

HERZLICH
WILLKOMMEN

Effektives Fault Handling mit Oracle SOA Suite 11g

Guido Schmutz
Technology Manager

16.11.2011

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

1

2011 © Trivadis

Oracle SOA Suite- Gartner Analysts Report
16.11.2011

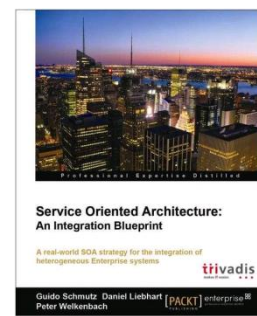
trivadis
makes IT easier. ■ ■ ■

Guido Schmutz

- Working for Trivadis for more than 14 years
- Oracle ACE Director for Fusion Middleware and SOA
- Co-Author of different books
- Consultant, Trainer Software Architect for Java, Oracle, SOA and EDA
- Member of Trivadis Architecture Board
- Technology Manager @ Trivadis

- More than 20 years of software development experience

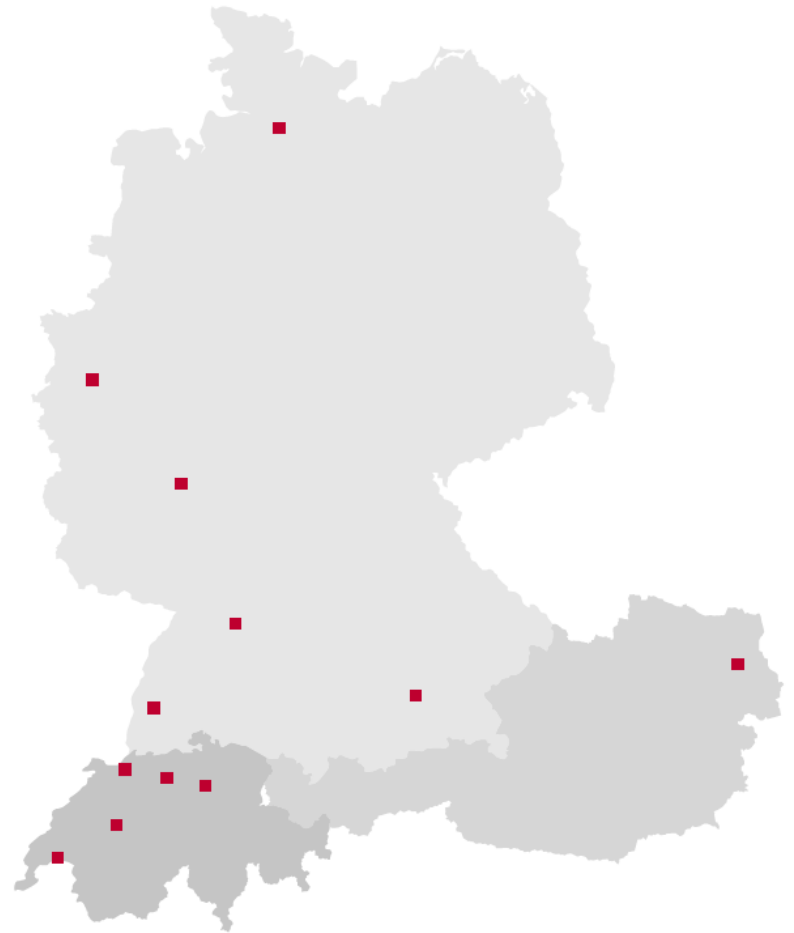
- Contact: guido.schmutz@trivadis.com
- Blog: <http://guidoschmutz.wordpress.com>
- Twitter: [gschmutz](https://twitter.com/gschmutz)



2011 © Trivadis

Trivadis Facts & Figures

- 11 Trivadis locations with more than 550 employees
- Financially independent and sustainably profitable
- Key figures 2010
 - Revenue CHF 101 / EUR 73 mio.
 - Services for more than 700 clients in over 1'800 projects
 - Over 170 Service Level Agreements
 - More than 5'000 training participants
 - Research and development budget: CHF 5.0 / EUR 3.6 mio



Agenda

- 1. What is Fault Handling**
2. Fault Handling in SOA vs. traditional systems
3. Scenario and Patterns
4. Implementation of Scenario
5. Summary and Best Practices

Error/Fault Handling

- The goal of every programmer should be to write unbreakable software
- depends on how good expected and unexpected exception conditions are handled and managed
- Object-oriented languages (C++, Java, C#) provide efficient way for handling exceptions
 - constructs such as `try`, `catch`, and `finally`.
- With SOA
 - most of what is available at language level is still valid and usable
 - different challenges will appear, once starting orchestrating services and creating composite applications
- Prevention vs. handling

What is a Fault ?

- Something outside normal operational activity or “happy flow” happened
 - Technical error
 - Programming error
 - Faulty operation by user
 - Exceptional business behavior



Two Types of Faults

Business faults

- Faults that service clients can **expect** and **recover** from
- Failure to meet a particular business requirement
- Often: expected, business value, contractual and recoverable

Technical faults

- Faults that service clients do **not expect** and **cannot (easily) recover** from
- Results of unexpected errors during runtime, e.g. null pointer errors, resources not available, and so on
- Often: unexpected, technical, implementation and non-recoverable



Business Fault

```
<wsdl:operation name="orderProduct">
  <wsdl:input message="order:OrderProductRequestMessage"/>
  <wsdl:output message="order:OrderProductResponseMessage"/>
  <wsdl:fault message="order:ProductNotInStockFaultMessage"
    name="ProductNotInStockFault"/>
  <wsdl:fault message="order:CustomerNotFoundFaultMessage"
    name="CustomerNotFoundFault"/>
</wsdl:operation>
```

1. Service contract including fault

```
<xsd:element name="CustomerNotFoundFaultMessage">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="CustName" type="xsd:string"/>
      <xsd:element name="City" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2. Fault message payload



Business Fault (II)

```
<soap:Envelope>
  <soap:Header/>
  <soap:Body>
    <soap:Fault>
      <faultcode>CST-1234</faultcode>
      <faultstring>Customer not found</faultstring>
      <detail>
        <CustomerNotFoundFault>
          <CustName>John Doe</CustName>
          <City>Long Beach</City>
        </CustomerNotFoundFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

3. Actual service response



Technical Fault

```
<wsdl:operation name="orderProduct">
  <wsdl:input message="order:OrderProductMessage"/>
  <wsdl:output message="order:OrderProductResponseMessage"/>
  <wsdl:fault message="order:ProductNotInStockFaultMessage"
    name="ProductNotInStockFault"/>
  <wsdl:fault message="order:CustomerNotFoundFaultMessage"
    name="CustomerNotFoundFault"/>
</wsdl:operation>
```

