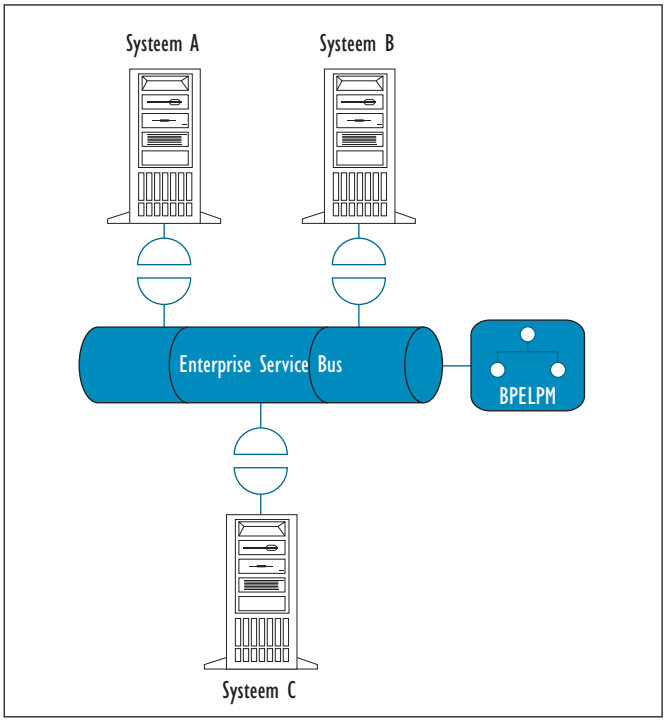
Foutafhandeling met Oracle ESB

Functionaliteit bovenop het adapter framework

Oracle heeft op OpenWorld 2006 de productieversie van Oracle Enterprise Service Bus, ESB, op de markt gebracht. Kernwoorden die Oracle gebruikte in de presentatie over Oracle ESB waren 'reliable messaging' en 'guaranteed delivery'. In deze presentatie werd uit de doeken gedaan hoe Oracle haar ESB zo robuust mogelijk heeft gemaakt. Maar wat nu als er, ondanks alles, alsnog een fout optreedt waardoor een bericht niet wordt afgeleverd?

Oracle introduceerde de verzamelterm "Error Hospital" als de component die binnen ESB gebruikt wordt voor het opslaan en opnieuw verzenden van berichten die toch om wat voor reden dan ook fout zijn gegaan. Hoe is deze Error Hospital nu te gebruiken voor het signaleren van fouten en voor het correct afhandelen van deze fouten? In dit artikel wordt gekeken naar de foutafhandeling in Oracle ESB en hoe de "Error Hospital" mogelijkheden te gebruiken zijn.



Figuur 1. Schematische weergave ESB

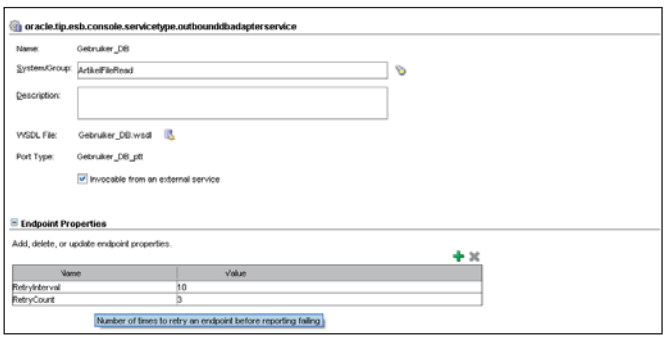
Adapter Framework foutafhandeling

Adapters verzorgen de technische koppeling tussen diverse systemen en zijn om die reden een belangrijk onderdeel van de SOA Suite. Ze verzorgen de verbinding met de "buitenwereld". Dit kunnen andere systemen zijn, zoals bijvoorbeeld databases, JMS queues, topics of FTP servers. Als er een fout optreedt in een adapter, dan moet deze netjes opgevangen en afgehandeld worden. Alle meegeleverde adapters zijn ontwikkeld volgens de Java Connector Architecture (JCA) standaard. JCA ondersteunt de volgende methoden om fouten te verwerken:

- Endpoint retry
- Rejection Handlers

Endpoint retry

Op het moment dat er een fout optreedt bij een uitgaande verbinding, dan is het mogelijk om dezelfde verbinding nogmaals aan te roepen. Er moet dan wel sprake zijn van een herstelbare fout (een zogenaamde "PCRetryableResourceException"), bijvoorbeeld wanneer de database waarnaar men een verbinding probeert te maken niet te bereiken is.
Deze foute berichten kunnen opnieuw worden aangeboden



Figuur 2. Endpoint properties in JDeveloper

door een "retry policy" voor de adapter aan te maken. Hoewel deze methode nog niet officieel ondersteund wordt, kan deze toch gebruikt worden door JDeveloper op te starten met de parameters -J"-Dpreview_mode=true". Nu is het mogelijk om bij een adapter "endpoint properties" te definiëren zoals in figuur 2 te zien is. Dit geldt overigens voor JDeveloper 10.1.3.1 en 10.1.3.2. In JDeveloper 10.1.3.3 is deze feature standaard.

Hetzelfde effect kan bereikt worden door het .esbsvc bestand aan te passen. Daarvoor moet na de </invocation> tag het volgende XML fragment geplaatst worden:

```
<endpointProperties>
    <property name="RetryInterval" value="15"/>
    <property name="RetryCount" value="3"/>
 </endpointProperties>
```

Het bovenstaande XML fragment zorgt ervoor dat er drie keer, met een tussentijd van vijftien seconden, een poging wordt gedaan om verbinding te maken. Na drie mislukte pogingen wordt er een "RemoteFault" teruggegeven.

Rejection Handlers

Bij binnenkomende verbindingen bestaat de mogelijkheid om een "Message Rejection Handler" te definiëren. Hiermee kan een ontwikkelaar een alternatief proces starten zodra er een fout bij het ontvangen van gegevens optreedt. Net zoals de "Endpoint Retry" word de Rejection Handler gedefinieerd in JDeveloper als deze opgestart is in preview mode of door de XML source van de adapter service aan te passen. Er zijn vier verschillende Rejection Handlers:

- File;
- RAW AQ;
- BPEL;
- WSIF.

File system based Rejection Handler

Deze Rejection Handler is het makkelijkst te gebruiken en schrijft de foutmelding weg naar een bestand. De syntax is als volgt:

```
INVALID_MSG_ + <proces-naam> +<operatie-naam>+<huidige-tijd>
```

Het bestand bevat de foutmelding. Om dit bestand weg te laten schrijven moet de volgende endpoint property worden gedefinieerd:

```
<endpointProperties>
      <property name="rejectedMessageHandlers"
value="file://<directory-pad>"/>
</endpointProperties>
```

RAW Oracle Advanced Queue based Rejection Handler

Het is ook mogelijk om de foutmelding op Oracle AQ te zetten Het bericht komt dan tijdelijk in een queue om deze later uit te lezen en actie te ondernemen. De endpoint properties hiervoor zijn als volgt:

```
<endpointProperties>
<property name="rejectedMessageHandlers" value="queue://
jdbc:oracle:thin:@<db-host>:<tns-poort>:<sid>|<gebruiker>/<wachtwoord>|<
queue-naam>"/>
</endpointProperties>
```

Het wachtwoord kan versleuteld worden door middel van het script encrypt.bat, dat in de <SoaHome>\bpel\bin directory staat.

BPEL Process Rejection Handler

Dit is één van de krachtigste Rejection Handlers omdat hiermee meteen een BPEL Proces opgestart kan worden om de fout te corrigeren of om een beheerder te waarschuwen. Er zijn een aantal randvoorwaarden om de BPEL Process Rejection Handler te kunnen gebruiken.

