Oracle BPEL 10.1.3.3 Fault Policy Management

Veel nieuwe functionaliteit

In de SOA Suite 10.1.3.3 is veel nieuwe functionaliteit voor het afhandelen van fouten opgenomen. Deze maakt het de ontwikkelaar een stuk makkelijker en draagt bij aan het schoon houden van BPEL-processen. Foutafhandeling is weliswaar noodzakelijk, maar was in eerdere versies iets te nadrukkelijk aanwezig in BPEL-processen. In dit artikel zetten we uiteen wat de mogelijkheden van dit framework zijn. Aan de hand van een voorbeeld doorlopen we stap voor stap een praktische invulling daarvan.

In BPEL-processen kunnen we de fouten categoriseren in *business*fouten en runtimefouten.
Business fouten zijn applicatiespecifiek en zullen alleen optreden als we gebruik maken van een 'throw' activiteit of als we een partnerlink 'invoken' welke terug komt met een bepaalde fout.
Het is aan de *business* om te bepalen wat er moet gebeuren met de fout en welke activiteiten er nodig zijn om de fout af te handelen.
Naast de *business*fouten hebben we nog de *runtime*fouten. Runtime fouten worden niet gedefinieerd door de ontwikkelaar, en daarom moeten we in het proces een stuk afhandeling definiëren om de fout af te handelen. In BPEL zullen we in zo'n geval gebruik kunnen maken van de 'catch' activiteit. In de 'catch' kunnen we dan de onze eigen foutafhandelingprocessen aanroepen of bepaalde andere activiteiten (retry e.d.) uitvoeren om ons proces weer in een valide staat te krijgen.

Op de Oracle site staat een goede technote over alle mogelijke fouten en de extra fouten die gedefinieerd zijn binnen de Oracle BPEL-process Manager: ("http://www.oracle.com/technology/products/ias/BPEL/htdocs/oraBPEL_technotes.tn007.html").

Framework

In de goede oude tijd (vóór patch 10.1.3.3) moesten we per BPEL-proces de foutafhandeling definiëren. Op dat moment konden we gebruik maken van bijvoorbeeld een Error Hospital, maar we moesten nog steeds elk proces aanpassen om in staat te zijn de foutafhandeling toe te passen. Met de laatste patch voor de Oracle SOA Suite (10.1.3.3), biedt Oracle ons een nieuw Fault Management framework. Dit framework geeft ons de mogelijkheid om alle business- en runtime-fouten voor een 'invoke' activiteit af te vangen. In het framework definiëren we per BPEL domain één policy.

In een zogenaamde *fault-policy-binding* kunnen we configureren welke policies we gaan gebruiken en welke processen/partnerlinks en porttypes gebruik zullen maken van een policy. Het framework zal de gedefinieerde binding gebruiken in de volgende volgorde van prioriteit:
• BPEL.xml
• policy gedefinieerd op de server

In een zogenaamde *fault-policy* kunnen we configureren welk type fout we willen afvangen en welke acties er ondernomen moeten worden om de fout af te handelen. Indien er in het BPEL-proces al een foutafhandeling is gedefinieerd (catch), dan zal het framework deze overrulen en gebruiken maken van een policy, mocht deze gedefinieerd zijn.

Fault-policy-binding

De *fault-policy-bindings* worden op de volgende locatie gedefinieerd: /BPEL/domains/default/config/fault-policy-binding.xml

Het xsd-schema voor de fault-policy-bindings kan worden gevonden op de volgende locatie: /BPEL/system/xmllib/fault-policy-binding.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicyBindings version="2.0.1"
                     xmlns="http://schemas.oracle.com/BPEL/faultpoli-
cy"
                     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>
```

In een policy kunnen we op verschillende levels (proces,partnerLink,portType) definiëren of we gebruik willen maken van een policy. In het commentaar van de xml staat een goede uitleg over welke tag zorgt voor welke functionaliteit.

Fault-policy

De *fault-policies* worden op de volgende locatie gedefinieerd: /BPEL/domains/default/config/fault-policies

Het xsd-schema voor de *fault-policies* kan worden gevonden op de volgende locatie: /BPEL/system/xmllib/fault-policy.xsd

Oracle levert de volgende policy standaard mee:

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicy version="2.0.1" id="DefaultPolicy" xmlns:env="http://sche-
mas.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 termina-
te-->
                <Action id="ora-terminate">
                        <abort/>
                </Action>
        </Actions>
</faultPolicy>
```