1. Service contract including fault

2. Actual service response

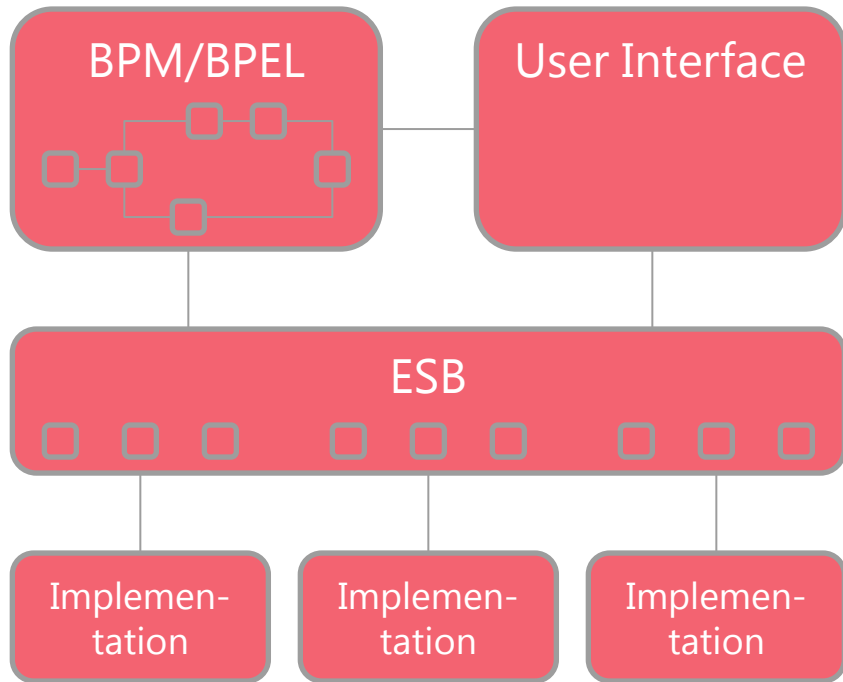
```
<soap:Envelope>
  <soap:Header/>
  <soap:Body>
    <soap:Fault>
      <faultcode>S:Server</faultcode>
      <faultstring>Could not connect to URL 127.0.0.1 on port 8001</faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```



Agenda

1. What is Fault Handling
- 2. Fault Handling in SOA vs. traditional systems**
3. Scenario and Patterns
4. Implementation of Scenario
5. Summary and Best Practices

Fault Handling SOA vs. traditional



Higher degree of complexity

Multiple service consumers

Services part of larger unit

Heterogeneous & external components

Long running processes

Asynchronous

Timed events

Often enterprise-wide

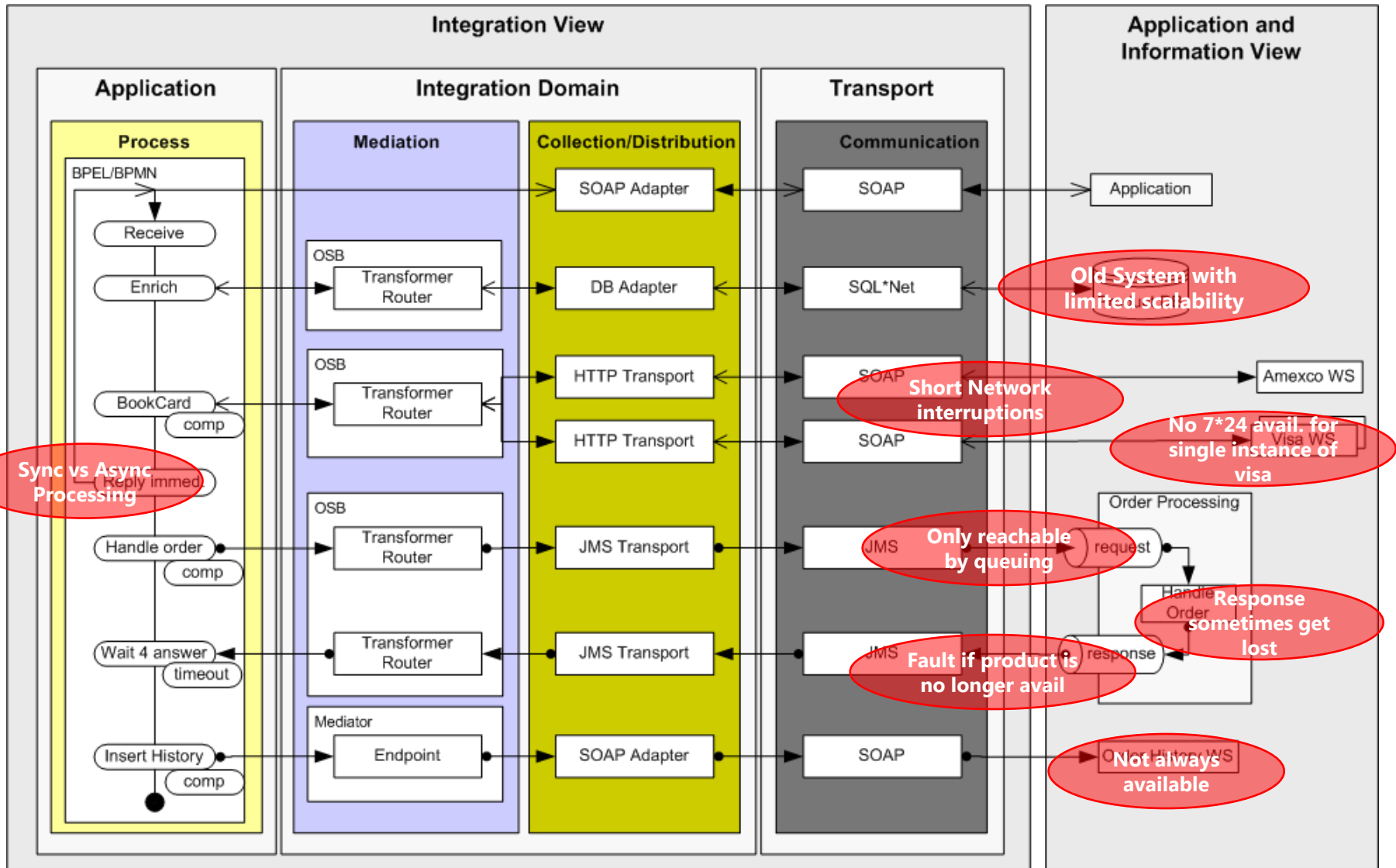
Transactions



Agenda

1. What is Fault Handling
2. Fault Handling in SOA vs. traditional systems
- 3. Scenario and Patterns**
4. Implementation of Scenario
5. Summary and Best Practices

Scenario (Areas of Problems/Faults)



Patterns for Fault Tolerant Software

Compensation

Exception shielding

(Limit) Retry

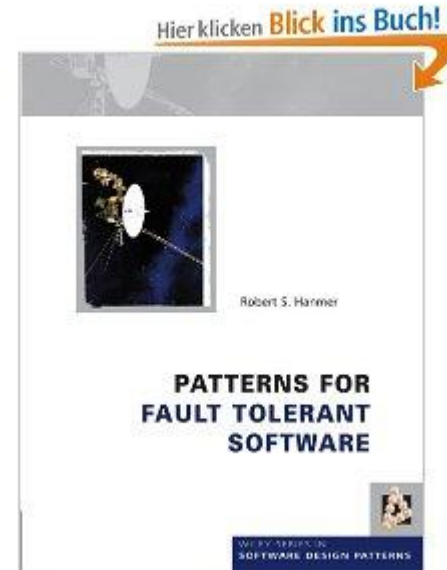
Share the load

Alternative

Exception handler

Heartbeat

Throttling

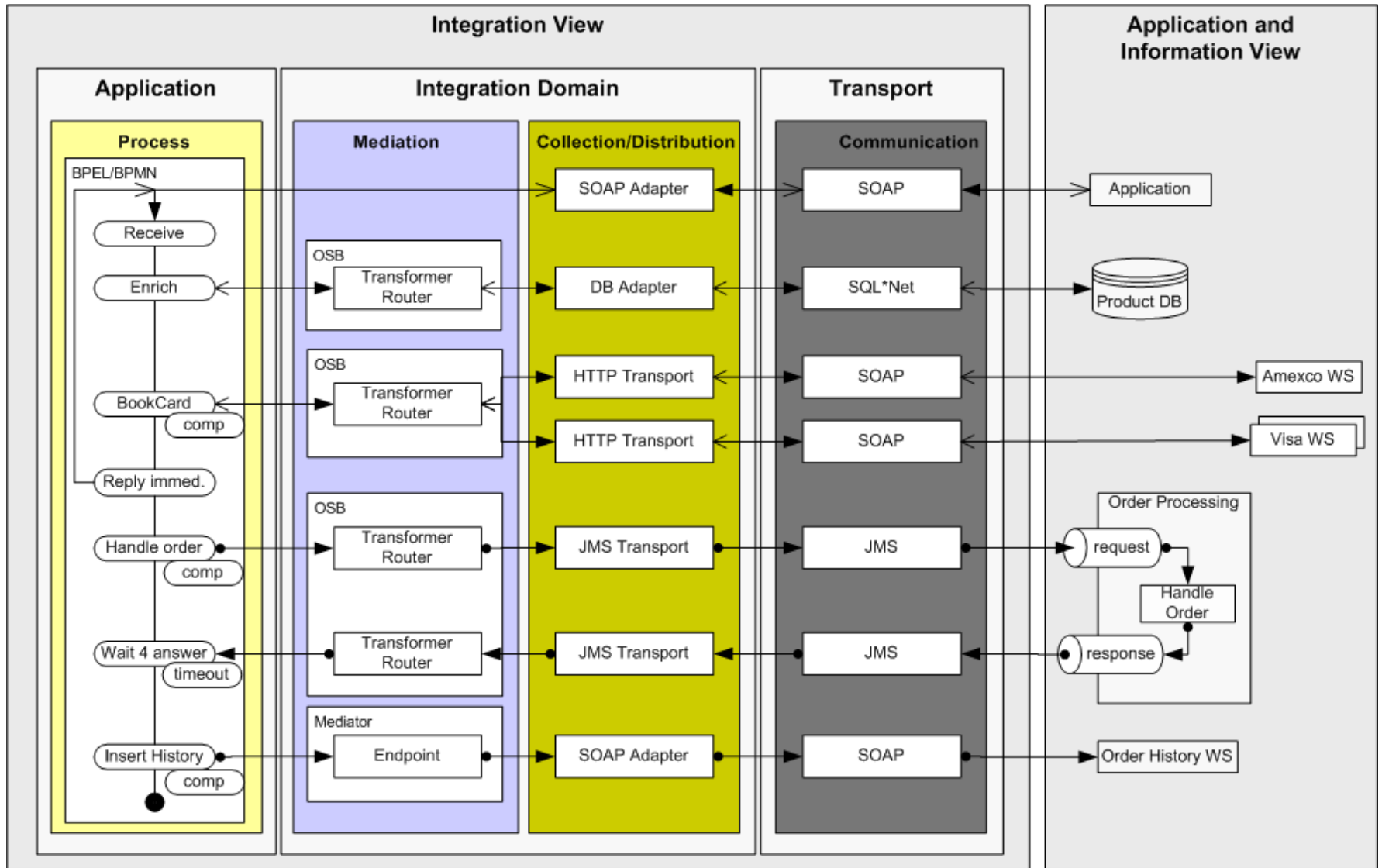


2011 © Trivadis

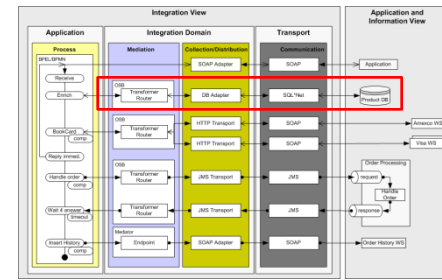
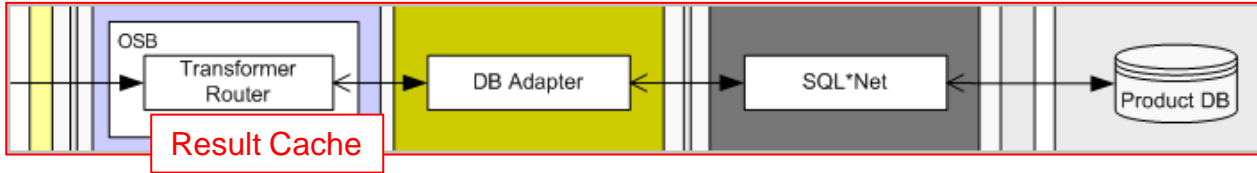
Agenda

1. What is Fault Handling
2. Fault Handling in SOA vs. traditional systems
3. Scenario and Patterns
- 4. Implementation of Scenario**
5. Summary and Best Practices

Implementation of Scenario



Product Management Result Caching



Problem

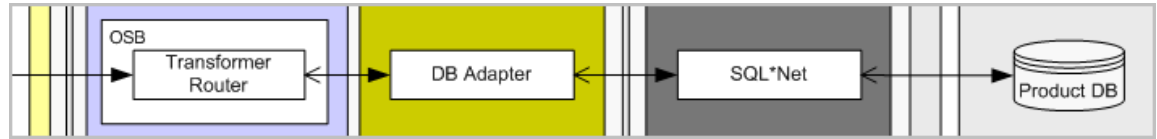