Aanpassingen op de WSDL van het BPEL proces:

1. De WSDL "RejectionMessage.WSDL" moet geïmporteerd worden:

```
<import namespace="http://xmlns.oracle.com/pcbpel/rejectionHandler"
location="http://<host>:<poort>/orabpel/xmllib/jca/RejectionMessage.
wsdl"/>
```

2. De definitie van de Rejection Handler moet toegevoegd worden.

```
xmlns:rej=http://xmlns.oracle.com/pcbpel/rejectionHandler
```

3. Een referentie moet aangemaakt worden naar de RejectionMessage in het element portType.

```
<portType name="MyRejectionHandlerPortType">
   <operation name="myHandleRejectionOperation">
      <input message="rej:RejectionMessage"/>
   </operation>
</portType>
```

Als het BPEL-proces nu gedeployed wordt, moet de Rejection Handler in de ESB adapter als volgt geconfigureerd worden:

```
<endpointProperties>
<property name="rejectedMessageHandlers" value="bpel://<bpel-
domein[:<wachtwoord>]>|<proces-naam>|<operatie-naam>|<input-bericht>"/>
</endpointProperties>
```

Ook hier kan het wachtwoord worden versleuteld door encrypt.bat.

WSIF Based Rejection Handler

Web Services Invocation Framework (WSIF) is een Java API om web services aan te roepen. Het maakt daarbij niet uit hoe of waar dat de services vandaan komen. Voor dit framework kan een Rejection Handler gedefinieerd worden op de volgende manier:

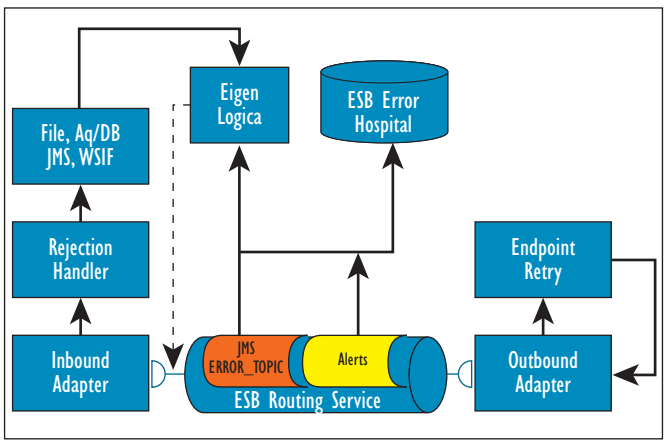
```
<endpointProperties>
<property name="rejectedMessageHandlers" value="wsif://<wsif-WSDL-loca-
tie>|< operatie -naam>|<input- bericht >
</endpointProperties>
```

Hierbij geldt hetzelfde als bij het bovenstaande BPEL Proces; de messagetype moet "RejectionMessage" zijn.

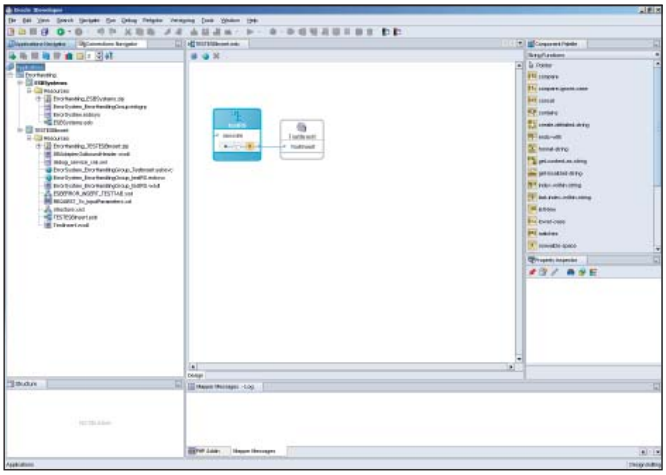
Samenvattend

De adapters van Oracle ESB bieden veel mogelijkheden om fouten af te handelen. De "Endpoint Retry" kan via een extra opstartparameter geconfigureerd worden in JDeveloper. Wanneer er met "Rejection Handlers" gewerkt gaat worden dan moeten er handmatig aanpassingen gedaan worden in een aantal XML bestanden. Er is grote flexibiliteit omdat er veel verschillende soorten Rejection Handlers aangeboden worden.

