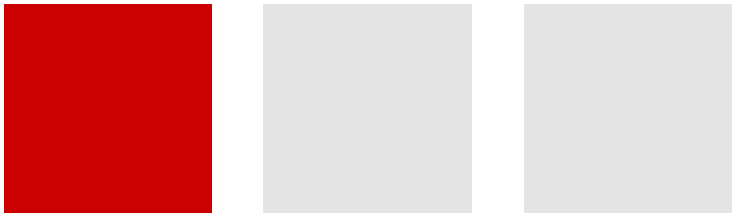


trivadis
makes IT easier.

# Batch Processing

# Hinweise zum Design von Batchprozesen in SCA Composites

Matthias Furrer
Principal Consultant
Dezember 2013

**Dieses Dokument beschreibt allgemeine Hinweise und best-practices zum Design von Batchprozessen mit Oracle SOA Suite. Folgende Themenbereiche werden behandelt:**

- **Einlesen von grossen Files im Streaming Mode**
- **Einsatz von Pre- und Postprocessing Handler für Logging**
- **Fehlerbehandlung mit Wiederaufsetzpunkten über Fault-Policies**

**Die in diesem Dokument beschriebenen Szenarien wurden getestet mit Oracle SOA Suite 11.1.1.6 (11gR1 PS5).**

# 1 Einlesen von grossen Files im Streaming-Mode

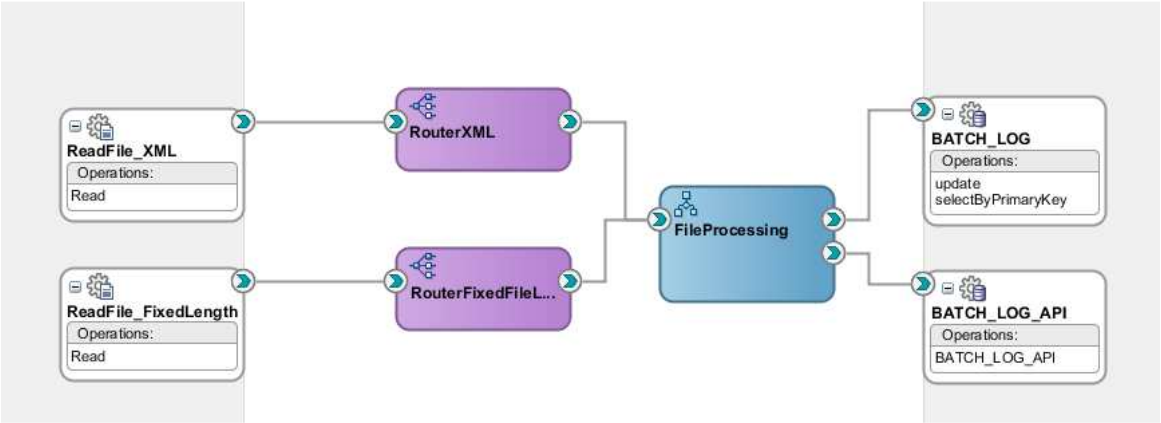
In einer Batchverarbeitung können sehr grosse Import-Files mit bis zu mehreren 10'000 Records pro Datei zur Verarbeitung angeliefert werden. Um das System vor Überlastung zu schützen muss verhindert werden, dass der gesamte Inhalt der Datei in den Payload und damit in eine DOM-Struktur im Memory aufgebaut wird. Weiter wird ebenfalls aus Performancegründen von Oracle empfohlen, keine grösseren Iterationen in einer BPEL Komponente zu verwenden.
Oracle SOA Suite stellt im JCA FileAdapter einen Mechanismus zur Verfügung, der Streaming beim Lesen von Dateiinhalten unterstützt und die Aufteilung des gelesenen Inhalts auf eine konfigurierbare Anzahl Instanzen erlaubt.

Damit ergeben sich folgende Verabeitungsschritte im jeweiligen SCA Composite für die Verarbeitung eines Files:

1) Einlesen des Files über JCA File-Adapter im Streamingmode
2) Initiierung der SCA Instanzen gemäss konfigurierter Publish-Size (Anzahl Records pro Instanz)
3) Transformation in Mediator in benötigtes Format des BPEL-Komponentenservice (falls notwendig, z.B. bei mehreren Importformaten für die gleiche Verarbeitung).
4) Recordweise-Verarbeitung in BPEL Komponente

Die ideale Anzahl Records pro Verarbeitungsinstanz (Mediator/BPEL Komponente) muss – ggf. individuell – in Lasttests noch ermittelt werden – bewegt sich aber erfahrungsgemäss im Rahmen von 100-1000 Records.