- Not to overload the old, non-scalable product system with the new demand

Solution

- Use **Result Caching** to cache the product information (read-only operation)
- Use **Service Throttling** to limit the number of concurrent requests

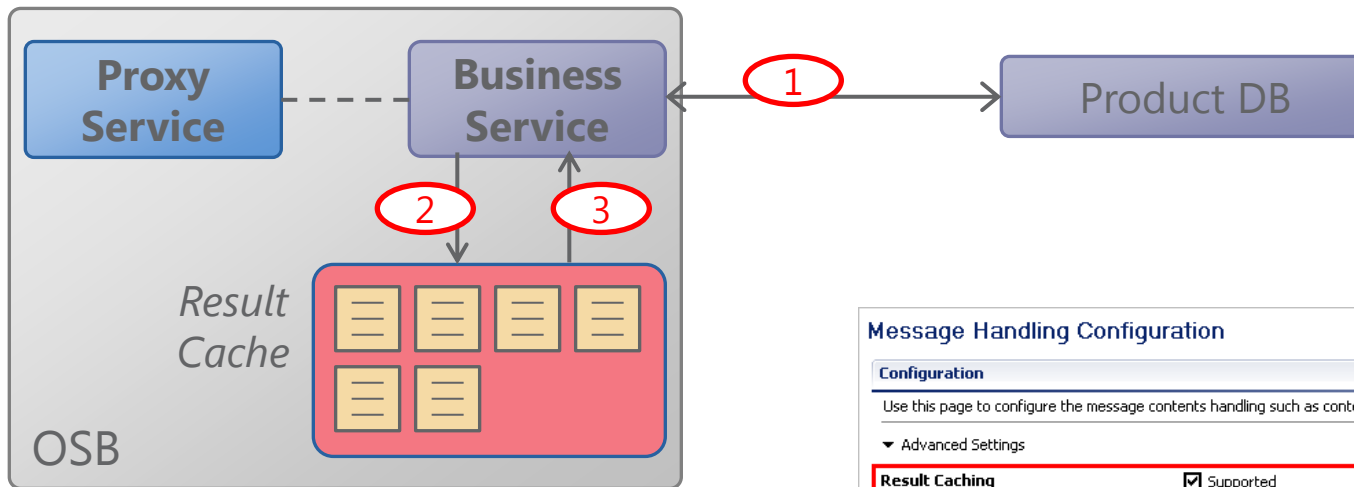


Product Management Result Caching



Results are returned from cache rather than invoking always the external service

- Product data is rather static, so ideal candidate for caching



Message Handling Configuration

Configuration

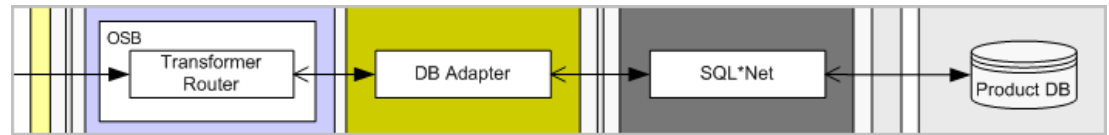
Use this page to configure the message contents handling such as content streaming, XOP/MTOM etc.

▼ Advanced Settings

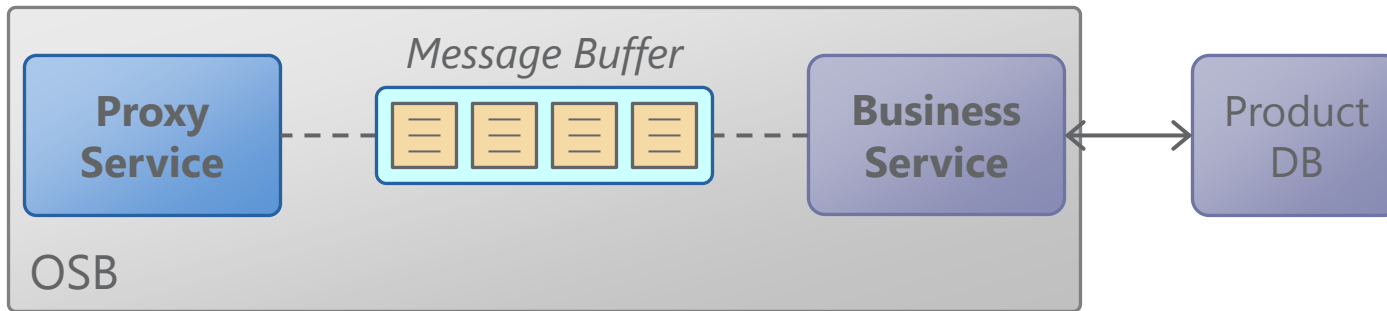
Result Caching	<input checked="" type="checkbox"/> Supported
Cache Token Expression	<code>\$body/prod:ProductDBServiceSelect_productCodeInputParameters/prod:productCode</code>
Expiration Time	<input type="radio"/> Use Default
	<input checked="" type="radio"/> Duration <input type="text" value="0"/> Days <input type="text" value="1"/> Hours <input type="text" value="1"/> : <input type="text" value="0"/> min : sec
	<input checked="" type="radio"/> XQuery Expression <input type="text" value="Request"/> <input type="button" value="v"/>
	<Expression>



Product Management Service Throttling



restricts the number of messages on the message flow to a business service

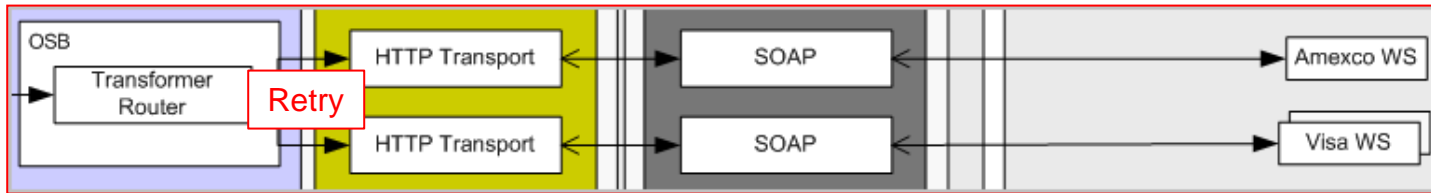
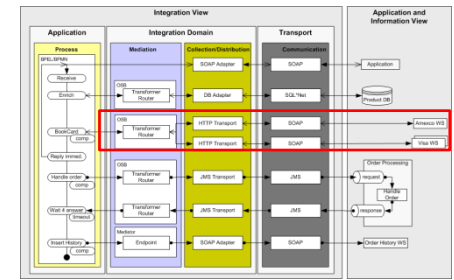


- Set from **Operational Settings** on the OSB console

Throttling	
Throttling State	<input checked="" type="checkbox"/> Enabled
Maximum Concurrency*	<input type="text" value="5"/>
Throttling Queue*	<input type="text" value="100"/> messages
