Service Component Architecture WS-BPEL C&I Extensions for Event Processing and Pub/Sub

27 July 2009

Table of Contents

1 Introduction

This document describes the event processing and pub/sub extensions to the SCA WS-BPEL Client and Implementation Specification **[SCA-BPEL]**. It specifies how an SCA Extended WS-BPEL runtime may support event processing and pub/sub described by SCA Assembly Model Specification Extensions for Event Processing and Pub/Sub **[SCA-Eventing]**.

1.1 Normative References

[SCA-BPEL]	OASIS Committee Draft 02, "Service Component Architecture WS-BPEL Client and Implementation Specification Version 1.1," March 2009. http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd02.pdf
[SCA-Eventing]	SCA Assembly Model Specification Extensions for Event Processing and Pub/Sub, April 2009. http://osoa.org/download/attachments/35/SCA_Assembly_Extensions_for_Event_Processing_and_PubSub_V1_0.pdf?version=1
[SCA-POLICY]	OASIS Committee Draft 02, "SCA Policy Framework Version 1.1, " February 2009. http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd02.pdf
[WS-BPEL]	OASIS Standard, "Web Services Business Process Execution Language Version 2.0," http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf

2 SCA Event Processing Extensions to WS-BPEL

2.1 Event Link

SCA extensions for event processing and pub/sub use the concept of events that are produced by producers and consumed by consumer through an intermediary called a channel. There is no interface contract associated either with a producer, or a consumer or a channel. Events instead may be typed using event types defined in an Event Description Language (EDL). Because of the decoupling between the producers and consumers, and the lack of interface contract, a BPEL `partnerLink`, which is typed using `parnerLinkType` (and therefore a WSDL `portType`(s)), is not suitable for event processing.

This specification introduces a new concept of event link. An event link is a way to group event activities and allow a WS-BPEL process to specify producers and consumers of events. The syntax for an `eventLink` is as follows:

```
<sca-bpel:eventLink name="xs:NCName">
  <sca-bpel:consumer requires="list of xs:QName"?>
        <sca:filters/>?
  </sca-bpel:consumer>?
  <sca-bpel:producer requires="list of xs:QName"?
                     typeNames="list of xs:Name"?
                     typeNamespaces="list of xs:anyURI"?>?
</sca-bpel:eventLink>
```

An `eventLink` has the following attributes:

- `name` (1..1) – name of the event link. Each `eventLink` is named, and this name is used for all event interactions via that event link. Similar to a partner link name, the name of the event link must be unique within its scope.

An `eventLink` can be declared to be a consumer, producer or both, using the children elements `consumer` and/or `producer`. An `eventLink` element must contain either the `consumer` element, `producer` element or both.

An `eventLink` element can be declared anywhere a `partnerLink` can.

2.1.1 Consumer

This element declares its parent `eventLink` as a consumer of events. It has the following attributes:

- `requires` (0..n) – a list of policy intents. See SCA Policy Framework for a description of this attribute **[SCA-POLICY]**.

This element has the following children elements:

- `filters` (0..1) – filter elements. See Section 1.13 of the SCA Assembly Model Specification Extensions for Event Processing and Pub/Sub **[SCA-Eventing]** for a detailed description of filters.

2.1.2 Producer

This element declares its parent `eventLink` as a producer of events. It has the following attributes:

- `requires` (0..n) – a list of policy intents. See SCA Policy Framework for a description of this attribute **[SCA-POLICY]**.

- `typeNames` (0..1) – a list of one or more Event Type QNames for events that may be generated by this producer. See the SCA Assembly Model Specification Extensions for Event Processing and Pub/Sub **[SCA-Eventing]** for a detailed description of Event Types.
- `typeNamespaces` (0..1) – a list of one or more Event Type namespace URIs for events that may be generated by this producer. See the SCA Assembly Model Specification Extensions for Event Processing and Pub/Sub **[SCA-Eventing]** for a detailed description of Event Type namespaces.