Folgende Ansicht zeigt graphisch die Service-Komponenten im Beispielprojekt :




Weiter sind bei der Definition des Composites folgende Punkte zu beachten:

- Propagierung der JCA Properties von Mediator zum BPEL-Prozess
- Deklaration der Batch-Notification Handler (Kapitel 2)
- Zuordnung der Fault-Policy Definitionen (Kapitel 3)

## 1.1 Standardverhalten des File-Adapters im Fehlerfall

Im Falle eines Fehlers beim Einlesen der Datei – bspw. aufgrund einer fehlerhaften Recordstruktur – verhält sich der Adapter wie folgt:

- Kein Einfluss auf bereits initiierte Instanzen
- Ablage des Input-Files *ab fehlerhaftem Record* unter: DOMAIN_HOME/rejmsgs/SERVER_NAME/COMPOSITE_NAME
- Ablage des *kompletten* fehlerhaften Files unter in `PhysicalErrorArchiveDirectory` definiertem Speicherort.

## 1.2 Standardverhalten bei abgebrochenen BPEL-Instanzen

Abbruch von BPEL Instanzen durch System oder Business Faults haben keinen Einfluss auf bereits gelaufene oder neue BPEL-Instanzen für den gleichen Batch. Dies bedeutet, es wird auch kein Transaktions-Rollback durchgeführt und neue Instanzen für den nachfolgenden Publish des nächsten Batches werden ebenfalls ordnungsgemäss durchgeführt.

Falls jedoch der `DeliveryMode` der BPEL-Komponente auf `sync` gesetzt wird, erfolgt ein-Abbruch nach dem Fehler, frühere Transaktionen bleiben jedoch bestehen. Da das File jedoch nicht gelöscht wird, werden wieder neue Instanzen mit den noch nicht gelesenen Records gestartet. Daher muss der `DeliveryMode` zwingend auf dem Standardwert `async.persist` belassen werden.

## 1.3 Implementierung

### 1.3.1 Polling Adapter für Streaming (File und FTP)

Bei der Definition des Service Adapters werden folgende Eigenschaften explizit definiert:

| | |
|---|---|
| Operation Type: | Read File |
| Use File Streaming : | yes |
| Files contain multiple Messages: | yes |
| Publish Messages in Batches of: | 10 – 1000 (frei konfigurierbar) |

Im folgende Darstellung zeigt ein Beispiel des JCA Adapterkonfigurations Artefakts (*.jca):

```
<adapter-config name="ReadFile_FixedLength" adapter="File Adapter"
                wsdlLocation="ReadFile_FixedLength.wsdl"

xmlns="http://platform.integration.oracle/blocks/adapter/fw/metadata">
  <connection-factory location="eis/FileAdapter" UIincludeWildcard="*.log"/>
  <endpoint-activation portType="Read_ptt" operation="Read">
    <activation-spec
className="oracle.tip.adapter.file.inbound.ScalableFileActivationSpec">
      <property name="DeleteFile" value="true"/>
      <property name="MinimumAge" value="0"/>
      <property name="PhysicalDirectory" value="/tmp/inbound"/>
      <property name="Recursive" value="true"/>
      <property name="PublishSize" value="100"/>
      <property name="PollingFrequency" value="30"/>
      <property name="PhysicalArchiveDirectory" value="/tmp/archive"/>
      <property name="IncludeFiles" value=".*\.log"/>
      <property name="UseHeaders" value="false"/>
      <property name="PhysicalErrorArchiveDirectory" value="/tmp/error"/>
```

```
    </activation-spec>
  </endpoint-activation>
</adapter-config>
```

Das Property `PhysicalErrorArchiveDirectory` dient zur Definition der Ablage für nicht vollständig verarbeitete Files und kann nicht über die graphische Benutzeroberfläche definiert werden.

#### *1..3.1.a.1 Verarbeitungsreihenfolge*

Insbesondere beim Import von Stammdaten sollte sichergestellt werden, dass die Files in der Reihenfolge des Eintreffens verarbeitet werden. Dies kann durch ein Property in der JCA-Adapter-Konfiguration sichergestellt werden.

```
  <endpoint-activation portType="Read_ptt" operation="Read">
    <activation-spec
className="oracle.tip.adapter.file.inbound.ScalableFileActivationSpec">
      <property name="ListSorter"

value="oracle.tip.adapter.file.inbound.listing.TimestampSorterAscending"/>
      <property name="SingleThreadModel" value="true"/>
    </activation-spec>
  </endpoint-activation>
```

Das `SingleThreadModel` stellt zudem sicher, dass nur jeweils ein File zur gleichen Zeit verarbeitet wird und schützt das System damit vor Überlastung.