Message Expiration	<input type="text" value="0"/> msecs



Credit Card Booking Retry Configuration



Problem

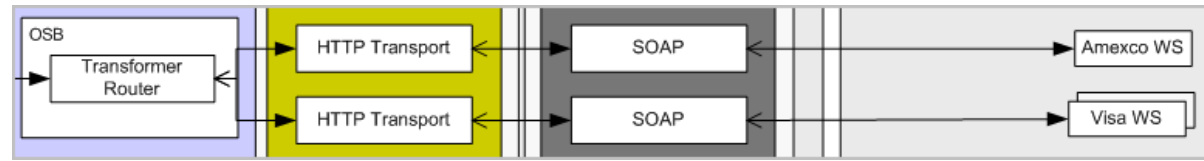
- Unstable network between us and the external services

Solution

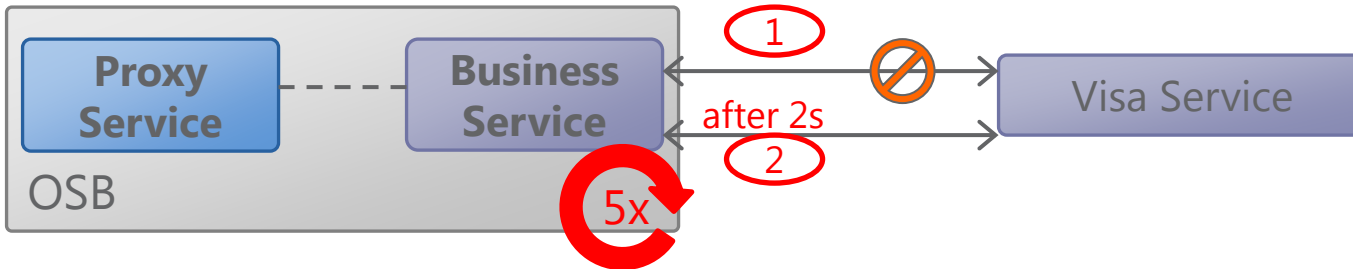
- Use **Retry** mechanism of OSB to try many times
- No Fault Management necessary for service consumer if network interruption is only for a short time



Credit Card Booking Retry Configuration



Configured on the business service in OSB



Transport Configuration

Use this page to configure the transport information for this service.

Protocol*

Load Balancing Algorithm

Endpoint URI* Format: `http://host:port/someService`

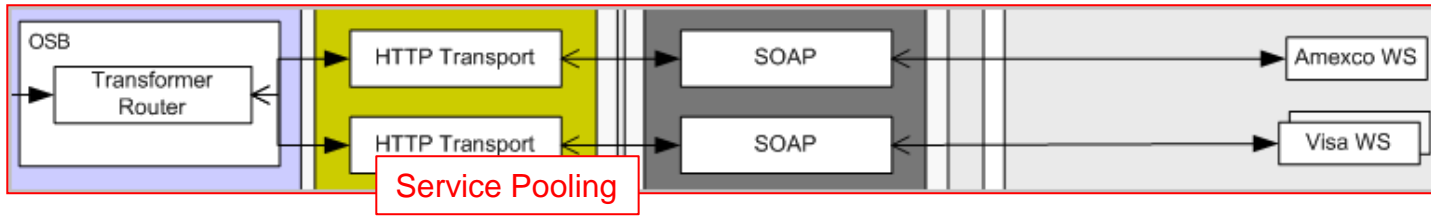
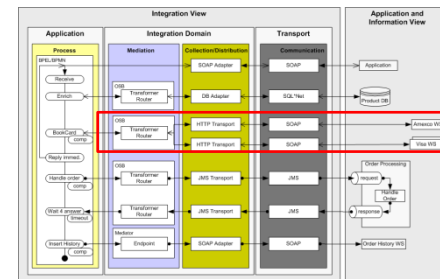
Existing URIs	
<code>http://localhost:8092/visa-card-ws-v3/v3</code>	<input type="button" value="Up"/>
	<input type="button" value="Down"/>
	<input type="button" value="Delete"/>

Retry Count

Retry Iteration Interval

Retry Application Errors Yes No

Credit Card Booking Service Pooling



Problem

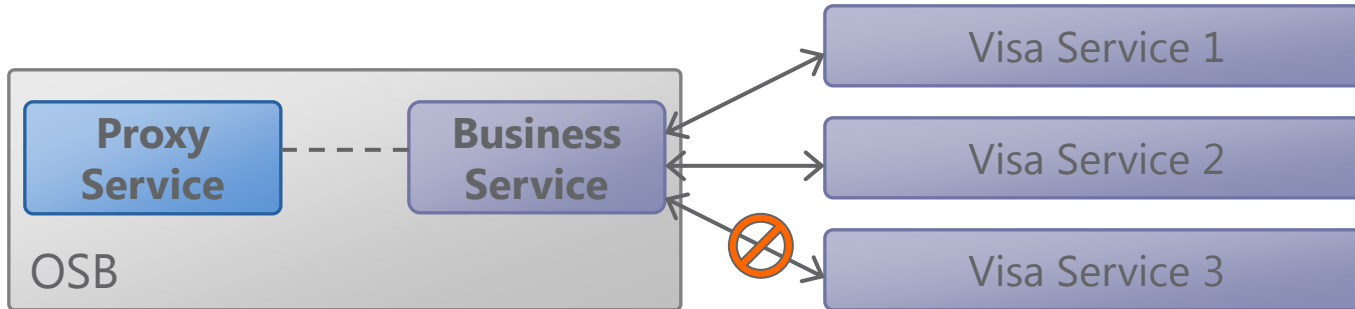
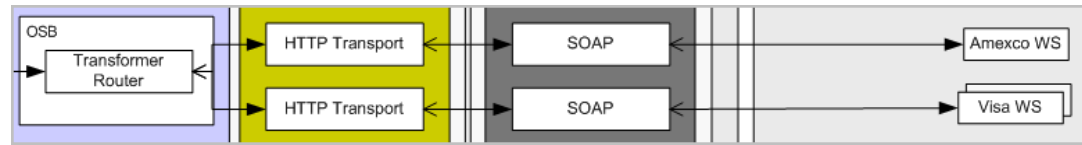
- Visa Service does not guarantee 7*24 availability for one single instance

Solution

- Use the multiple instances (endpoints) visa provides and use service pooling feature of OSB
- No Fault Management for the service consumer if at least one endpoint is available



Credit Card Booking Service Pooling



Transport Configuration

Configuration

Use this page to configure the transport information for this service.

Protocol *

Load Balancing Algorithm

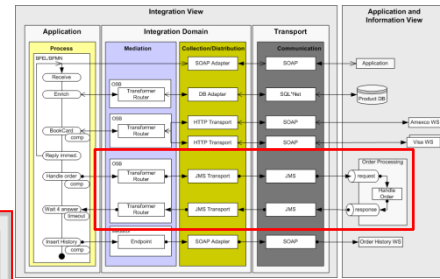
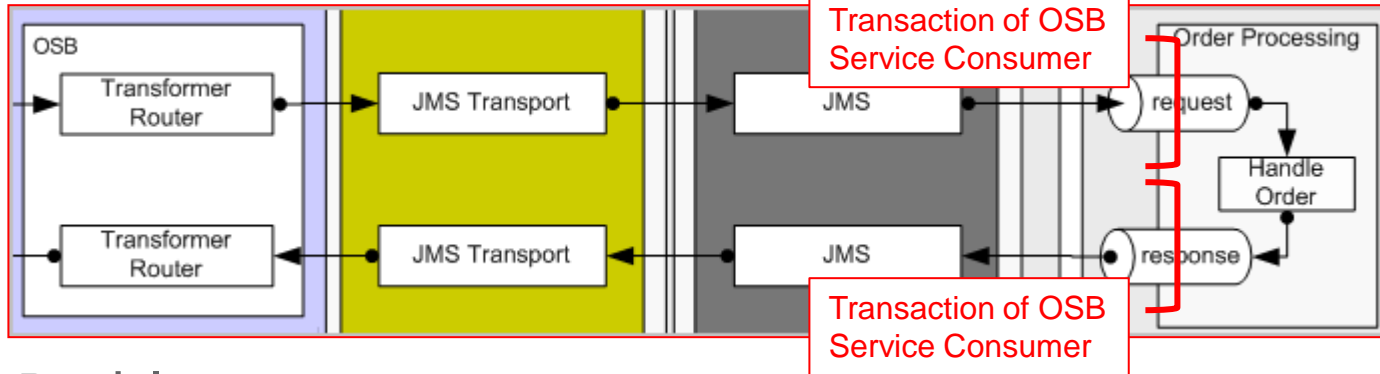