Een policy bestaat uit een *conditions*-gedeelte en een *actions*-gedeelte. In het condition-gedeelte kunnen we definiëren welke fouten we willen afhandelen en wat de *actions* zijn om uit te voeren. In de onderstaande code zal het framework alle 'remoteFaults' afvangen en de *action* 'ora-retry' uitvoeren.

```
<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>
</Conditions>
```

In het volgende voorbeeld zal het framework controleren of de 'fault-code' gelijk is aan de waarde 'INVALID_WSDL'. De actie die vervolgens zal worden uitgevoerd is: 'ora-terminate'.

```
<condition>
        <test>$fault.code/code="INVALID_WSDL"</test>
        <action ref="ora-terminate"/>
</condition>
```

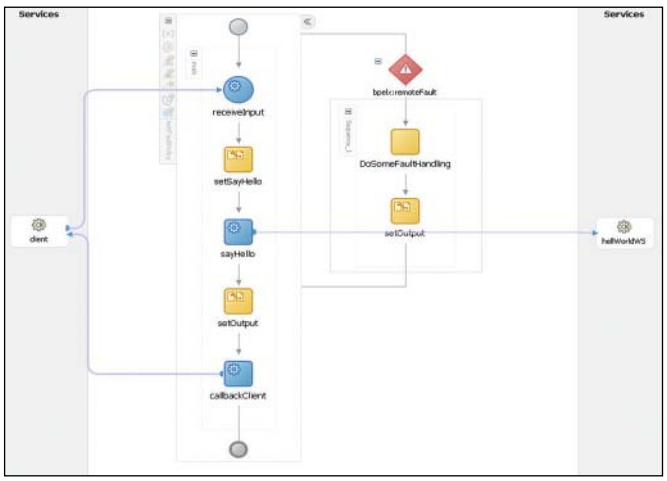
Oracle levert bij het framework een standaard lijst met *actions* mee. Op dit moment is het niet mogelijk om deze lijst uit te breiden met eigen actions. Mochten we toch gebruik willen maken van zelfgedefinieerde foutafhandelingen dan kan er gebruik worden gemaakt van de action 'ora-java'. Deze action maak het mogelijk om zelf eigen java-code te definiëren waarin beslist kan worden wat er moet gebeuren met de opgetreden fout.

Standaard aanwezige actions:

- ora-retry: retry de activiteit
- ora-replay-scope: replay de scope waarin de fout is opgetreden
- ora-rethrow-fault: system zet de fout door naar de 'BPEL fault handlers'
- ora-human-intervention: de huidige activiteit krijgt een *freeze* status. In de console zal de gebruiker dan de mogelijkheid hebben om de verschillende *actions* uit te voeren
- ora-terminate: terminate de instance
- ora-java: uitvoeren van zelfgedefinieerde java-code

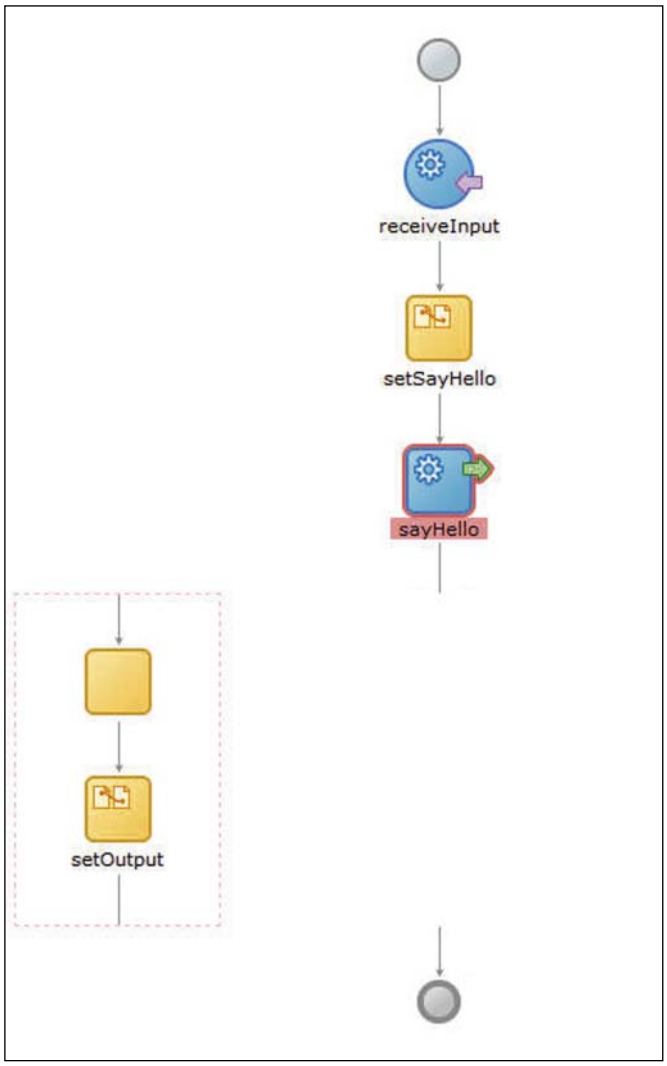
De praktijk!