### 1.3.2 Propagierung der JCA Adapter Properties von Mediator zu BPEL Prozessen

Gewisse Properties des JCA File-Adapters werden in den BPEL Komponenten zur Korrelation und Aktualisierung in der Batch Log Tabelle (BATCH_LOG) benötigt. Damit diese in den BPEL Prozessen zur Verfügung stehen, müssen sie vom Mediator an den BPEL Prozess übergeben und dort Variablen zugeordnet werden:

#### *1..3.2.a.1 Propagierung im Mediator:*

*MyMediator*.mplan Definition:

```
   <action>
      <transform>
         <part name="$out.body"
               function="xslt(xsl/InputFile_FixedLength_To_InputFile.xsl,
$in.body)"/>
      </transform>
      <assign>
         <copy target="$out.property.jca.file.Batch"
            value="$in.property.jca.file.Batch"/>
         <copy target="$out.property.jca.file.BatchIndex"
            value="$in.property.jca.file.BatchIndex"/>
         <copy target="$out.property.jca.file.FileName"
            value="$in.property.jca.file.FileName"/>
         <copy target="$out.property.jca.file.Size"
            value="$in.property.jca.file.Size"/>
         <copy target="$out.property.jca.file.Directory"
            value="$in.property.jca.file.Directory"/>
      </assign>
      <invoke reference="FileProcessing.fileprocessing_client"
              operation="FileProcessing"/>
   </action>
```

Diese Zuordnung kann auch über die graphische Benutzeroberfläche erfolgen.

#### *1..3.2.a.2 Zuordnung in BPEL Prozessen:*

*MyBPEL*.bpel Definition (bei ‚Receive' Activity):

```
    <receive name="receiveInput" partnerLink="fileprocessing_client"
             portType="ns1:FileProcessingPortType" operation="FileProcessing"
             variable="inputVariable" createInstance="yes">
      <bpelx:fromProperties>
        <bpelx:fromProperty name="jca.file.Batch" variable="BatchName"/>
        <bpelx:fromProperty name="jca.file.BatchIndex" variable="BatchIndex"/>
        <bpelx:fromProperty name="jca.file.FileName" variable="FileName"/>
        <bpelx:fromProperty name="jca.file.Directory" variable="DirectoryName"/>
        <bpelx:fromProperty name="jca.file.Size" variable="FileSize"/>
      </bpelx:fromProperties>
    </receive>
```

Die Zielvariablen müssen vorgängig definiert werden; die Zuordnung kann auch über die graphische Benutzeroberfläche erfolgen.

### 1.3.3 Composite Sensors für Suche nach Filenamen in EM

Damit in Oracle Fusion Middleware Control Instanzen nach dem verarbeiteten Filenamen gesucht werden können, kann dieser mittels eines Composite Sensors registriert werden:

Dies kann über den graphischen Designer in JDeveloper konfiguriert werden und/oder direkt über das Artefakt `sensor.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<sensors xmlns="http://xmlns.oracle.com/bpel/sensor">
    <sensor sensorName="FileName" kind="service" target="undefined" filter="">
        <serviceConfig service="ReadFile_XML"
expression="$in.property.jca.file.FileName" operation="Read" outputDataType="string"
outputNamespace="http://www.w3.org/2001/XMLSchema"/>
    </sensor>
    <sensor sensorName="FileName" kind="service" target="undefined" filter="">
        <serviceConfig service="ReadFile_FixedLength"
expression="$in.property.jca.file.FileName" operation="Read" outputDataType="string"
outputNamespace="http://www.w3.org/2001/XMLSchema"/>
    </sensor>
</sensors>
```

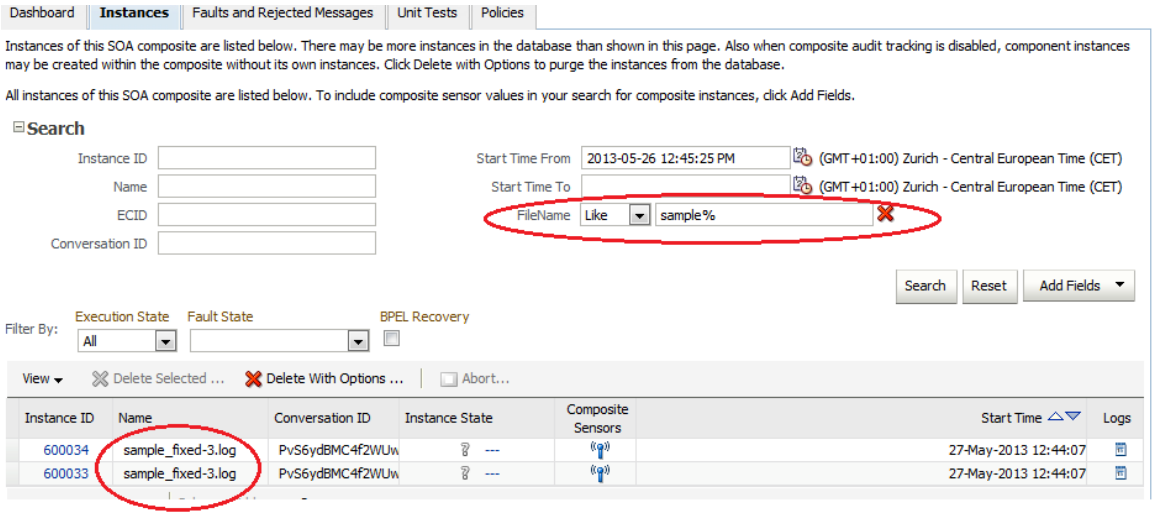
### 1.3.4 Anzeige des verarbeiteten Filenamens in EM Instanzübersicht

Der Filename kann bei Bedarf als Composite Instance Titel benutzt werden, damit dieser in der Instanzübersicht in Oracle Fusion Middleware Control gleich mit angezeigt wird.

Die Zuweisung erfolgt über eine Assign Aktion mit der entsprechenden Oracle Advanced Function:

```
<copy>
    <from>ora:setCompositeInstanceTitle($FileName)</from>
    <to>$CompositeInstTitle</to>
</copy>
```

Dabei ist zu beachten, dass der Aufruf der Funktion `CompositeInstanceTitle` bereits korrekt setzt, diese jedoch nur in einer Activity – wie bspw. Assign - wie oben abgebildet aufgerufen werden kann.


Dashboard
Instances
Faults and Rejected Messages
Unit Tests
Policies
Instances of this SOA composite are listed below. There may be more instances in the database than shown in this page. Also when composite audit tracking is disabled, component instances may be created within the composite without its own instances. Click Delete with Options to purge the instances from the database.
All instances of this SOA composite are listed below. To include composite sensor values in your search for composite instances, click Add Fields.
Search
Instance ID
Name
ECID
Conversation ID
Start Time From
2013-05-26 12:45:25 PM
(GMT+01:00) Zurich - Central European Time (CET)
Start Time To
(GMT+01:00) Zurich - Central European Time (CET)
FileName
Like
sample%
Search
Reset
Add Fields
Filter By:
Execution State
All
Fault State
BPEL Recovery
View
Delete Selected ...
Delete With Options ...
Abort...
Instance ID
Name
Conversation ID
Instance State
Composite Sensors
Start Time
Logs
600034
sample_fixed-3.log
PvS6ydBMC4f2WUw
---
27-May-2013 12:44:07
600033
sample_fixed-3.log
PvS6ydBMC4f2WUw
---
27-May-2013 12:44:07

### 1.3.5 Audit Level

Um zu verhindern dass unnötige Audit Trace Informationen geschrieben werden – und damit auch die Durchlaufzeiten zu optimieren –sollte der Audit Trace für Batch-Jobs komplett deaktiviert werden.

Dies kann mit dem entsprechenden Property in `composite.xml` auf der BPEL-Komponente gesetzt werden:

```
<component name="TargaImport" version="2.0">
  <implementation.bpel src="MyBPEL.bpel"/>
  <property name="bpel.config.auditLevel" type="xs:string" many="false"
            override="may">Off</property>
</component>
```

### 1.3.6 Stabilität und Durchlaufzeiten

Folgende Ergebnisse zeigen aufgrund einmaliger Messung einen voraussichtlichen Ausblick auf die zu erwartenden Durchlaufzeiten – in Relation zur gewählten Batch-Grösse (Publish-Size).

Die Tests beziehen sich auf das REFERENCE_FileAdapter Composite. Dieses enthält in der eigentlichen Verarbeitung (Loop der Hauptverarbeitung) mit Ausnahme einer Zählerinkrementierung keine weitere Aktionen. Nichtsdestotrotz lassen die Zahlen erste Rückschlüsse auf die optimal zu wählende Publish-Size zu.

Zeitangabeformat MIN'SEC"MSEC

| **Publish Size** | **File mit 10'000 Records** | **File mit 50'000 Records** | **File mit 100'000 Records** |
|---|---|---|---|
| 100 | 9"781 | 45"153 | 1'21"332 |
| 1'000 | 4"623 | 14"689 | 28"784 |
| 5'000 | 4"814 | 15"685 | 39"956 |
| 10'000 | 11"295 | 1'05"099 | Out-of-memory fault |

## 2. Pre- und Postprocessing Handler

Häufig bestehen spezifische Anforderungen zur Protokollierung der Verarbeitung der eingehenden Dateien. Insbesondere soll dabei aufgezeichnet werden:

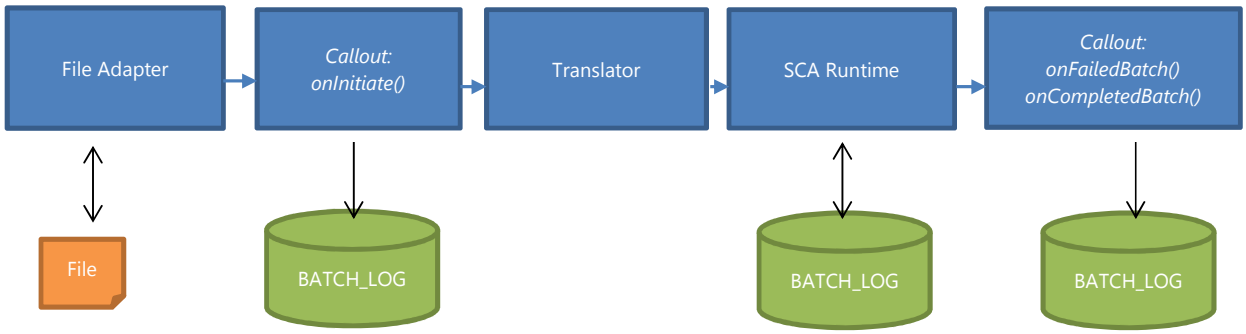
- Verarbeitungsbeginn und –Ende
- Anzahl erfolgreich (und fehlerhaft) verarbeiteter Records

Im vorliegenden Beispiel wird dabei die Tabelle BATCH_LOG verwendet. Die Protokollierung findet einerseits im Pre- und Postprocessing Handler des Adapterframeworks, und anderseits in den BPEL-Komponenten selber statt.

Für Pre- und Postprocessing stellt das Adapterframework zwei verschiedene Mechanismen zur Verfügung :

- Batch Notification Callback (Java Class) via `BatchNotificationCallout` Klasse für File Debatching
- Batch Notification Handler mit Pipeline und Valves via `BatchNotificationHandler` Interface

Da das vorliegenden Szenario keine Modifikation des eigentlichen Files oder File-Inhalt benötigt, wird zur Zeit nur der Batch Notification Callback verwendet. Dies kann bei Bedarf erweitert werden:



## 2.1 Korrelierung der SCA-Instanzen

Da aufgrund des beschriebenen Publish-Mechanismus typischerweise mehrere SCA Instanzen für das gleiche Input-File initiiert werden, müssen die einzelnen Instanzen zu einem logischen Verarbeitungslauf pro File korreliert werden. Dazu können die JCA Laufzeit Properties `jca.file.Batch` und `jca.file.BatchIndex` verwendet werden.

Dabei wird eine Unique ID pro Verarbeitungslauf (physisches File) vergeben und einen Index, der die aktuell ausgeführte Instanz kennzeichnet.

**Achtung**: *Wird zweimal das gleiche File unverändert in das zu überwachende Verzeichnis transferiert, wird die gleiche ID wie bei beim vorherigen Verarbeitungslauf für dieses File vergeben. Dieses Verhalten kann zurzeit nicht geändert werden wird im vorgesehenen Prozess aber abgehandelt indem eine bereits vorhandene ID als doppelte Verarbeitung eingestuft und zurückgewiesen wird*.

## 2.2 Batch Log Tabelle in der Datenbank

Um die Aktivitäten in einer Batchverarbeitung sinnvoll aufzeichnen zu können, kann ein Logging-Mechanismus eingesetzt werden, der beispielsweise entsprechende Logging-Tabellen in der Datenbank schreibt. In einer Entität `BATCH_LOG` werden die einzelnen Verarbeitungen eines Files zur Nachvollziehbarkeit und Versand eines allenfalls benötigten Verarbeitungsprotokolls aufgezeichnet:

Der Ablauf und die Protokollierung ist dabei wie folgt:

1) **BatchNotificationCallout** : *onInitiateBatch()* erzeugt Record mit jca.file.batch als Unique Primary Key. Ist bereits ein Eintrag unter dieser ID vorhanden wird ein entsprechendes Flag gesetzt, welches die folgende Verarbeitung blockiert.
2) **SCA-Instanz (Beginn)**: Bestehender Eintrag für die aktuelle ID wird überprüft. Ist der Batchlauf blockiert (infolge doppelter ID) wird die Verarbeitung ohne Fehler abgebrochen
3) **SCA-Instanz (Beginn):** Bei der ersten Instanz für den entsprechenden Batch-Lauf (`jca.file.BatchIndex = 1`) wird Batch Log aktualisiert mit Start-Datum und weiteren Attributen.
4) **SCA-Instanz (Ende):** Aktualisierung des Batch-Logs mit der Anzahl erfolgreich und fehlerhaft verarbeiteten Records. Ebenso wird letztes Composite Verarbeitungsdatum protokolliert und Anzahl ausgeführter Instanzen summiert.
5) **SCA-Instanz(Ende):** Überprüfung ob File-Verarbeitung und Anzahl Total gesendeter Batch-Jobs (aus NotificationCallout) übereinstimmt mit der Anzahl der effektiv ausgeführten. Falls ja wird Benachrichtigungsprozess ausgelöst.
6) **BatchNotificationCallout:** *onFailedBatch()* und *onCompletedBatch()* setzen End-Datum der Fileverarbeitung, Anzahl gesendeter Batch-Jobs und Flag über erfolgreiche Verarbeitung.

## 2.3 Implementierung im SCA Composite

Wie im Kapitel 1 beschrieben werden die eigentlichen Verarbeitungsprozesse als BPEL Komponenten implementiert und über den JCA File-Adapter via Mediator instanziiert. Die nachfolgende Beschreibung bezieht sich ausschliesslich auf die Implementierung der BPEL Komponente.

### 2.3.1 Verarbeitungsablauf der BPEL Komponente

**1) Pre-Processing**

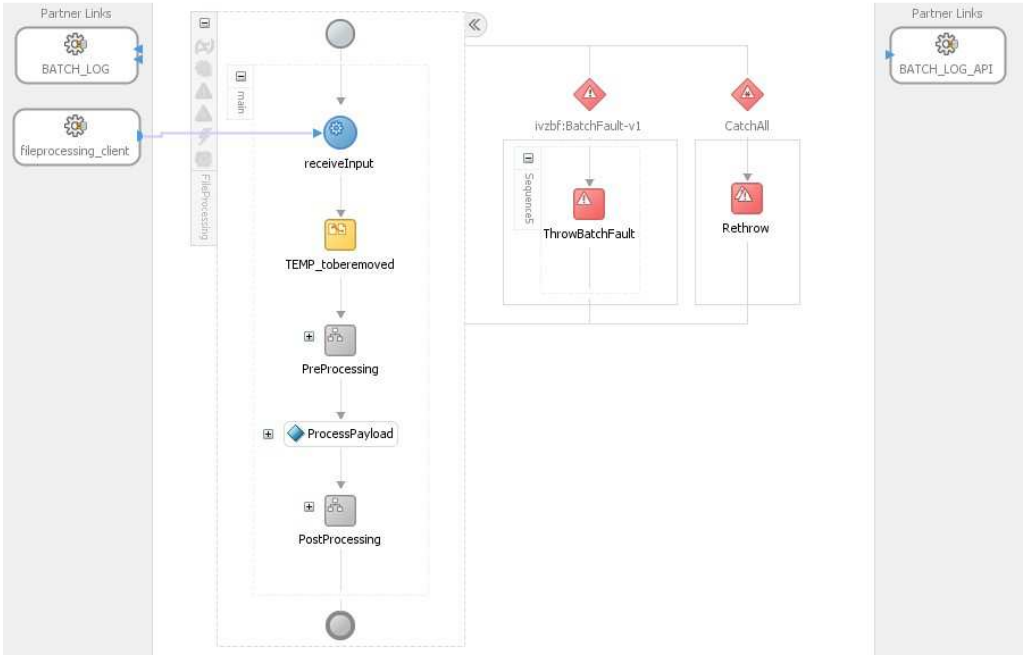
- Lesen der Informationen aus `BATCH_LOG` und Verarbeitungsende falls Batch gelockt ist (gemäss Kapitel 2.2)
  - Abbruch der Verarbeitung mit Fehler `samplebf:BatchFault-v1` falls kein Eintrag in `BATCH_LOG` vorhanden ist. Diese Instanzen erscheinen als abgebrochen im Enterprise Manager (Business Fault)
- Aktualisierung `BATCH_LOG` bei erster Instanz gemäss Kapitel 2.2

**2) Main-Processing**

- Record-Verarbeitung in Iteration über Input-Payload
- Inkrementierung Counter für Anzahl erfolgreiche/fehlerhafte Record

**3) Post-Processing**

- Aktualisierung `BATCH_LOG` Entität über PL/SQL
- Erkennung Verarbeitungsende gemäss Kapitel 2.2 und Auslösen des Benachrichtigungs-Prozesses



### 2.3.1 Konfiguration des BatchNotificationCallout

Damit die BatchNotificationCallout Klassen für Pre- und Postprocessing aufgerufen werden, müssen diese auf der JCA Service Komponente definiert werden.

Dies erfolgt in der Composite Definition composite.xml wie folgt :