Hoe worden de fouten afgehandeld die niet binnen het adapter framework vallen? Bijvoorbeeld wanneer bij gebruik van een database adapter de aanroep van een database functie mis gaat?



Figuur 3. Architectuur foutafhandeling.



Figuur 4. Voorbeeld ESB proces in JDeveloper.

Foutafhandeling binnen ESB

In figuur 3 is te zien hoe de foutafhandeling intern verloopt binnen ESB. Als een flow binnen ESB synchroon is gedefinieerd dan wordt de fout teruggestuurd naar de aanroepende partij en is het mogelijk om de adapter rejection handler te starten. Onafhankelijk van het soort flow, synchroon of asynchroon, wordt de JMS ESB_ERROR topic gevuld , is het mogelijk om een notificatie te versturen en wordt het bericht opgeslagen in de ESB repository in de ESB_FAULTED_INSTANCE tabel.

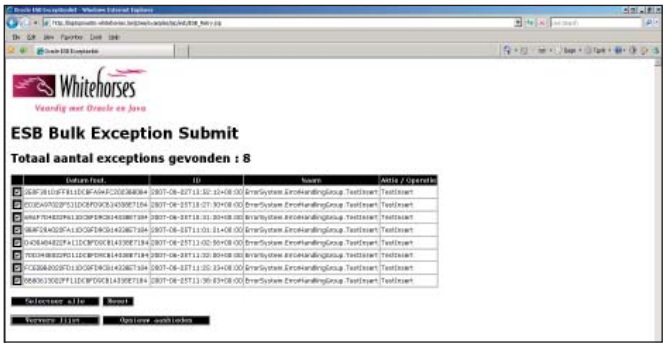
Voor onderstaand voorbeeld is gebruik gemaakt van een simpel "fire and forget" integratiescenario. De database adapter roept een procedure aan. Deze procedure slaat een record op in een tabel. De procedure is invalid gemaakt zodat de aanroep altijd een foutmelding teruggeeft.

Notificaties

Na de deployment van bovenstaand scenario is in de ESB Console deze flow aangepast. In de ESB Console is voor deze flow een Email ID toegevoegd onder de notification details. Alle fouten die worden veroorzaakt in deze flow worden naar dit emailadres gestuurd. Hieronder is te zien hoe deze e-mails eruitzien. Wat opvalt is dat er twee e-mails zijn binnengekomen: één bericht voor een fout in de database adapter en één melding voor een fout in de Routing Service. We zien dezelfde informatie als we met HermesJMS de ESB_ERROR en ESB_MONITOR topics volgen. HermesJMS is een gratis programma dat onder andere JMS queues en topics kan monitoren. Dit levert een enorme hoeveelheid aan foutberichten op die richting het beheerteam gaat. Er zijn geen instellingen om deze berichtenstroom in te dammen en/of samen te vatten.

Instance resubmit

In de ESB Console kunnen we alleen losse instances bekijken, aanpassen en opnieuw aanbieden. Daarbij moet een aantal stap-



Figuur 5. ESB_Retry.jsp. Overzicht van alle resubmittable exceptions.

pen doorlopen worden voordat het bericht opnieuw wordt aangeboden. Het is niet mogelijk om in batches meerdere berichten tegelijkertijd opnieuw aan te bieden.

Is de door Oracle ESB geboden functionaliteit voldoende?