2.2 WS-BPEL Extension Activities

This specification creates two new WS-BPEL extension activities: eventConsume and eventProduce. These new extension activities are meant for event processing.

An event instance can contain the event data as well as metadata, (for example, creation time) associated with the event instance. Similar to WS-BPEL (as it applies to message metadata access), this specification does not specify how a runtime provides access to the event metadata, except for the event type QName.

2.2.1 `eventConsume` Extension Activity

The `eventConsume` activity is used to consume events from the channel(s) that the consumer corresponding to the event link is connected to, in the composite configuration. The event instance data and type is received in the variables pointed to by the `eventConsume` element.

The syntax for the `eventConsume` activity is as follows:

```
<sca-bpel:eventConsume eventLink="xs:QName"
                       variable="xs:NCName"?
                       typeVariable="xs:NCName"?
                       createInstance="yes|no"?>
  <sca-bpel:correlations>
    <bpel:correlation set="xs:NCName" initiate="yes|join|no" />*
  </sca-bpel:correlations>?
</sca-bpel:eventConsume>
```

An `eventConsume` activity has the following attributes:

- `eventLink` (1..1) – event link consumer used as the event sink.
- `variable` (0..1) – WS-BPEL variable name. If specified, it is used to receive the event instance data. The variable used must not be a variable that is declared using the `@messageType` attribute. The XML Schema type or element used on the variable declaration must match that of the event instance.
- `typeVariable` (0..1) – WS-BPEL event type variable name. If specified, it is used to specify and receive the event instance type name. The variable specified here must be a variable that is declared using the `@type` attribute with a value of "xs:QName". This variable is an in/out variable. I.e., if this variable contains a value, then only an event instance that has the same event type value as the one specified in the variable, is delivered. If the variable does not have a value, then any event available for the corresponding eventLink consumer is delivered. On receipt, this variable always contains the event instance's event type value, if present in the event instance.
- `createInstance` (0..1) – as specified in the Web Services Business Process Execution Language Version 2.0 **[WS-BPEL]**.

An `eventConsume` activity has the following children elements:

- `correlations` (0..n) – WS-BPEL correlations element. This is similar to the correlations element specified for the BPEL receive activity.

Note that WS-BPEL “standard attributes” and “standard elements” are allowed in the `eventConsume` activity and are not included above for brevity.

WS-BPEL allows for concurrent activities. If multiple concurrent `eventConsume` activities are present, an event that arrives on an event link is delivered to at most one activity within a process instance. Which particular concurrent activity is picked is out of scope for this specification.

WS-BPEL pick activity waits for the occurrence of exactly one event from a set of events. It is possible to use `eventConsume` in a pick activity along with `onMessage` and `onAlarm`.

WS-BPEL defines event handlers that run concurrently and get invoked when specified events occur. It is possible to use `eventConsume` in `eventHandlers` along with `onEvent` and `onAlarm`.

2.2.2 `eventProduce` Extension Activity

The `eventProduce` activity is used to produce events for the channel(s) that the producer corresponding to the event link is connected to, in the composite configuration. The event instance data and type are provided by the variables pointed to by the `eventProduce` element.

The syntax for the `eventProduce` activity is as follows:

```
<sca-bpel:eventProduce eventLink="xs:QName"
                       variable="xs:NCName"?
                       typeVariable="xs:NCName">
  <sca-bpel:correlations>
    <bpel:correlation set="xs:NCName" initiate="yes|join|no" />*
  </sca-bpel:correlations>?
</sca-bpel:eventProduce>
```

An `eventProduce` activity has the following attributes:

- `eventLink` (1..1) – event link producer used as the event source.
- `variable` (0..1) – WS-BPEL variable name. If specified, it holds the event instance data. The variable used must not be a variable that is declared using the `@messageType` attribute.
- `typeVariable` (0..1) – WS-BPEL event type variable name. If specified, it holds the event instance type name. The variable specified here must be a variable that is declared using the `@type` attribute with a value of "xs:QName".