Endpoint URI * Format: http://host:port/someService

Existing URIs

- http://localhost:8092/visa-card-ws-v3/v3
- http://localhost:8093/visa-card-ws-v3/v3



Order Management Transaction configuration



Problem

- Guarantee that the message will be delivered to the order management system

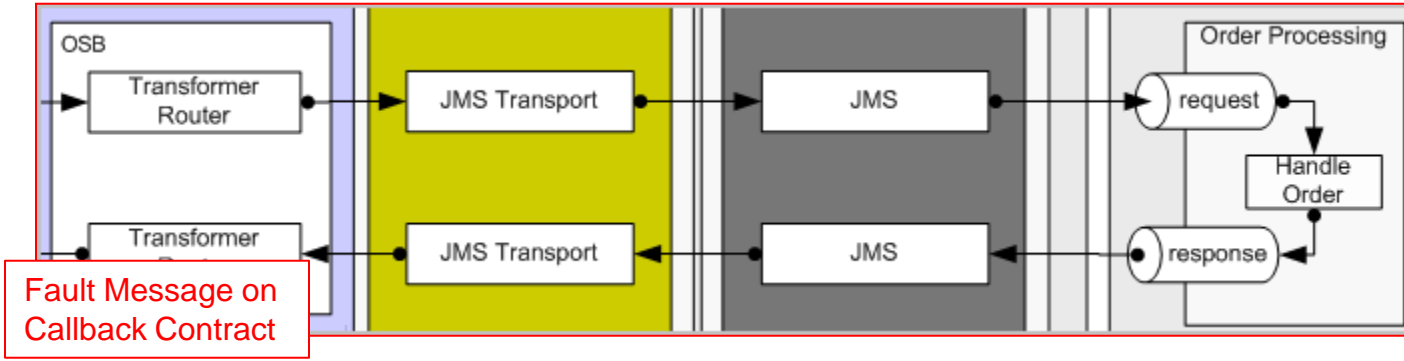
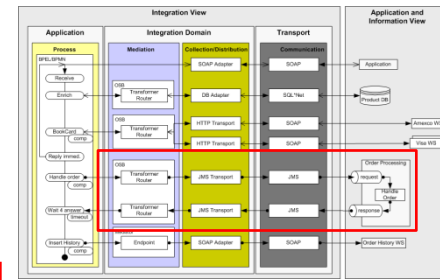
Solution

- Make sure that queues are available, even if the Handle Order system is not available, messages can still be sent
- Make sure that queuing runs within the same transaction as the message producer (BPEL => ESB)



Order Management (II)

Fault Message on Callback Contract



Problem

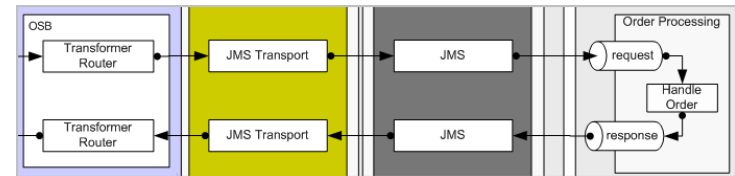
- Need to return a "Product No Longer Available" Business Fault over an Asynchronous MEP

Solution

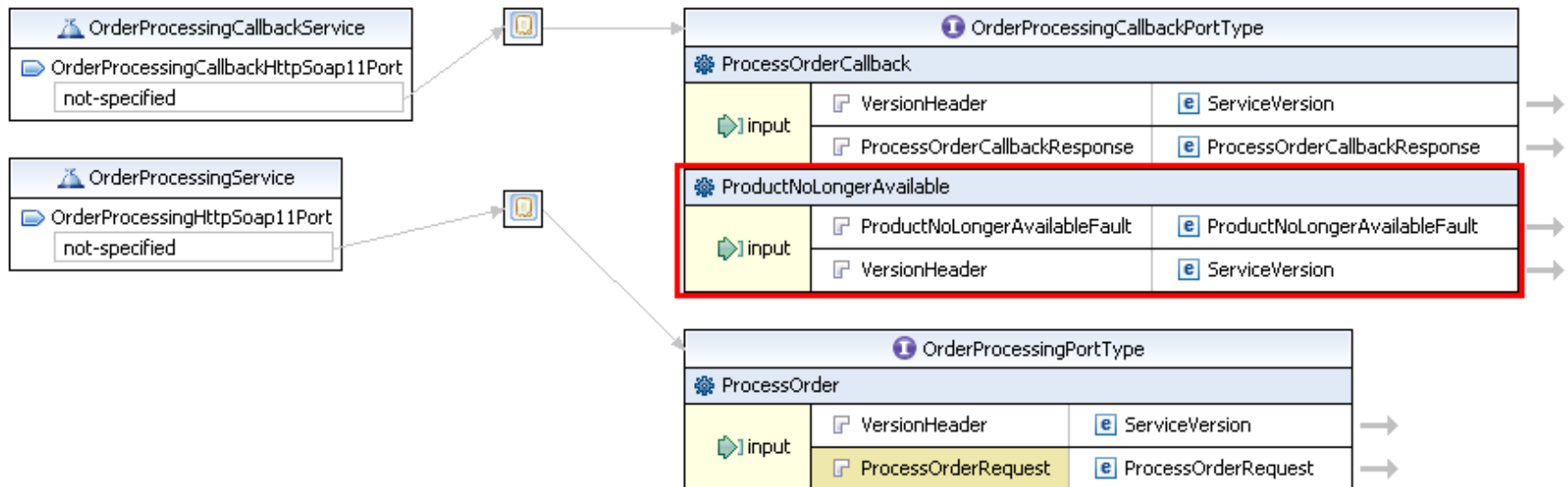
- Design a separate Fault Message and Operation on the Callback contract (WSDL) and use that

Order Management (II)

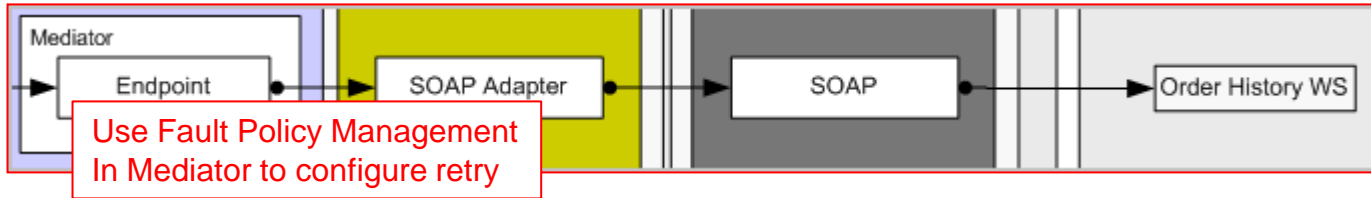
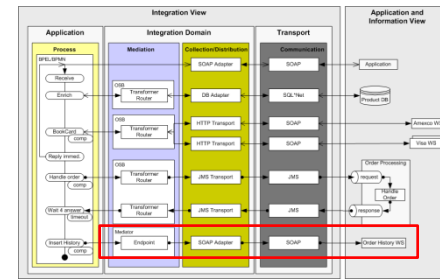
Fault Message on Callback Contract



“Business Fault” modeled as another operation on the Callback WSDL



Order History Fault Management Framework



Problem

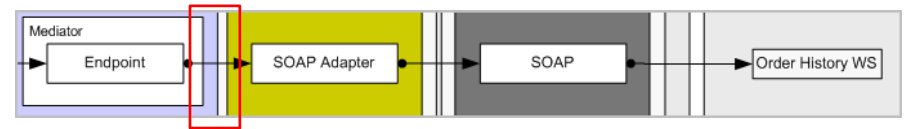
- Order History System not available should have no impact on Business Process

Solution

- Use Mediator (ESB) with Fault Management Framework to configure retry

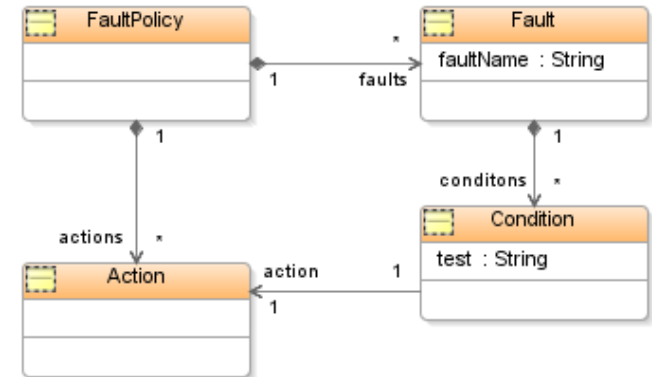


Order History Fault Management Framework



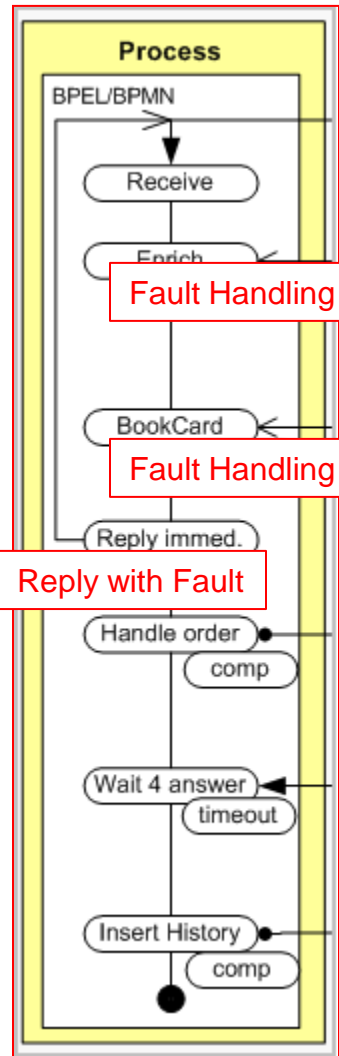
```
<faultPolicyBindings version="2.0.1"
  <composite faultPolicy="OrderProcessingFaults"/>
</faultPolicyBindings>
```