De werking van het framework zullen we nu laten zien aan de hand van een klein voorbeeld. We hebben een simpele java-webservice (helloWorld) gedeployed. Daarnaast hebben we een klein BPEL-proces met een paar activiteiten erin (assign, invoke van de java webservice, etc). Als foutafhandeling hebben we een catch gedefinieerd voor de 'remoteFault'. In de catch zullen we geen verdere foutafhandeling doen, behalve het assignen van de output van het proces.



Afbeelding 1

Om een 'remoteFault' te genereren hebben we de Java-webservice ge-undeployed, en daarna voeren we het BPEL-proces opnieuw uit. Na het uitvoeren zal het proces in de catch komen die we hebben gedefinieerd.



Afbeelding 2

Aan de structuur van het proces zullen we nu niks veranderen. Op de server definiëren we nu een policy (fault-bindings.xml).

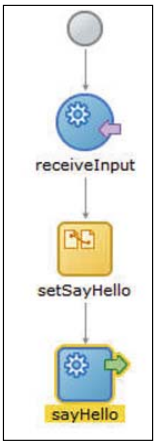
```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicyBindings version="2.0.1"
                        xmlns="http://schemas.oracle.com/BPEL/faultpoli-
cy"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
        <process faultPolicy="BlogTestPolicy"/>

<!-- DefaultPolicy is defined in ./fault-policies/DefaultPolicy.xml -->
        <partnerLink faultPolicy="BlogTestPolicy"/>
</faultPolicyBindings>
```

De policy is genaamd "BlogTestPolicy" en alle processen en partnerlinks zullen er gebruik van maken. De volgende stap is het definiëren van de policy (BlogTestPolicy.xml).

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicy version="2.0.1" id="BlogTestPolicy" xmlns:env="http://sche-
mas.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>
                        <condition>
                                <action ref="ora-human-interventi-
on"/>
                        </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 termina-
te-->
                <Action id="ora-terminate">
                        <abort/>
                </Action>
        </Actions>
</faultPolicy>
```

We hebbben nu een policy gedefinieerd welke alle *remoteFaults* zal gaan afhandelen. Mocht er een remoteFault optreden dan zal het framework 2 *actions* uitvoeren. Als eerste de 'ora-retry' action, mocht de fout daarna nog steeds optreden, dan zal de tweede 'action' worden uitgevoerd, de 'ora-human-intervention' action. Na het aanmaken van de policy restarten we de oc4j-container om te zorgen dat de policy actief wordt.
Voordat we met de test beginnen zorgen we ervoor dat we dezelfde situatie hebben als in het begin. We hebben de Java-webservice welke we aanroepen in het proces ge-undeployed. De catch in het proces laten we gewoon staan. Op de server hebben we een policy gedefinieerd. Dit policy zal alle *remote-Faults* afhandelen in plaats van de *catch* in het proces zelf.

Als we het proces uitvoeren en vervolgens in de BPEL console gaan kijken bij de instance zien we het volgende:



Afbeelding 3

De catch van het proces zelf is niet uitgevoerd. Als we nu in de audit-trail gaan kijken zien we de volgende melding (zie afbeelding 3a):

Het ziet ernaar uit dat onze policy actief is. Het framework heeft de policy toegepast. Het heeft als eerste de 'ora-retry' action uitgevoerd. Van deze action zien we geen output in de audit-trail. Na de 'ora-retry' is de action 'ora-human-intervention' uitgevoerd. Dus op dit moment zouden we in staat moeten zijn om in de consolle verschillende *actions* uit te voeren op de activiteit in het proces waarbij de foutafhandeling is gaan optreden ('invoke' van de java webservice). Vanaf dit punt hebben we twee opties:

```
[2007/08/17 19:45:31]
[FAULT RECOVERY] Marked Invoke activity as "pending manual recovery".
```

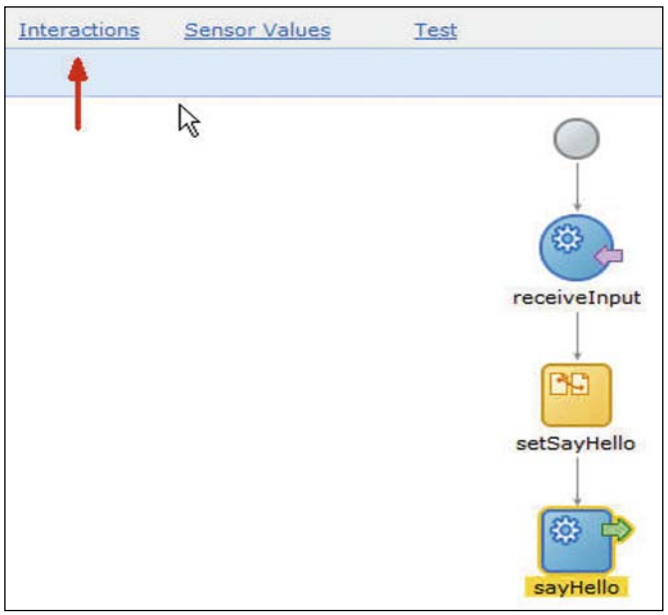
Afbeelding 3a

De eerste optie is het *activities*-tabblad. Op dit tabblad vinden we alle *activities* die zijn uitgevoerd en waarbij fouten zijn opgetreden met bijhorende foutnaam. Aan de linkerkant selecteren we de 'Pending' state in de 'Activity State' selectbox. Aan de rechterkant zien we de foutgelopen 'invoke' activiteit van onze webservice. Als we de checkbox aan de linkerkant van de 'activities' selecteren kunnen we gebruik maken van de 'bulk update' functionaliteit op meerdere instances.
We selecteren de 'invoke' activiteit en in de listbox kunnen we de volgende actions selecteren :

- Retry
- Abort
- Rethrow

• Replay
• Continue

De tweede optie is het tabblad *interactions*, welke beschikbaar is wanneer we de foutgelopen instance selecteren.



Afbeelding 4

In dit scherm zien we alle activiteiten van het huidige proces instance. Selecteer dan de 'invoke' activiteit (sayHello). In het scherm dat dan volgt zien we een soort van *Error Hospital*. Daarin wordt (rechtsboven) de status van de huidige instance getoond: 'open.pending.recovery'.

Afbeelding 5

In dit scherm hebben we de mogelijkheid om de payload van de variabelen van de activiteiten in het proces te updaten, verschillende *actions* uit te voeren, 'chain' van *actions* of de mogelijkheid om een nieuwe instance te creëren.

We selecteren de variabele in de "Available Variables" listbox voor welke we de payload willen updaten. We klikken op de button. In het tekstveld krijgen we dan de mogelijkheid om nieuwe payload in te voeren. Na het invoeren van de juiste waardes, klik de 'Set' button en gaan we naar de "Actions available" listbox. Daar selecteren we de gewenste action. De laatste stap is de keuze om de huidige instance te herstellen (recover) of om een nieuwe instance te creëren.

In ons voorbeeld hebben we de service opnieuw gedeployed, dus de wsdl is weer bereikbaar. We hebben de sayHello input-variabele geupdate zodat we later in de herstelde instance de veranderingen kunnen waarnemen. Daarna klikken we de 'Recover' button en selecteren we de visual flow, en bekijken de 'invoke' activiteit (sayHello).

Afbeelding 6

Zoals we kunnen zien in de output is de payload van de input-variabele van de 'sayHello' activiteit ge-update. We hebben webservice opnieuw ge-deployed, zodat deze weer bereikbaar was, en hierna is de 'invoke'-activiteit wél correct uitgevoerd.

Conclusie

Met het nieuwe Fault Policy Management-framework biedt Oracle ons de mogelijkheid om op vrij eenvoudige manier, en generiek, foutafhandeling toe te passen op BPEL-processen. Het framework is nog vrij nieuw, dus de komende tijd zal ongetwijfeld verder worden uitgebreid met functionaliteit. In vrij korte tijd zijn de verschillende configuratiebestanden aangemaakt, en kan het framework in gebruik worden genomen. Dit scheelt de ontwikkelaars erg veel tijd vergeleken met de situatie waarin we alle processen stuk voor stuk moesten gaan aanpassen.

Eric Elzinga, IT-eye