```
  <service name="ReadFile_FixedLength"
           ui:wsdlLocation="ReadFile_FixedLength.wsdl">
    <interface.wsdl
interface="http://xmlns.oracle.com/pcbpel/adapter/file/SAMPLE_TEST/REFERENCE_FileAdap
ter/ReadFile_FixedLength#wsdl.interface(Read_ptt)"/>
    <binding.jca config="ReadFile_FixedLength_file.jca"/>
    <property name="batchNotificationHandler" type="xs:string" many="false"
override="may">java://
ch.admin.trivadis.sample.soa.batch.control.ControlServiceBatchNotificationCallout</pr
operty>
  </service>
```

### 2.3.2 Einbindung des BatchFault WSDL

Damit in JDeveloper die BatchFaults über das graphische Design-Utility ausgelöst und abgefangen werden können, muss das entsprechende Fault WSDL in das entsprechende SCA Projekt importiert werden:
`oramds:/apps/sample/wsdl/fault/batch/v1/BatchFaults-v1.0.wsdl`

- BPEL Komponente (*.bpel)
- WSDL der BPEL Komponente

### 2.3.3 Transaktionsverhalten

Standardmässig werden alle Interaktionen mit Referenzen (z.B. DBAdapter) in einer globalen Transaktion ausgeführt. Diese Transaktion wird commited, wenn ein expliziter Dehydration-Point (z.B. async Web-Service Aufruf, Wait, Pick oder Receive Activity) erreicht wird oder nach erfolgreichem Abschluss der kompletten Instanz (abhängig der eingestellten Transactionproperties für die Komponenten). Dies kann bedeuten, dass im Falle eines unbehandelten Fehlers im Ablauf auch die gesamte Transaktion bis zum letzten Checkpoint zurückgerollt wird.

Die effizienteste Methode um dies zu verhindern, ist die Konfiguration einer dedizierten Datasource auf Weblogic Ebene welche nicht an der globalen Transaktion partizipiert (Non-XA) und diese für die Persistierung der Log-Einträge zu verwenden.

#### *2..3.3.a.1 Alternativ über Idempotent Property*

Grundsätzlich kann auch das `idempotent` Property verwendet werden um die Transaktion während des Prozesses zu einem bestimmten Zeitpunkt zu committen. Allerdings muss dabei beachtet werden, dass dabei die gesamte, bisher ausgeführte globale Transaktion committed wird was unter Umständen unerwünschte Nebeneffekte und Performance-Einbussen mit sich bringen kann.

**SOA Suite PS5**

```
<property name="partnerLink.BatchLog.idempotent" type="xs:string"
          many="false">false</property>
```

**SOA Suite PS6**

*(ab Version 11.1.1.7 kann das Property auf Operationsebene gesetzt werden):*

```
<property name="bpel.partnerLink.BatchLog.update.idempotent"
             type="xs:string" many="false">false</property>
```

Das gleiche Vorgehen könnte auch angewendet werden, um beispielsweise allfällige Transaktionen in der Iteration der Hauptverarbeitung sofort und unabhängig von der globalen Transaktion zu committen.

## 2.4 Findings und Issues

### 2.4.1 Fehlerhafte Anzahl gesendete Batch-Jobs

Für die Anzahl gesendeter Batch-Jobs wird das bei den Methoden `onFailedBatch()` und `onCompletedBatch()` zur Verfügung stehende Argument `finalBatchSize` verwendet.

Dieses ist im aktuellen Release 11.1.1.6.4 immer um 1 zu hoch. Aus diesem Grund wird zur Zeit hardcodiert der Wert `finalBatchSize -1` verwendet zur Protokollierung in BATCH_LOG. (Falls gleichzeitig ein BatchNotificationHandler über Pipeline Konfiguration eingesetzt wird, ist der wert korrekt – sic!).

*Dieses Fehlverhalten muss in Version 11.1.1.7 nochmals getestet werden und ggf. in BatchNotificationCallout modifiziert werden.*

### 2.4.2 Infrastruktur: benötigte Patches für Oracle SOA Suite 11.1.1.6 (PS5)

Für Version 11.1.1.6 der SOA Suite muss ein Patch eingespielt werden, andernfalls werden alle Batchläufe bei Verwendung der BatchNotificationHandler als fehlerhaft gekennzeichnet. In Version 11.1.1.7 (PS6) ist der Fehler behoben.

**Patch 13562653: DEBATCHING FLOW FAILS WHEN USING BATCHNOTIFICATIONHANDLER**

## 3. Fehlerbehandlung mit Wiederaufsetzpunkgen via Fault Policies

Sobald eine SCA Instanz vom JCA Adapter initiiert worden ist, liegt es in der Verantwortung des gestarteten SCA, dass diese auch ordnungsgemäss beendet wird. Der JCA-Adapter, der die Instanzen startet, hat darüber keine Kontrolle.

Um eine vollständige Verarbeitung gewährleisten zu können – beispielsweise auch im Falle eines temporären Systemfehlers auf einer Enterprise Ressource oder Webservice Endpoint – wird das Fault-Policy Framework der SOA-Suite eingesetzt. Dieses erlaubt die Konfiguration von Retry-Versuchen im Falle eines Fehlers auf der aufgerufenen Referenz und Definition von Fehler-Aktionen bei erfolgloser Anzahl Neuversuche.

Folgendes Verhalten wird typischerweise gemäss jeweiligen Anforderungen definiert:

- Für RemoteFaults wird eine umgebungsspezifische Anzahl von Neuversuchen (Retries) per Fault Policy konfiguriert
- Falls die Neuversuche nicht erfolgreich waren, wird als Default-Aktion für den Prozess ‚Human Intervention' ausgelöst. Dies bedeutet, dass die betroffenen Prozessinstanzen, *ab der entsprechenden erfolglosen Aktion i*m Prozess später manuell über den Enterprise Manager wiederaufgesetzt werden können.
- Alle anderen System Faults werden mit ‚Rethrow' an die Prozessinstanz zurückgegeben, worauf diese mit Fehler abbricht.
- Die Fault-Policy Artefakte werden umgebungsspezifisch mit entsprechendem Postfix im Dateinamen im MDS abgelegt

## 3.1 Implementierung im SCA Composite

Die vordefinierten Fault Policies werden in der Composite Definition `composite.xml` registriert:

```
  </service>
  <property
name="oracle.composite.faultPolicyFile">oramds:/apps/sample/policy/batchjob_faultpoli
cy_DEV.xml</property>
  <property
name="oracle.composite.faultBindingFile">oramds:/apps/sample/policy/batchjob_faultbin
ding_DEV.xml</property>
  <component name="….>
```

Die JCA Adapter Komponenten in der SOA Suite haben per default ein Retry-Mechanismus definiert, welcher nicht auf ‚0' gesetzt werden kann. Um redundante Retries zu vermeiden, müssen die entsprechenden JCA-Binding Properties auf den Referenzen in `composite.xml` komplett entfernt werden:

```
  <reference name="BATCH_LOG" ui:wsdlLocation="BATCH_LOG.wsdl">
    <interface.wsdl
interface="http://xmlns.oracle.com/pcbpel/adapter/db/SAMPLE_TEST/REFERENCE_FileAdapte
r/BATCH_LOG#wsdl.interface(BATCH_LOG_ptt)"/>
    <binding.jca config="BATCH_LOG_db.jca"/>
<!–
    <property name="jca.retry.count" type="xs:int" many="false"
override="may">1</property>
    <property name="jca.retry.interval" type="xs:int" many="false"
              override="may">1</property>
    <property name="jca.retry.backoff" type="xs:int" many="false"
              override="may">2</property>
    <property name="jca.retry.maxInterval" type="xs:string" many="false"
              override="may">120</property>
-->
  </reference>
```

### 3.1.1 Umbgebungsspezfische Anpassung der Fault Policy

Die Fault Policies sind typischerweise in den verschiedenen Umgebungen (Dev, Test, Prod) unterschiedlich, insbesondere die verwendeten Retry Werte können unterschiedlich sein. Um dies zu erreichen, müssen die entsprechend zu verwendenden Fault-Policies in den jeweiligen Configuration-Plans des Composites angepasst werden:

```
  </service>
  <property name="oracle.composite.faultPolicyFile">
     <replace>faultpolicy/batchjob_faultpolicy.xml</replace>
  </property>
  <property name="oracle.composite.faultBindingFile">
     <replace>faultpolicy/batchjob_faultbinding.xml</replace>
  </property>
```

### 3.1.2 Besonderheit beim Deployment der Fault Policy

Wenn eine Fault-Policy geändert wird, ist ein alleiniger Deploy des MDS Projektes nicht ausreichend. Um die Modifikationen zu aktivieren, müssen die importierenden SCA Projekte ebenfalls neu deployed werden.

## Dokumentation


| | |
|---|---|
| [ORA- E10231-12]. | Oracle File and FTP Adapter Concepts<br>**Oracle® Fusion Middleware User's Guide for Technology Adapters**<br>Oracle Corp.<br>http://docs.oracle.com/cd/E23943_01/integration.1111/e10231/adptr_file.htm#TKADP321 |
| | |
| | |