSD Structure)**

```xml
<?xml version="1.0" encoding="windows-1252" ?>
```
Fault Interaction Patterns
Faults Interactions are established through three different patterns listed below.

- Replying Faults to Caller Service
- Throw Faults to Caller Service
- Using Fault Policy Actions to handle Faults

RePLYING Faults to Caller Service
RePLYing Faults to the caller is key pattern of communicating faults to callers in the service interactions. This will help the caller to understand the fault has occurred and it need to act upon in it based on the Fault Management configuration.

Throwing Faults to Caller Service
Throwing Faults (SOAP Faults – System as well as custom Faults) to the caller is one way of communicating the fault to the caller in the service interactions. This will help the caller to understand the fault has occurred and it need to act upon in it based on the Fault Management Configuration or Catch Blocks.

Fault Policy Definitions
Fault Policy Definition allows fault actions configurations to handle faults that occur in a process. Fault Policies can be configured to rethrow faults to caller.
Listing 6: Fault policy defined with re-throw options for remote faults

```xml
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy">
  <faultPolicy version="2.0.1" id="SOAFaultPolicy"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:bpelix="http://schemas.oracle.com/bpel/faultpolicy"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Conditions>
      <faultName xmlns:bpelix=http://schemas.oracle.com/bpel/extension
        name="bpelix:remoteFault">
        <condition>
          <action ref="ora-rethrow"/>
        </condition>
      </faultName>
    </Conditions>
    <Actions>
      <Action id="ora-rethrow-fault"><rethrowFault/></Action>
    </Actions>
  </faultPolicy>
</faultPolicies>
```

Impacts of Fault Management in SOA Design Patterns

The below table would provide the right strategy in handling faults for different SCA CI Patterns.

<table>
<thead>
<tr>
<th>Faults &amp; SCA CI Patterns</th>
<th>Invocation</th>
<th>Process Type</th>
<th>Throw Fault</th>
<th>Reply Fault</th>
<th>Fault Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoke Based</td>
<td>Synchronous</td>
<td></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
</tr>
<tr>
<td></td>
<td>One-Way</td>
<td></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td>N/A</td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
</tr>
<tr>
<td>Events</td>
<td>Event</td>
<td></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td>N/A</td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
</tr>
<tr>
<td>Adapters</td>
<td>Read/Poll</td>
<td></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
<td><img src="checkmark.png" alt="Checkmark" /></td>
</tr>
</tbody>
</table>

**Note:** Throwing fault and Replying the fault, both are essential. Replying fault communicates about the error that happens in the process, throwing faults helps the SOA EM (Enterprise Manager) to show the composite instance as faulted and halt the execution.

The table below shows the possible combinations of Interaction patterns between different SCA Interfaces exposed from the components. We have identified each of the interaction patterns with
an identifier. We will make use of the identifier to understand the impacts of the fault management in these interaction patterns.

The table below shows the Applicable Fault Interaction Patterns in various SCA CI Patterns.
<table>
<thead>
<tr>
<th>Interaction Pattern Number</th>
<th>Throw Fault</th>
<th>Reply Fault</th>
<th>Fault Policy Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Sync &gt; Sync</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Sync &gt; One-Way</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Sync &gt; Adapter (Write)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Sync &gt; Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ASync &gt; Sync</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ASync &gt; ASync</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>ASync &gt; One-Way</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>ASync &gt; Adapter (Write)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>ASync &gt; Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>One-Way &gt; Sync</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>11</td>
<td>One-Way &gt; ASync</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>12</td>
<td>One-Way &gt; One-Way</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>13</td>
<td>One-Way &gt; Adapter (Write)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>14</td>
<td>One-Way &gt; Event</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>15</td>
<td>Adapter (Invoke) &gt; Sync</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>16</td>
<td>Adapter (Invoke) &gt; ASync</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>17</td>
<td>Adapter (Invoke) &gt; One-Way</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>18</td>
<td>Adapter (Invoke) &gt; Adapter (Write)</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>19</td>
<td>Adapter (Invoke) &gt; Event</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>20</td>
<td>Event (Invoke) &gt; Sync</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>21</td>
<td>Event (Invoke) &gt; ASync</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
How to create Custom Fault Actions in Fault Management Framework?
Fault management framework is a declarative environment through which the faults can be managed. Both system and business faults can be managed through a fault management framework. The fault management framework provides features to manage both BPEL and Mediator faults. The Oracle SOA Suite 11g fault management framework can be leveraged by writing custom fault-policies.xml files to declare possible faults and their associated actions. These policy files can then be associated with composites or components through the fault-bindings.xml file.

Listing 7: The following are composite.xml properties to identify fault policies.
<property name="oracle.composite.faultPolicyFile">
  oramds://apps/faultpolicyfiles/fault-policies.xml
</property>
<property name="oracle.composite.faultBindingFile">
  oramds://apps/faultpolicyfiles/fault-bindings.xml
</property>

Listing 8: The following code shows the implementation details of the custom fault action package com.fmf.customfault;

Java Fault Action should be implemented by using IFaultRecoveryJavaClass
Deploy the Package to Custom Fault Action Jar
Attach the Custom Action to Fault Policy
import com.collaxa.cube.engine.fp.BPELFaultRecoveryContextImpl;
import oracle.integration.platform.faultpolicy.IFaultRecoveryContext;
import oracle.integration.platform.faultpolicy.IFaultRecoveryJavaClass;

public class CustomJavaAction implements IFaultRecoveryJavaClass {
    public void handleRetrySuccess(IFaultRecoveryContext ctx) {
        System.out.println("handle retry success");
        handleFault(ctx);
    }

    public String handleFault(IFaultRecoveryContext ctx) {
        System.out.println("Action context:
" + ctx.toString());
        BPELFaultRecoveryContextImpl bpelCtx = (BPELFaultRecoveryContextImpl) ctx;
        bpelCtx.addAuditTrailEntry("hi there");
        System.out.println("Policy Id: " + ctx.getPolicyId());
        System.out.println("Composite Name: " + bpelCtx.getCompositeName());
        Element payload = (Element) bpelCtx.getVariableData("inputVariable", "payload", "/"));
        Node node = payload.getFirstChild();
        return "ABORT";
    }
}

Copy the JAR file to the server and put it inside the oracle.soa.ext_11.1.1 directory, you should find it inside the SOA Suite product's directory.

Listing 9: The following code shows the implementation details of the fault policy configuration

```xml
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <faultPolicy id="customPolicy" version="0.0.1"
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Conditions>
            <faultName>
                <condition>
                    <action ref="custom-java"/>
                </condition>
            </faultName>
        </Conditions>
        <Actions>
            <Action id="custom-java">
```
<javaAction className="com.fmf.customfault.CustomJavaAction"
  defaultAction="default-human-intervention">
  <returnValue value="ABORT" ref="default-abort"/>
</javaAction>
</Action>
</Actions>
</faultPolicy>
</faultPolicies>