An `eventProduce` activity has the following children elements:

- `correlations` (0..n) – WS-BPEL correlations element. This is similar to the correlations element specified for the BPEL reply activity.

Note that WS-BPEL “standard attributes” and “st`andar`d elements” are allowed in the `eventProduce activity` and are not included above for brevity.

3 Producers and Consumers in the Introspected Component Type

For each event link in the process:

1. If there is a `sca-bpel:consumer` child element then it results in an `sca:consumer` element in the introspected component type.
    a. The value of the `@name` attribute of the `sca:consumer` element is the same as that of the `sca-bpel:eventLink` parent element, unless there is a name clash across scopes. If there is a name clash, then the algorithm defined in Section 2.3 of SCA WS-BPEL Client and Implementation Specification **[SCA-BPEL]** for name-mangling, is used to generate a unique consumer name.
    b. The value of the `@requires` attribute of the `sca:consumer` element is the same as that of the `sca-bpel:consumer` element.
    c. The `policySet` attribute is omitted.
    d. The `sca:bindings` child element is omitted.
    e. The `sca:filter` element is copied from the `sca-bpel:consumer` element to the `sca:consumer` element.
2. If there is a `sca-bpel:producer` child element then it results in an `sca:producer` element in the introspected component type.
    a. The value of the `@name` attribute of the `sca:producer` element is the same as that of the `sca-bpel:eventLink` parent element, unless there is a name clash across scopes. If there is a name clash, then the algorithm defined in Section 2.3 of SCA WS-BPEL Client and Implementation Specification **[SCA-BPEL]** for name-mangling, is used to generate a unique producer name.
    b. The value of the `requires`, `typeNames`, and `typeNamespaces` attributes of the `sca:producer` element are the same as that of the `sca-bpel:producer` element.
    c. The `policySet` attribute is omitted.

4 Examples

The example below shows a process containing an event link and an `eventProduce` activity.

```
<process name="PublishEventProcess" ...>

  ...

  <!-- Event link declaration -->
  <sca-bpel:eventLink name="POProducer">
    <sca-bpel:producer typeNames="ns1:POEvent" />
  </sca-bpel:eventLink>

  <variables>
    <variable name="order" element="ns1:PO"/>
    <variable name="eventType" type="xs:QName"/>
    ...
  </variables>

  <sequence name="main">

    ...

    <assign name="AssignPO">
      <copy>
        <from variable="..." />
        <to variable="order" />
      </copy>
      <copy>
        <from><literal>ns1:POEvent</literal></from>
        <to variable="eventType" />
      </copy>
    </assign>

    <!-- Event produce activity -->
    <extensionActivity>
      <sca-bpel:eventProduce eventLink="POProducer" variable="order"
                             typeVariable="eventType"/>
    </extensionActivity>

    ...

  </sequence>

</process>
```

The component type introspected from the above BPEL process is the following:

```
<componentType ...>

  <producer name="POProducer" typeNames="ns1:POEvent" />

</componentType>
```

The next example below shows a process containing an event link and an `eventConsume` activity.

```
<process name="ConsumeEventProcess" ...>

  ...

  <!-- Event link declaration -->
  <sca-bpel:eventLink name="POConsumer">
```

```
    <sca-bpel:consumer>
      <sca:filters>
        <sca:type qnames="ns1:POEvent" />
      </sca:filters>
    </sca-bpel:consumer>
  </sca-bpel:eventLink>


  <variables>
    <variable name="order" element="ns1:PO"/>
    <variable name="eventType" type="xs:QName"/>
    ...
  </variables>

  <sequence name="main">

    ...

    <!-- Event consume activity -->
    <extensionActivity>
      <sca-bpel:eventConsume eventLink="POConsumer" variable="order"
                             typeVariable="eventType" />
    <extensionActivity>

    ...

  </sequence>

</process>
```

The component type introspected from the above BPEL process is the following:

```
<componentType ...>

  <consumer name="POConsumer">
      <filters>
        <type qnames="ns1:POEvent" />
      </filters>
  </consumer>

</componentType>
```