Het antwoord hierop is nee. Het is bijvoorbeeld niet mogelijk om standaard alle berichten nogmaals aan te bieden of om dit met een ingestelde policy te doen. Het is ook niet mogelijk om dieper op de notificaties en alerts in te gaan door bijvoorbeeld elk uur een e-mail te versturen met hierin het aantal berichten dat fout is gegaan.

Deze functionaliteit moet zelf gebouwd worden met bijvoorbeeld BPEL of Java door middel van een Message Driven Bean. In onderstaand voorbeeld gaan we kijken hoe we de eerder beschreven functionaliteit, notificaties en bulk resubmit, met onder andere Oracle BPEL en een Oracle database snel kunnen implementeren.

Dit BPEL proces luistert naar het ESB_ERROR topic. De payload die binnenkomt is gedefinieerd als opaque. Dit houdt in dat de volledige input als binaire data wordt ingelezen zonder conversie naar XML. Dit proces converteert met een stuk embedded Java deze data naar een string en geeft deze string door aan een database adapter. Tevens worden de JMS header properties met een variabele ook doorgegeven aan de database.

Vervolgens verstuurt een database job elk uur een rapportage naar een beheerder. Op deze manier kunnen de verschillende fouten eenvoudig worden geaggregeerd en gemaild.

Het controleren van de ESB_ERROR topic had ook met de JMS Adapter in Oracle ESB gekund en gezien de positionering van Oracle ESB is dat een logische architectuur. Toch is hier expres niet voor gekozen. Wat gebeurt er namelijk als er een fout optreedt tijdens het uitlezen en routeren van deze error flow? Op dat moment wordt er een nieuw bericht op de ESB_ERROR topic geplaatst en belandt het proces in een oneindige lus.

Zoals eerder aangegeven is het niet mogelijk om met de ESB Console meerdere berichten tegelijkertijd opnieuw aan te bieden. Om dit mogelijk te maken is een aparte Client API nodig die beschikbaar is op OTN. Met deze API kunnen instances bekeken en in batch opnieuw aangeboden worden. Dit is te zien in figuur 5. De bijbehorende code ziet er als volgt uit:

```
String doresubmit = request.getParameter ("RESUBMIT");
If ( doresubmit != null && doesubmit.contains("ReSubmit") )
{
          String flowids[] = request.getParameterValues( "FLOWIDS" );
          String systemids[] = request.getParameterValues( "SYSTEMIDS"
)
          If(flowids != null && systemids != null )
          {
                    for ( int j = 0; j < flowids.length; j++ )
                    {
                              String thisflowid = flwoids[j].toSt-
ring();
                              String thisssytemid = systemids[j].toSt-
ring();
                              HashMap<String, String> requestProps =
new HashMap<String, String>();
                              RequestProps.put( "flowId", thisflowid
);
                              RequestProps.put( "systemId", thissy-
stemid );
                              Client.perform( "ResubmitInstanceById",
requestProps );
                    }
          }
}
```

Deze code is ingebouwd in een JSP pagina om de bulk submit functionaliteit te demonstreren, maar soortgelijke code kan uiteraard ook worden gemaakt voor het automatisch opnieuw aanbieden van deze instances.

Samenvatting

Foutafhandeling is een onmisbaar onderdeel op elk willekeurig project, ongeacht de gebruikte technologie en architectuur. In dit artikel is gedemonstreerd hoe de foutafhandeling binnen Oracle ESB te gebruiken is. Oracle heeft hiervoor de term "Error Hospital" geïntroduceerd. De functionaliteit beperkt zich tot het opslaan van fouten, het versturen van notificaties en het handmatig opnieuw verzenden van losse fouten door middel van de ESB Console. In dit artikel is ter sprake gekomen hoe extra functionaliteit bovenop het adapter framework en de Error Hospital gebouwd kan worden om fouten nog beter te verwerken.

Martin Kleinman en **Marcel Maas** werken als SOA Suite specialisten bij Whitehorses BV in Nieuwegein.