```
<faultPolicies>
  <faultPolicy version="2.0.1"
    id="OrderProcessingFaults"...>
    <Conditions>
      <action ref="ora-retry"/>
    </Conditions>
    <Actions>
      <Action id="ora-retry">
        <Retry>
          <retryCount>3</retryCount>
          <retryInterval>2</retryInterval>
          <exponentialBackoff/>
          <retryFailureAction ref="ora-java"/>
          <retrySuccessAction ref="ora-java"/>
        </Retry>
      </Action>
    </Actions>
  </faultPolicy>
</faultPolicies>
```



Order Handling Process

Return Errors as synchronous response

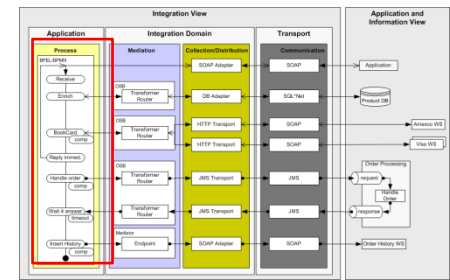


Problem

- Both Product Management and Credit Card Booking can return Business Faults

Solution

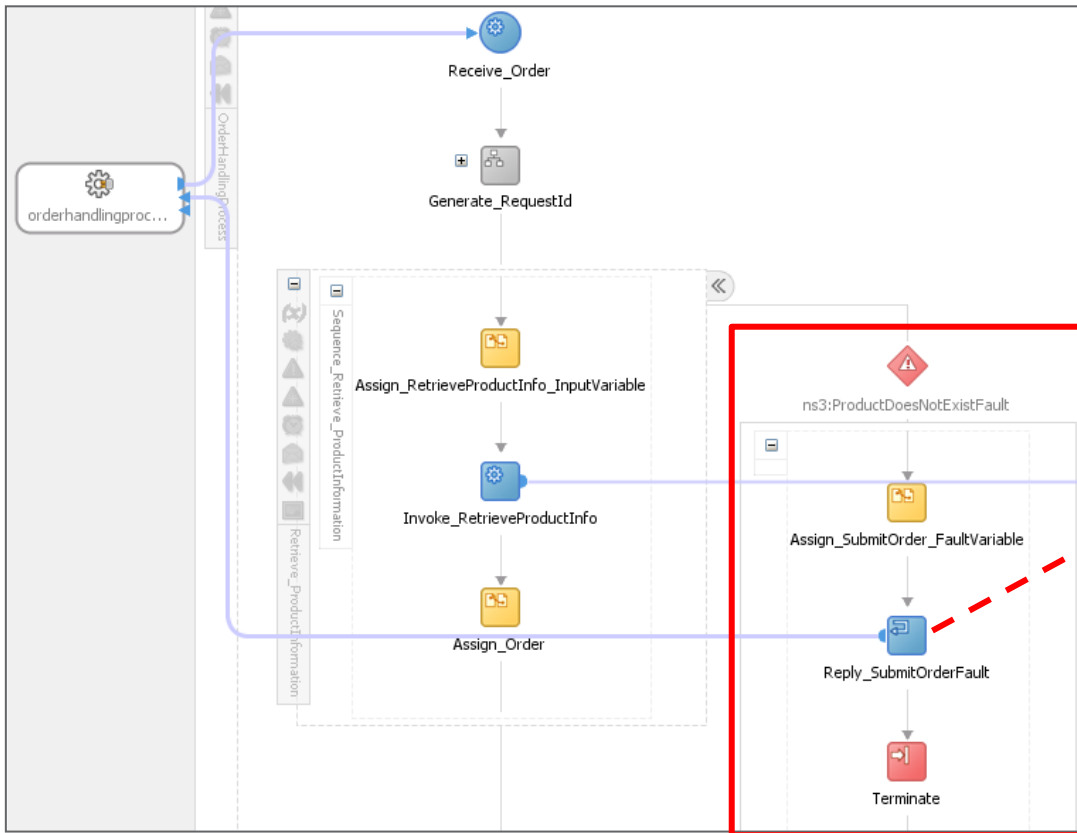
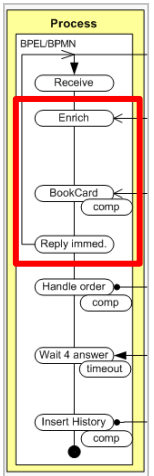
- Handle errors and map them to errors returned to the service consumer (i.e. the caller of the process)
- After that process changes from synchronous to asynchronous



Order Handling Process

Return Errors as synchronous response

Handle Business Faults in BPEL error handler and reply with an error



Edit Reply

Assertions Skip Condition Targets Sources Headers
General Correlations Properties Annotations

Name: Reply_SubmitOrderFault

Interaction Type: Partner Link

My Role WebService Interface

Partner Link: orderhandlingprocess_client

Operation: SubmitOrder

Variable: SubmitOrder_FaultVariable

Fault QName

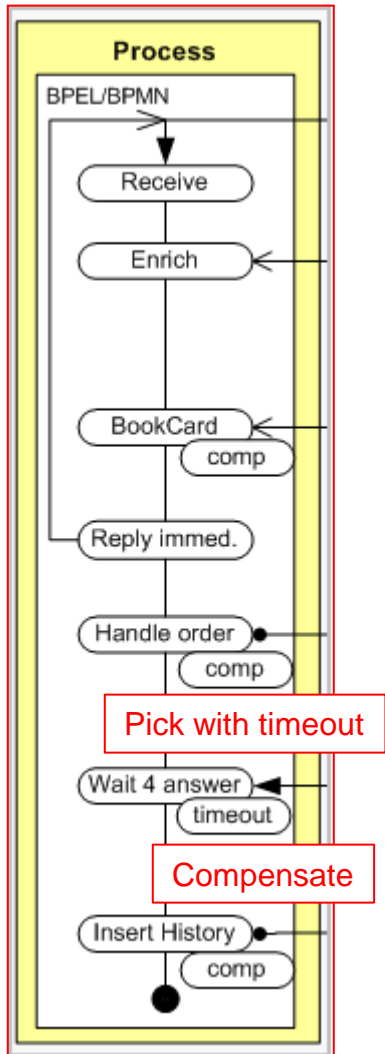
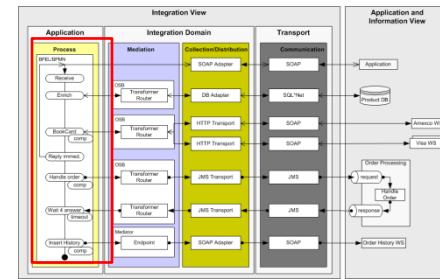
Namespace URI: http://somecompany.com/contract/OrderHandlingProcess/v1

Local Part: SubmitOrderFault

Help Apply OK Cancel

Order Handling Process (II)

Handle missing callback message with timeout



Problem

- Order Processing Response Message can get lost in the Order Processing system, i.e. the callback message will never arrive in BPEL

Solution

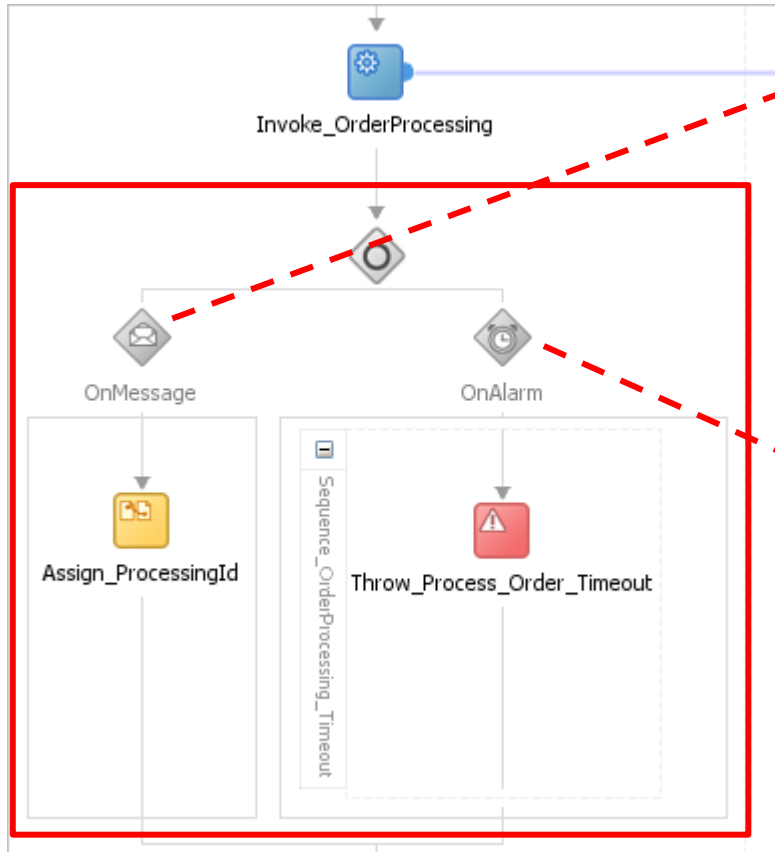
- Timeout on the Wait For Answer with a BPEL pick activity with a timeout
- Undo the process by doing compensation
- Use the BPEL compensate activity together with compensation handler to undo the Booking of the Credit Card