A. XML Schema

```
<schema
   targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca-bpel/200801"
   xmlns:sca-bpel="http://docs.oasis-open.org/ns/opencsa/sca-bpel/200801"
   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"
   xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/abstract"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <!-- SCA-Assembly XML Schema -->
   <import
      namespace="http://docs.oasis-open.org/ns/opencsa/sca/200903"
      schemaLocation="http://docs.oasis-open.org/opencsa/sca-assembly/sca-
core-1.1-cd03.xsd" />

   <!-- WS-BPEL 2.0 XML Schema for abstract BPEL -->
   <import
      namespace="http://docs.oasis-open.org/wsbpel/2.0/process/abstract"
      schemaLocation="http://docs.oasis-
open.org/wsbpel/2.0/OS/process/abstract/ws-bpel_abstract_common_base.xsd" />

  <!--
     SCA BPEL eventLink. This is analogous to BPEL partnerLink for the
     eventing world.
  -->
   <element name="eventLink" type="sca-bpel:EventLinkType" />

   <complexType name="EventLinkType">
         <sequence>
            <element ref="sca-bpel:consumer" minOccurs="0" maxOccurs="1" />
            <element ref="sca-bpel:producer" minOccurs="0" maxOccurs="1" />
            <any namespace="##other" processContents="lax"
                 minOccurs="0" maxOccurs="unbounded" />
         </sequence>
         <attribute name="name" type="xsd:NCName" use="required" />
         <anyAttribute namespace="##other" processContents="lax" />
   </complexType>

  <!--
     SCA BPEL consumer. This can be present as a child of eventLink.
     This results in a SCA consumer showing up in the Component Type.
  -->
   <element name="consumer" type="sca-bpel:ConsumerType" />

   <complexType name="ConsumerType">
         <sequence>
            <element ref="sca:filters" minOccurs="0" maxOccurs="1" />
            <any namespace="##other" processContents="lax"
                 minOccurs="0" maxOccurs="unbounded" />
         </sequence>
         <anyAttribute namespace="##other" processContents="lax" />
   </complexType>

  <!--
     SCA BPEL producer. This can be present as a child of eventLink.
     This results in a SCA producer showing up in the Component Type.
  -->
   <element name="producer" type="sca-bpel:ProducerType" />

```

```
   <complexType name="ProducerType">
         <sequence>
            <any namespace="##other" processContents="lax"
                 minOccurs="0" maxOccurs="unbounded" />
         </sequence>
         <attribute name="requires" type="sca:listOfQNames" />
         <attribute name="typeNames" type="sca:listOfQNames" />
         <attribute name="typeNamespaces" type="sca:listOfAnyURIs" />
         <anyAttribute namespace="##other" processContents="lax" />
   </complexType>

  <!--
     SCA BPEL consume extension activity. This activity allows one to consume
     an event.
  -->

   <element name="eventConsume" type="sca-bpel:EventConsume" />

   <complexType name="EventConsume">
     <complexContent>
       <extension base="bpel:tActivity">
          <attribute name="eventLink" type="QName" use="required" />
          <attribute name="variable" type="bpel:BPELVariableName" />
          <attribute name="typeVariable" type="bpel:BPELVariableName" />
          <attribute name="createInstance" type="bpel:tBoolean"
                     default="no" />
       </extension>
     </complexContent>
   </complexType>

  <!--
     SCA BPEL produce extension activity. This activity allows one to produce
     an event.
  -->
   <element name="eventProduce" type="sca-bpel:EventProduce" />

   <complexType name="EventProduce">
     <complexContent>
       <extension base="bpel:tActivity">
          <attribute name="eventLink" type="QName" use="required" />
          <attribute name="variable" type="bpel:BPELVariableName" />
          <attribute name="typeVariable" type="bpel:BPELVariableName" />
       </extension>
     </complexContent>
   </complexType>

   <element name="correlations" type="bpel:tCorrelations" />

</schema>
```