OpenESB Newsletter
Edition 3, volume 1 - July 2013

The best integration environment

Editorial

Confirmation: OpenESB Version 2.3 is a great success! The download rate is high and there are thousands of downloads every month. Thank you for your support and your trust. Through the OpenESB forums, you can send us your feedback, bugs and fixes. This month, we will talk about a new OpenESB platform for JBoss and JVM, which are new OpenESB distributions designed, developed and proposed by the community. These versions, outside the Glassfish Application Server, are a great opportunity to develop the OpenESB market for new projects and to target new users.

We also present the IEP component (Intelligent Event Process), which was reintroduced in OpenESB 2.3. This month's tips and tricks have been written by **Gerhard Kersten** from **Acando**. Gerhard teaches us how the XSLT features of BPEL can be useful in replacing BPEL Mapper as well as proposing three XSLT examples, which are must know features in OpenESB. Please share your OpenESB skills and send us your OpenESB tips. We would also like to thank **LogiCoy,** our sponsor this month. LogiCoy is a large part of the community effort to make "The best open source integration environment on the market". We are eager to read other comments so feel free to contact us at contact@open-esb.net.

New platforms for OpenESB

For the last few months, a huge effort has been made to adapt OpenESB onto new platforms. The current targets are JBoss 5, JBoss 7, Tomcat and a standalone JVM. These will be the first OpenESB distributions outside the Glassfish application server. They allow JBoss, Tomcat and pure Java fans to run OpenESB as you would in Glassfish AS. OpenESB Standalone Edition will run perfectly on the cloud in an agile multi instance mode. It starts in less than 3 seconds and the core uses only 200 mb of memory. We are working on Netbeans plugin for these platforms as well to allow you to develop, design, debug and deploy OpenESB applications from Netbeans. If you are familiar with OpenESB on Glassfish, working on these new platforms will not change your habits.

However, some features strongly linked with the Glassfish application server are not present on these distributions:

- Java EE SE is not implemented in OpenESB JBoss and standalone versions. Java EE SEis strongly tied with Glassfish technologies and does not run on other applications servers.
- Glassfish console cannot be used to monitor and manage your application on other platforms. We are developing an agnostic console to replace GF admin console
- Clustering defined by Sun in Glassfish is replaced with multi instance architecture.

More detail will be provided in the next newsletters. If you want to be involved in these projects, feel free to contact us.

Useful links on OpenESB new platforms

OpenESB SE: https://openesb.atlassian.net/wiki/display/ESBCOMP/Standalone+mode+or+OpenESB+SE
OpenESB for JBoss: https://openesb.atlassian.net/wiki/display/ESBCOMP/Developer+Guide

Other Useful links on OpenESB

OpenESB Download and documentation at: http://www.open-esb.net/
Open-ESB forum and discussion: http://openesb-users.794670.n2.nabble.com/
LinkedIn Group: http://www.linkedin.com/groups?home=&gid=126145&trk=anet_ug_hm
Bugs report: https://openesb.atlassian.net/secure/Dashboard.jspa
Contact the community: contact@open-esb.net

OpenESB V2.3 new components

Two new components have been added in OpenESB V2.3, IEP (intelligent Event Process) and WLM (Worklist Management). These two components were developed by Sun but not embedded in the previous version. For months, no one upgraded the components and consequently, they were not ready to be part in V2.2 or V2.3. This year, thanks to the community, we fixed many bugs and made them available for 2.3.

IEP SE: Intelligent Event Processing Component

IEP SE sends and receives events from all the external systems that OpenESB supports. The events generated by OpenESB components or any external events can be viewed as a **cloud** or a **streams** of events. The IEP SE analyzes both types of events, **Continuous Query Language** (**CQL**) and a rich set of operators to extract accurate information from your events.

If you plan to use IEP, we recommend that you to read these books first.

Event Processing in Action: http://www.manning.com/etzion/
Event processing for business: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470534850.html

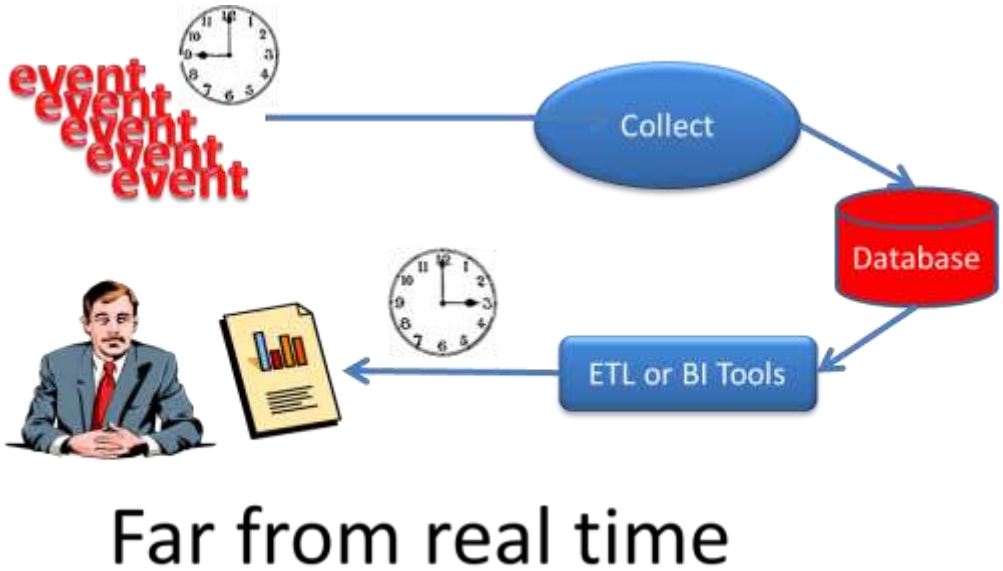
IEP Component Features

Complex Event Processing (CEP)

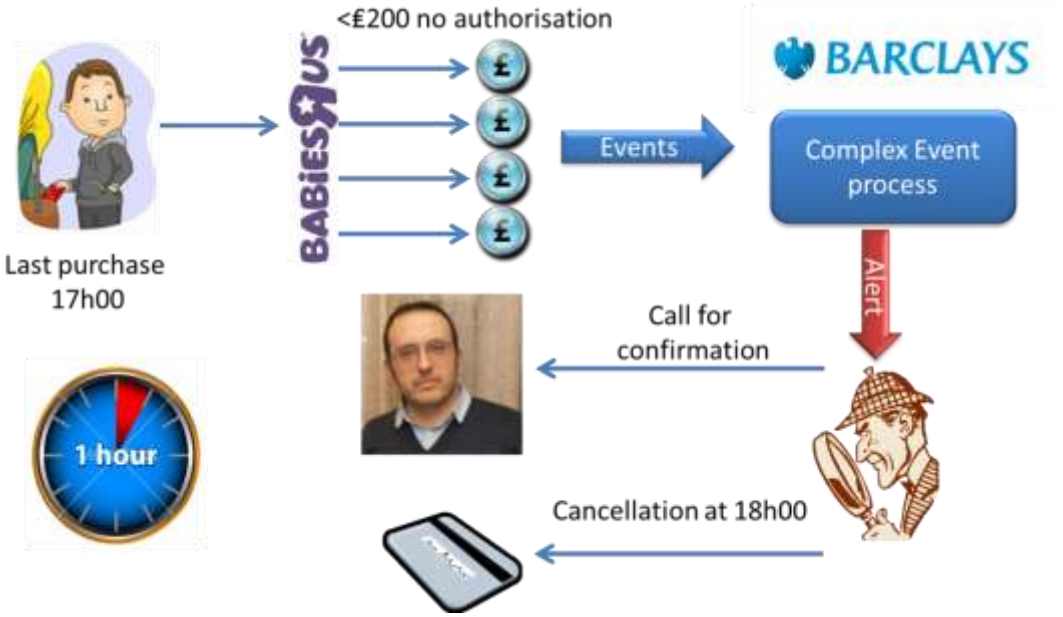
- ✓ Event Stream Processing (ESP)
- ✓ Continuous Query Language (CQL)
- ✓ Rich set of built in operators to do event processing. Supports aggregation, filtering, partitioning, and correlation.
- ✓ Rich GUI tools to create event processes
- ✓ Analyzes messages from clouds and streams
- ✓ Connectivity with all the external systems through Open ESB

To be a bit more concrete, an event can be a purchase order, a new user logging onto the system, selling a bag of fruits (RFID), a financial trade, an airplane landing or email confirmation. Companies receive a great number of events from multiple places which commonly contain ordinary information. IT Teams dedicate few efforts to processing events and often miss critical or important ones (ex: faults, frauds, abnormal behaviour) or fail to gather events to reveal a trend.

The most classical example is ETL or BI tools. These tools produce reports to the management in a batch mode and are not able to provide real time alert if a critical event occurs.



Event processing allows companies to process external or internal events in a mode close to real-time. This is very efficient for fraud detection, alerts and prompt decision. The picture below describes a real example of event processing at the Barclays bank in UK, where the time between fraud and alert is often less than one hour.



At 17h00, someone steals a credit card in a shopping centre and buys many goods cheaper than 200 GBP at "BabiesRUs". 200 GBP is the maximum amount where VISA validation is not required. However, Barclays traces and processes these events and applies the rule: "If more than 3 payments under 200 GBP have been made in the same shop in the same hour, an alert is raised. Barclays gives a call to the VISA card owner to check if the purchases are valid. If not, the card is cancel. This process takes less than one hour to be completed.

More information and tutorial on IEP component on our web site: www.open-esb.net. For professional support and consulting on IEP project, please contact us on contact@open-esb.net

Tips and tricks: How to use XSLT transformations in OpenESB

The BPEL editor is a useful tool to copy single data between data trees but some data transformation operations for adaption to predefined data structures tend to get very tedious in a graphical editor

- ✓ Changing or stripping a namespace
- ✓ Rearranging nodes inside a tree

Assume you have a well-established data structure (with schema validation) in use with *n* partners, but now partner *n+1* insists in using his proprietary interface. Examples:

- ✓ We all use the namespace `http://voodoofone.com/customer/7.1`, but the partner insists on his own namespace, e.g. `http://nsa.gov/prism/1.0`, although the data structure is identical. Or the partner can't handle namespaces at all.
- ✓ We use a `<person>` node consisting of `given-name`, `last-name`, `location`, and some 100 more child nodes, including sub nodes; but the partner only accepts a `<person>` node ordered `last-name`, `given-name`, `location`, …

There is only one little difference, but with a BPEL editor you have to copy all leaves (i.e. the string nodes) manually to implement only one little change. Furthermore, if a 101st element is added, then don't forget to add it to your transformation commands too! Exclude elements with cardinality > 1, which each require a loop coding.

XSLT describes the transformation of a XML data structure. With XSLT, the same task can be formulated concisely by two rules:

1. Generally copy everything from A to B
2. If condition x, then do something special

Thus, there is one default rule. Otherwise, you only have to deal with the special cases. XSLT code examples are presented in the appendix.

To start the transformation, only one BPEL line is required. Although this command is provided by the graphical BPEL editor, you will have to edit the XML code to get a line like this:

```
bpel:doXslTransform('urn:stylesheets:myXSL.xsl', $myData)
```

The path to the XSL file is neither displayed nor edited by the graphical editor, which also tends to insert some useless commas. Therefore, you have to check this command directly in the BPEL XML code.

XSLT propose many powerful features not available in XPath and OpenESB users will have a great benefit in using XSLT when message mapping becomes complex and fastidious. Using XSLT is a nice way to externalize mapping to XML transformation experts. OpenESB does not propose a graphical editor for XSLT. We recommend MapForce from Altova to XSLT developers.

Gerhard Kersten (www.acando.de)

Appendix: XSLT examples

Example 1: Changing a namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Mapping data from NSA to VOO namespace and vice versa -->
```

```
<xsl:stylesheet version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:nsa=" http://nsa.gov/prism/1.0"
  xmlns:voo="http://voodoofone.com/customer/7.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>

<!-- If you find a VOO element: recreate it as NSA -->
<xsl:template match="voo:*">
  <xsl:element name="{concat('nsa:',local-name())}"
    namespace="http://nsa.gov/prism/1.0">
    <!-- Copy all attributes to the new element -->
    <xsl:for-each select="@*">
      <xsl:attribute name="{concat('nsa:',local-name())}">
        <xsl:value-of select="."/>
      </xsl:attribute>
    </xsl:for-each>
    <!-- proceed with my children -->
    <xsl:apply-templates/>
  </xsl:element>
</xsl:template>

<!-- If you find a NSA element: recreate it as VOO -->
<xsl:template match="nsa:*">
  <xsl:element name="{concat('voo:',local-name())}"
    namespace=" http://voodoofone.com/customer/7.1">
    <!-- Copy all attributes to the new element -->
    <xsl:for-each select="@*">
      <xsl:attribute name="{concat('voo:',local-name())}">
        <xsl:value-of select="."/>
      </xsl:attribute>
    </xsl:for-each>
    <!-- proceed with my children -->
    <xsl:apply-templates/>
  </xsl:element>
</xsl:template>

</xsl:stylesheet>
```

Example 2: Stripping a namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Mapping data from VOO to default namespace -->
<xsl:stylesheet version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:nsa=" http://nsa.gov/prism/1.0"
  xmlns:voo="http://voodoofone.com/customer/7.1"
```

```
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>

<!-- If you find a VOO element: recreate it in default namespace -->
<xsl:template match="voo:*">
  <xsl:element name="{local-name()}"
    <!-- Copy all attributes to the new element -->
    <xsl:for-each select="@*">
      <xsl:attribute name="{local-name()}">
        <xsl:value-of select="."/>
      </xsl:attribute>
    </xsl:for-each>
    <!-- proceed with my children -->
    <xsl:apply-templates/>
  </xsl:element>
</xsl:template>

</xsl:stylesheet>
```

Example 3: Rearranging nodes in the tree

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Mapping and reordering data from NSA to VOO namespace and v.v. -->
<xsl:stylesheet version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:nsa=" http://nsa.gov/prism/1.0"
  xmlns:voo="http://voodoofone.com/customer/7.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>

<!-- If you find a VOO element: recreate it as NSA -->
<xsl:template match="voo:*">
...
</xsl:template>
<!-- If you find a NSA element: recreate it as VOO -->
<xsl:template match="nsa:*">
...
</xsl:template>

<!-- If you find a voo:person element: reorder it's children -->
<xsl:template match="voo:person">
  <xsl:element name="{concat('nsa:',local-name())}"
    namespace="http://nsa.gov/prism/1.0">
    <!-- No attributes -->
    <!-- reorder my children, NSA wants last-name first -->
    <xsl:apply-templates select="voo:last-name"/>
```

```
    <xsl:apply-templates select="*[local-name != 'last-name']"/>
  </xsl:element>
</xsl:template>

<!-- If you find a nsa:person element: reorder it's children -->
<xsl:template match="nsa:person">
  <xsl:element name="{concat('voo:',local-name())}"
    namespace="http://nsa.gov/prism/1.0">
    <!-- No attributes -->
    <!-- process my children, VOO wants given-name first -->
    <xsl:apply-templates select="nsa:given-name"/>
    <xsl:apply-templates select="*[local-name != 'given-name']"/>
  </xsl:element>
</xsl:template>
```

Our sponsor this month:

LogiCoy was incorporated in February 2009 with its headquarters in Los Angeles and 24/7 offices in US, UK, and India. It is a Global Information Technology and Services company providing products and services in integration and middleware across various industries, including financial, healthcare, telecommunications, manufacturing, and government. LogiCoy was created from the original architects, authors, developers and engineering managers/directors for Seebeyond eGate, ICAN, Sun JCAPS, Open/GlassFish ESB products from the inception to the end.

Today, LogiCoy is one of the main OpenESB sponsors. Its engineers are committers in the OpenESB codebase and are actively working on new JBI components, enhancements and bug fixes to existing components and new versions of OpenESB. Logicoy provides World Class Global 24x7 Production Support and Monitoring and Migration Services for OpenESB. Please see http://logicoy.com/support for further details. Contact our sponsor at info@logicoy.com

Thank for our English reviewer Kevan Moran