Order Handling Process (II)

Handle missing callback message with timeout

Pick Activity for handling callback message with timeout branch

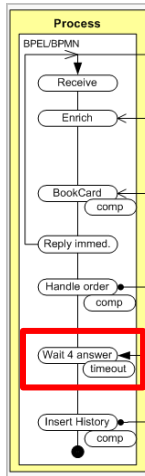


The 'Edit OnMessage' dialog box is shown with the following configuration:

- General tab selected.
- Conversation ID: [Empty field]
- Interaction Type: Partner Link
- Partner Link: OrderProcessingService
- Operation: ProcessOrderCallback
- Variable: IMessage_ProcessOrderCallback_InputVariable

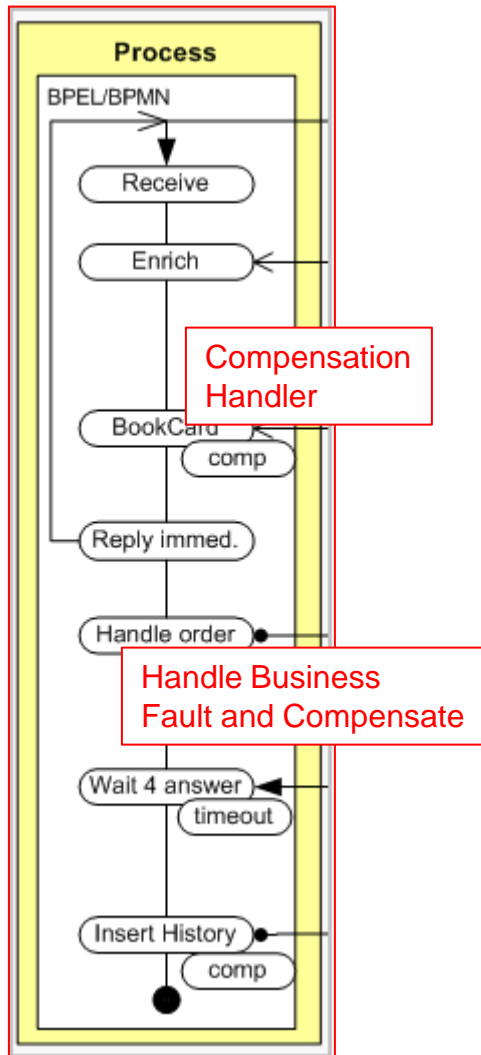
The 'Edit OnAlarm' dialog box is shown with the following configuration:

- General tab selected.
- For radio button selected.
- Time: 0 Yrs 0 Mons 0 Days 0 Hrs 2 Mins 0 Secs



Order Handling Process (III)

Compensation Handling

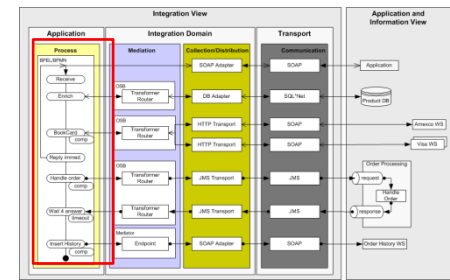


Problem

- Order Processing Callback Message can be a Product No Longer Available Business Fault

Solution

- Undo the process by doing compensation
- Use the BPEL compensate activity together with compensation handler to undo the Booking of the Credit Card

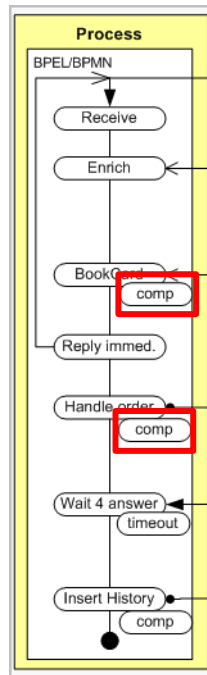
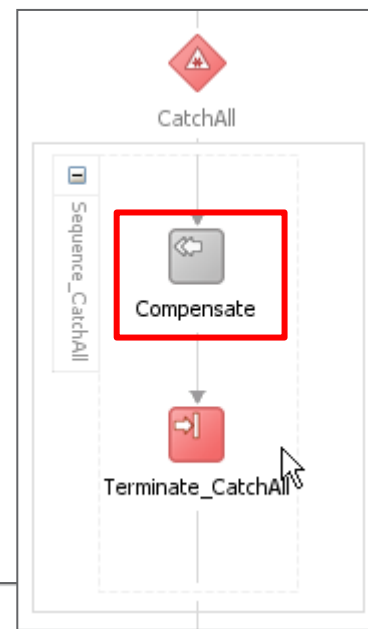
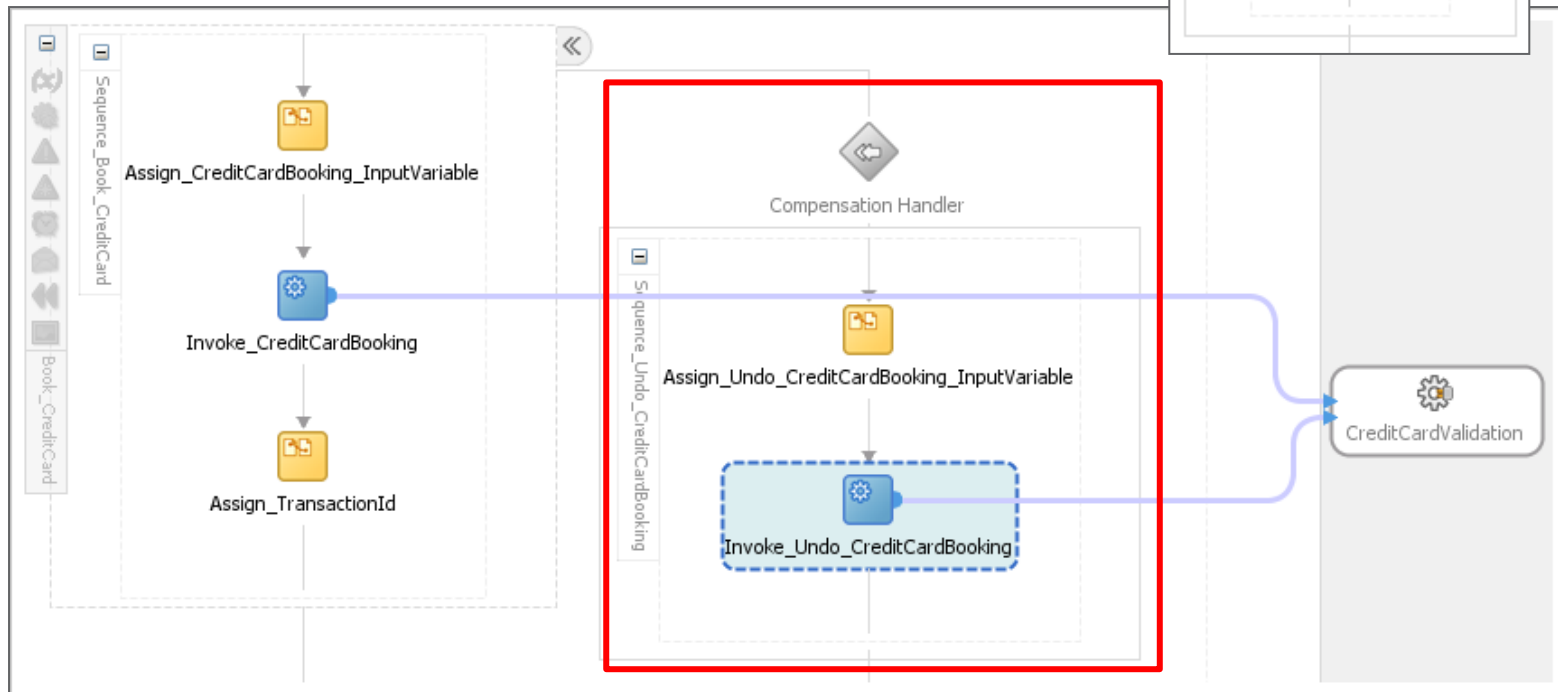


Order Handling Process (III)

Compensation Handling

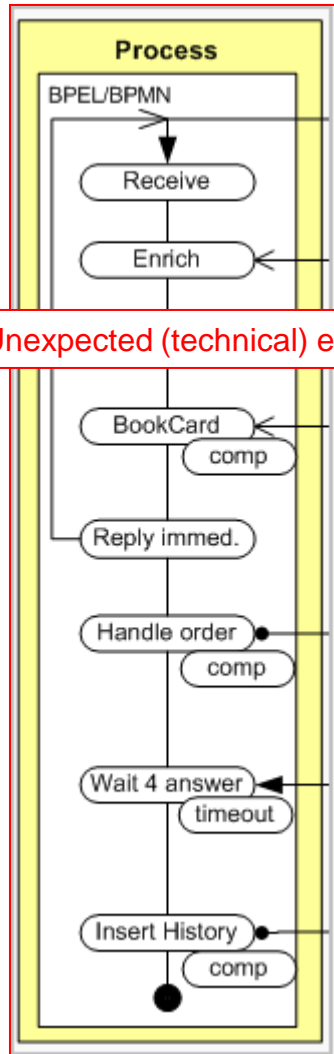
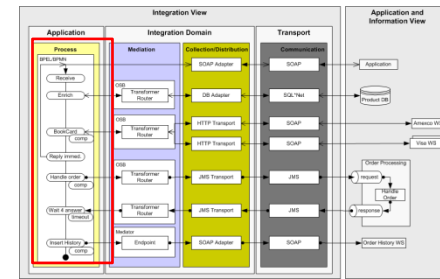
Compensate activity invokes compensation handling on the inner scope

- Can only be invoked from within a fault handler or another compensation handler



Order Handling Process (V)

Generic Error Handler w. Fault Policy Framework



Problem

- Unexpected (technical) fault
- Multiple processes deal with unexpected faults in their own way

Solution

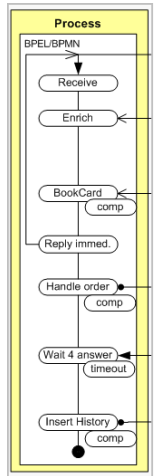
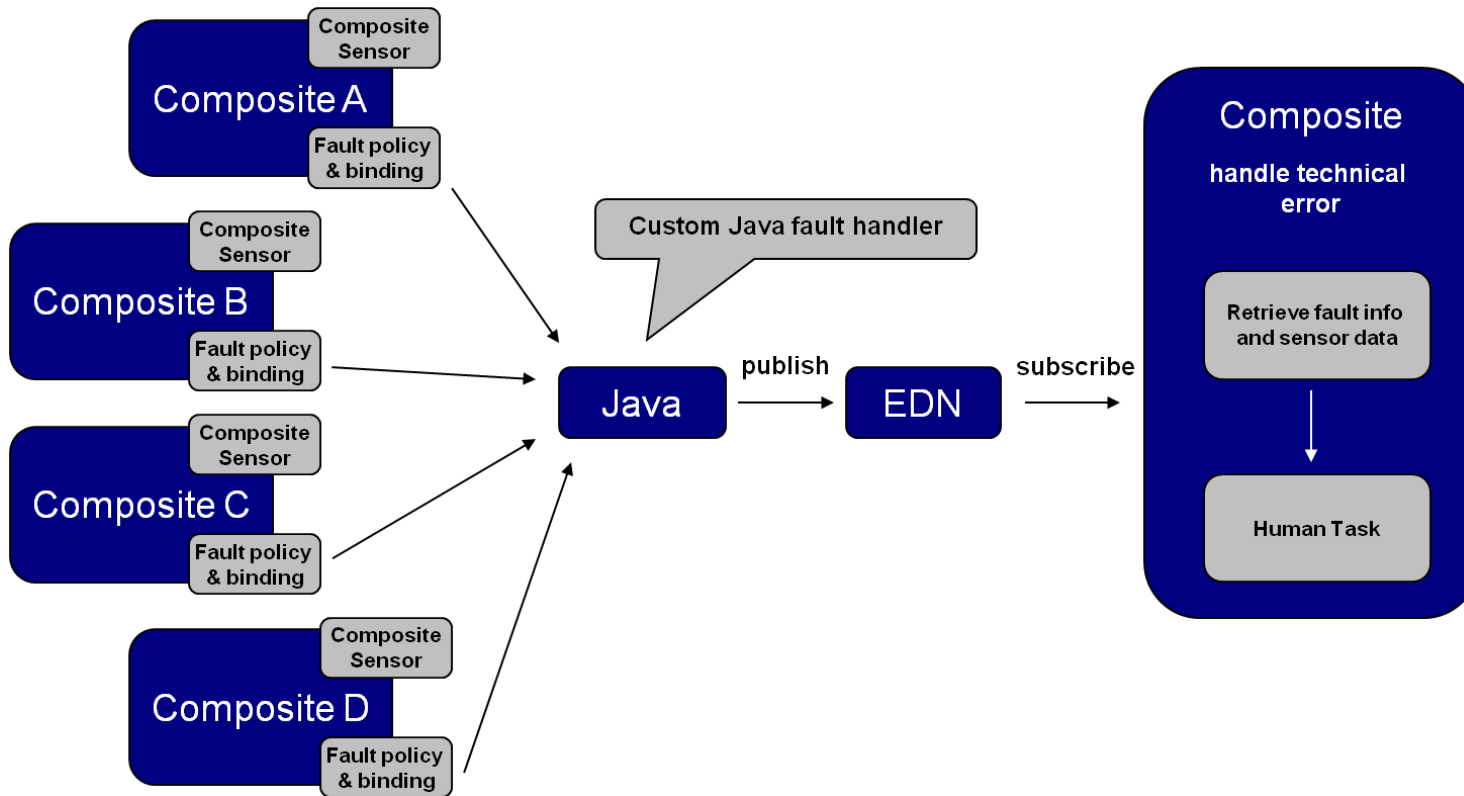
- Use fault handler mechanism to send error to error queue without adding process logic
- Create one process to listen to error queue and handle faults
- Retrieve process information by using (composite) sensors



Order Handling Process (V)

Generic Error Handler w. Fault Policy Framework

Explanation and demo of generic fault handler



Agenda

1. What is Fault Handling
2. Fault Handling in SOA vs. traditional systems
3. Scenario and Patterns
4. Implementation of Scenario
- 5. Summary and Best Practices**

Summary

Issue	Solution	Product
Overloading product management system	ThrottlingResult cache	OSB
Visa Service does not guarantee 7*24 uptime due to e.g. network problems	Multiple endpoints Service pooling	OSB
Guarantee message delivery to order management system	Availability of queues Enqueue and dequeue in service consumer transaction	OSB (and SOA Suite for XA propagation to OSB)
Returning business fault over async MEP from order management system	Separate operation and fault message	OSB and SOA Suite (callback contract between the two)
Order history service not available	Retry in Mediator using fault policy framework	SOA Suite
Business fault handling from service to process to consumer	Catch faults in process and reply fault to consumer	OSB and SOA Suite (correct contracts)
Detect missing response message	Timeout in pick activity	SOA Suite
Handle product no longer available	Compensation	SOA Suite
Avoid calling credit card booking twice	Set non-idempotent property	SOA Suite
Processes needing to deal with unexpected technical faults. All processes solving it in their own way using process logic.	Fault policy frameworks, error queue, generic error handler & (composite sensors.	SOA Suite



Summary & Best Practices

Differentiate between business and technical faults

Design service contracts with faults in mind: formally describe business faults in service contracts

Design with criticality, likeliness to fail, and cost in mind

Differentiate fault patterns in OSB and BPM/BPEL:

- OSB: Retry, Throttling, transactions
- BPM/BPEL: Compensation, business fault handling, generic fault handler, timeout

Handle unexpected errors generically

Make services autonomous

Fault-handling on scope of services and in wider perspective



VIELEN DANK.

Trivadis AG

Guido Schmutz

Europa-Strasse 5
CH-8152 Glattbrugg

Tel. +41-44-808 70 20
Fax +41-44-808 70 21

info@trivadis.com
www.trivadis.com

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN