Service Component Architecture WS-BPEL Client and Implementation Specification Version 1.1

Committee Draft 02 / Public Review Draft 01

5 March 2009

Specification URIs:

This Version:

http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd02.html
http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd02.doc
http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd02.pdf (Authoritative)

Previous Version:

http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd-01.html
http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd-01.doc
http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec-cd-01.pdf

Latest Version:

http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec.html
http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec.doc
http://docs.oasis-open.org/opencsa/sca-bpel/sca-bpel-1.1-spec.pdf

Technical Committee:

OASIS SCA-BPEL TC

Chair(s):

Anish Karmarkar, Oracle
Sanjay Patil, SAP

Editor(s):

Najeeb Andrabi, TIBCO Software
Martin Chapman, Oracle
Dieter König, IBM
Michael Rowley, Active Endpoints
Ivana Trickovic, SAP

Related work:

This specification is related to:

- Service Component Architecture – Assembly Model Specification – Version 1.1
- Service Component Architecture – Policy Framework Specification – Version 1.1
- Web Services – Business Process Execution Language – Version 2.0 – http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Declared XML Namespace(s):

- http://docs.oasis-open.org/ns/opencsa/sca-bpel/200801

Abstract:

The Service Component Architecture (SCA) WS-BPEL Client and Implementation model specifies how WS-BPEL 2.0 can be used with SCA. The goal of the specification is to address the following scenarios.

Start from WS-BPEL process. It should be possible to use any valid WS-BPEL process definition as the implementation of a component within SCA. In particular, it should be possible to generate an SCA Component Type from any WS-BPEL process definition and use that type within an SCA assembly. Most BPEL4WS 1.1 process definitions may also be used with SCA by using the backward compatibility approach described in section 4.

Start from SCA Component Type. It should be possible to use WS-BPEL to implement any SCA *Component Type* that uses only WSDL interfaces to define services and references, possibly with some SCA specific extensions used in process definition.

Start from WS-BPEL with SCA extensions. It should be possible to create a WS-BPEL process definition that uses SCA extensions and generate an SCA Component Type and use that type within an SCA assembly. Some SCA capabilities (such as properties and multi-party references) can only be used by WS-BPEL process definitions that use SCA extensions**.**

Status:

This document was last revised or approved by the OASIS Service Component Architecture / BPEL (SCA-BPEL) TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/sca-bpel/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/sca-bpel/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/sca-bpel/.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

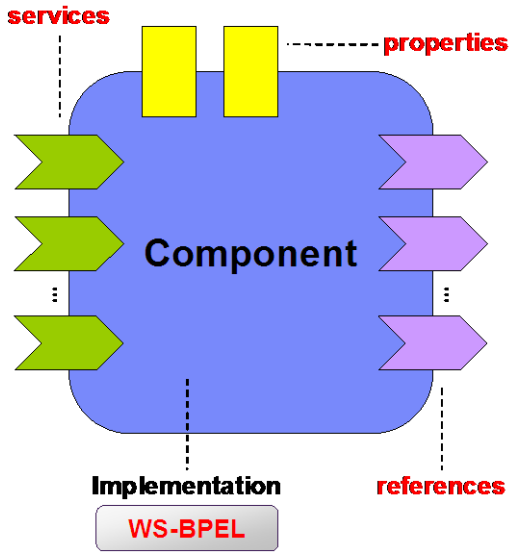
OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", [insert specific trademarked names and abbreviations here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

This specification describes how a WS-BPEL process definition can be used as the implementation of an SCA component.



For an SCA component to use a WS-BPEL process as an implementation, it uses an `<implementation.bpel/>` element::

```
<component ... >

   ...

   <implementation.bpel process="xsd:QName" />

   ...

</component>
```

The only aspect of this that is specific to WS-BPEL is the `<implementation.bpel>` element. [SBPEL1001] The `process` attribute of the `<implementation.bpel>` element MUST be the QName of an executable WS-BPEL process.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

1.2 Normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[SCA-Assembly] OASIS Committee Draft 03, *Service Component Architecture – Assembly Model Specification – Version 1.1*

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd03.pdf

[SCA-PolicyFramework] OASIS Committee Draft 02, *Service Component Architecture – Policy Framework Specification – Version 1.1,* http://docs.oasis-open.org/opencsa/sca-policy/sca-policy-1.1-spec-cd-02.pdf

[WS-BPEL] OASIS Standard, OASIS Web Services – Business Process Execution Language – Version 2.0, April 2007 http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

1.3 Non-Normative References

N/A

1.4 Naming Conventions

This specification follows some naming conventions for artifacts defined by the specification, as follows:

- For the names of elements and the names of attributes within XSD files, the names follow the CamelCase convention, with all names starting with a lower case letter.
  e.g. `<element name="componentType" type="sca:ComponentType"/>`
- For the names of types within XSD files, the names follow the CamelCase convention with all names starting with an upper case letter.
  e.g. `<complexType name="ComponentService">`

2 Introspected Component Type of a WS-BPEL Process

While a WS-BPEL process definition provides an implementation that can be used by a component, the process definition also determines the introspected ComponentType of any SCA component that uses that implementation. The introspected component type represents the aspects of the implementation that SCA needs to be aware of in order to support assembly and deployment of components that use that implementation. The generic form of a component type is defined in the SCA Assembly Specification **[SCA-Assembly]**.

```
<componentType ... >

   <service name="xsd:NCName" ... > ... </service>

   <reference name="xsd:NCName" ... > ... </reference>

   <property name="xsd:NCName" ... > ... </property>

   <implementation ... />

</componentType>
```

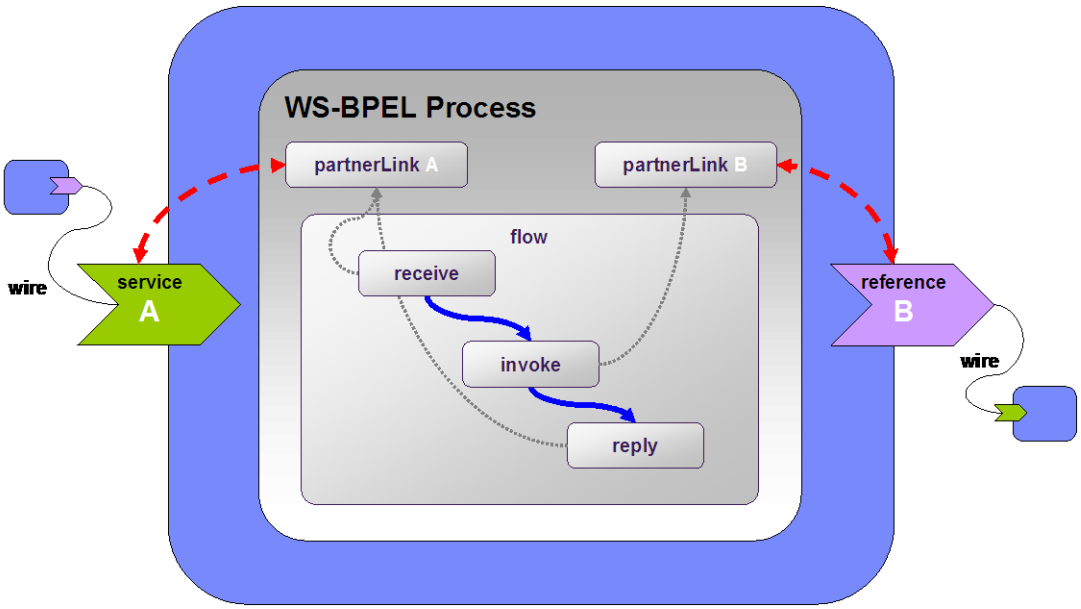
The SCA Assembly Specification defines an *asyncInvocation* policy intent for long-running operations. BPEL processes that implement long-running request-response operations are encouraged to use interfaces marked with this intent.

2.1 Services and References

In SCA, both *services* and *references* correspond to WS-BPEL's concept of partner link. In SCA, the difference between a service and a reference is determined by which party sends the first message in a conversation. No matter of how many messages a bi-directional conversation involves or how long it takes, there is always a first message. The sender of the first message is considered to be the *client* and the receiver is the *service provider*. Messages that go from the service provider to the client are called *callback messages*.

WS-BPEL's partner links are not differentiated based on who sends the first message. So, in order to map a WS-BPEL process to an SCA Component Type, it is necessary to determine which role sends the first message. A simple static analysis of the control flow, which does not involve determining the values of any expressions, is used to determine which role can send the first message.

It is also possible to override the default mapping of partner links to services or references as described by explicitly marking the partner link with an SCA attribute that describes the service or reference (i.e. `sca-bpel:service` or `sca-bpel:reference`). These attributes are described in section 3.3.



2.1.1 Generating Services and References

The following sections describe the rules that determine the contents of the introspected component type for a WS-BPEL process.

[SBPEL2001] If a partner link specifies a `sca-bpel:service` attribute, then a service MUST be generated for the introspected component type. [SBPEL2002] The name of the service MUST be the value of the `sca-bpel:service` attribute.

[SBPEL2003] If a partner link specifies a `sca-bpel:reference` attribute, then a reference MUST be generated for the introspected component type. [SBPEL2004] The name of the reference MUST be the value of the `sca-bpel:reference` attribute.

[SBPEL2005] If neither `sca-bpel:service` nor `sca-bpel:reference` is present on the partner link, then if a static analysis of the process determines that it is possible that the first message for a partner link will be received in a `<receive>` activity, the `<onMessage>` element of a `<pick>` activity or the `<onEvent>` element of an event handler then the introspected component type MUST include an SCA service that corresponds to the partner link in the component type. [SBPEL2006] If the name of the partner link is unique within the process, then it MUST be used as the name of the service. Otherwise, the name is determined according to the rules of section 2.3.

[SBPEL2007] If the rules [SBPEL2001]-[SBPEL2006] do not determine that the partner link maps to an SCA service, then the introspected component type MUST include an SCA reference that corresponds to the partner link in the component type. [SBPEL2008] If the name of the partner link is unique within the process, then it MUST be used as the name of the reference. Otherwise, the name is determined according to the rules of section 2.3.

[SBPEL2009] The *multiplicity* of the reference MUST be determined according to the algorithm defined by rules [SBPEL2010]-[SBPEL2013].

1. **Multi-Reference**. [SBPEL2010] If the partner link is declared with an `sca-bpel:multiRefFrom="aVariableName"` extension, the multiplicity of the SCA reference MUST be determined by the multiplicity attribute of `sca-bpel:multiReference` extension used in the corresponding variable. Details of these extensions are described in section 3.2.
2. **Required Reference**. [SBPEL2011] If [SBPEL2010] does not apply and the partner link has `initializePartnerRole="yes"`, then the multiplicity MUST be `"1..1"` (i.e. it is a *required reference*).

3. **Stub Reference**. [SBPEL2012] If neither [SBPEL2010] nor [SBPEL2011] apply and the analysis of the process determines that the first use of the partner link by any activity is in an `<assign>` activity that sets the partner role, then the multiplicity MUST be `"0..1"` and the attribute `wiredByImpl` MUST be set to `"true"`. A reference with `wiredByImpl="true"` is referred to as a *stub reference*. Although the target can't be set for such a reference, SCA can still apply bindings and policies to it and potentially need to set the endpoint address for callbacks, if the interface is bi-directional.

4. **Optional Reference**. [SBPEL2013] If neither [SBPEL2010] nor [SBPEL2011] nor [SBPEL2012] apply, then the `multiplicity` MUST be `"0..1"`.

2.1.2 Handling @initializePartnerRole on Services

SCA has no concept of multiplicity on services, but partner links that map to services can still be marked with an `initializePartnerRole` attribute. [SBPEL2014] If `initializePartnerRole="yes"` is specified for a partner link and the partner link maps to a service in the component type, then any component that uses this business process as an implementation MUST configure the corresponding service to use a binding that knows the identity of the partner as soon as the partner link becomes active (e.g. the binding cannot depend on using a "reply-to" field as the mechanism to initialize the partner role).

2.2 Partner Link Types and SCA Interfaces

When a partner link is determined to correspond to an SCA service, the type of the service is determined by the partner link type of the partner link. [SBPEL2015] The WSDL port type in the `<interface.wsdl>` declaration for the service in the introspected component type MUST be the same as the port type of the `myRole` of the partner link. [SBPEL2016] If the partner link type has two roles, then the `<interface.wsdl>` declaration MUST also have a `@callbackInterface` attribute whose value points to the same WSDL port type as the `partnerRole` of the partner link.

Consider an example that uses one of the partner link types used as an example in the WS-BPEL specification. The partner link type definition is:

```
<plnk:partnerLinkType name="invoicingLT">
   <plnk:role name="invoiceService"
      portType="pos:computePricePT" />
   <plnk:role name="invoiceRequester"
      portType="pos:invoiceCallbackPT" />
</plnk:partnerLinkType>
```

The "invoiceProcess", which provides invoice services, would define a partner link that uses that type with a declaration that would look like:

```
<partnerLink name="invoicing"
   partnerLinkType="lns:invoicingLT"
   myRole="invoiceService"
   partnerRole="invoiceRequester" />
```

Somewhere in the process, a start activity would use that partner link, which might look like:

```
<receive partnerLink="invoicing"
   portType="pos:computePricePT"
   operation="initiatePriceCalculation"
   variable="PO"
   createInstance="yes" />
```

Because the partner link is used in a start activity, SCA maps that partner link to a service for on the component type. In this case, the service element of the component type would be:

```
<service name="invoicing">
   <interface.wsdl
      interface="http://manufacturing.org/wsdl/purchase#
                              wsdl.interface(computePricePT)"
      callbackInterface="http://manufacturing.org/wsdl/purchase#
                                    wsdl.interface(invoiceCallbackPT)" />
</service>
```

Conversely, when a partner link is determined to correspond to an SCA reference in the introspected component type, then interface for the reference is also determined by the partner link type, but with the roles reversed. [SBPEL2017] The WSDL port type in the `<interface.wsdl>` declaration for the reference MUST be the same as the port type of the `partnerRole` of the partner link. [SBPEL2018] If the partner link type has two roles, then the `<interface.wsdl>` declaration MUST also have a `@callbackInterface` attribute whose value points to the same WSDL port type as the `myRole` of the partner link.

2.3 Handling of Local Partner Links

It is possible to declare partner links local to a `<scope>` in WS-BPEL, besides declaring partner links at the `<process>` level. The names of partner link declared in different `<scope>` could potentially share the identical name. [SBPEL2019] When multiple partner links share the same name, the scheme defined by [SBPEL2020]-[SBPEL2022] MUST be used to disambiguate different occurrences of partner link declaration.

- Let `"originalName"` be the original NCName used in multiple partner link declarations.
- [SBPEL2020] The introspected component type MUST include services or references corresponding to these partner links with names: `"_orginalName_1"` to `"_orginalName_N"`. Whether the partner link corresponds to a service or reference does not affect the name used. [SBPEL2021] The number suffixes for the partner links MUST be based on the lexical order of the corresponding partner link occurrences in the process definition.
- [SBPEL2022] If any `"_orginalName_i"` (where 1 <= i <= N) is already the name of a partner link declaration in the process definition, additional underscore characters MAY be added at the beginning of all aliases consistently to avoid collision.

3 SCA Extensions to WS-BPEL

It is possible to use WS-BPEL processes in conjunction with SCA, while the processes have no knowledge of SCA. A few SCA concepts are only available to WS-BPEL processors that support SCA specific extensions. The capabilities that require knowledge of SCA are provided by an SCA extension, whose namespace is "http://docs.oasis-open.org/ns/opencsa/sca-bpel/200801".

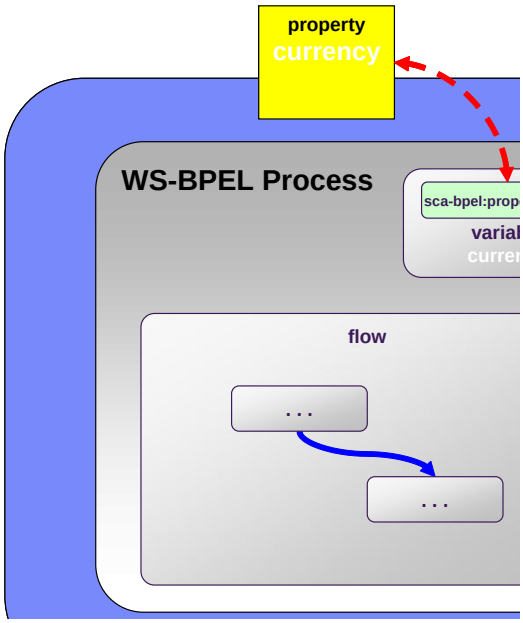
Whether this extension is mandatory or optional is specified by the mustUnderstand attribute as described in section 14 of the WS-BPEL 2.0 specification **[SCA-Assembly]**.

An example, where the SCA extension is mandatory, is as follows:

```
<process ...>
   <extensions>
      <extension
         namespace="http://docs.oasis-open.org/ns/opencsa/sca-bpel/200801"
         mustUnderstand="yes" />
   </extensions>
   ...
</process>
```

3.1 Properties

A WS-BPEL variable declaration can include an SCA extension that says that the variable represents an SCA property for the component represented by the WS-BPEL process.



The declaration looks like the following:

```
<variable name="currency" type="xsd:string"
   sca-bpel:property="yes" />
```

When `sca-bpel:property="yes"` is used on a variable declaration, the name of the variable is used as the name of a property of the component type represented by the WS-BPEL process. [SBPEL3001] The name of a variable used as a property of the component MUST be unique within the process.

If the variable has an initialization from-spec, then that becomes the default value for the variable in cases where the SCA component does not provide a value for that property.

If the from-spec is a literal value, where it has the following form:

```
    <from><literal>literal value</literal></from>
```

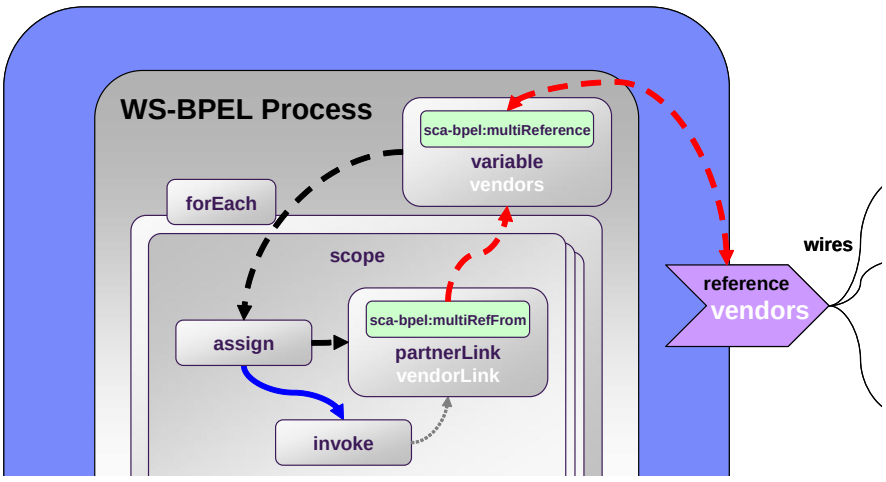
then the literal value will be represented as the default value in the component type for the process. Any other kind of initialization from-spec will not be represented in the component type. However, even though the other kinds of initialization from-spec are not represented in the component type, they would still be computed and used as the default value for the property when the component does not provide a value for that property.

[SBPEL3002] If a value is provided for a property, any initialization from-spec MUST still be evaluated, but the value of the variable will be changed to the provided property value immediately after the initialization is evaluated, and specifically, before any following variable initialization from-spec is evaluated. Thus, any side effects that result from the execution of the initialization from-spec will occur irrespective of whether the property is set.

[SBPEL3003] If a BPEL variable that is used as a property has an initialization from-spec then `mustSupply="false"` MUST be specified on the component type property declaration, even if the default value is not literal and therefore not represented in the component type.

3.2 Multi-Valued References

Component types can declare references with a multiplicity that allows a single reference to be wired to multiple targets. An example use of this capability is a purchasing component wired to a list of accepted vendors. SCA assumes that each programming language binding will provide its own approach for making the list of targets available within that programming language.



[SBPEL3004] In a WS-BPEL process definition, a variable MAY include an `sca-bpel:multiReference` extension element that declares that the variable represents a multi-valued reference. [SBPEL3005] When a variable declaration contains the `sca-bpel:multiReference`

extension, the type of the variable MUST be an element of `sca-bpel:serviceReferenceList`. However, since that type only specifies that the variable holds a list of endpoint references, the `sca-bpel:multiReference` element also has attributes to specify the partner link type and partner role of the target of the reference. [SBPEL3006] The introspected component type MUST include a reference with a multiplicity of either `"0..n"` or `"1..n"` that corresponds to a variable with the `sca-bpel:multiReference` element. [SBPEL3007] The type of the reference MUST be determined by the partner link type and the partner role attributes of the `sca-bpel:multiReference` extension element. [SBPEL3008] The `sca-bpel:multiRefFrom` attribute MUST NOT be specified for a partner link with a myRole attribute referencing a role which is the only role of a partner link type. [SBPEL3009] The `sca-bpel:multiRefFrom` attribute MUST NOT be specified for a partner link that has the `sca-bpel:service` attribute.

An example of a variable that represents a list of references to vendors would look like:

```
<variable name="vendors" element="sca-bpel:serviceReferenceList">
   <sca-bpel:multiReference partnerLinkType="pos:vendorPT"
      partnerRole="vendor" />
</variable>
```

Syntax of this extension:

```
<sca-bpel:multiReference partnerLinkType="xsd:QName"
   partnerRole="xsd:NCName"
   multiplicity="0..n or 1..n"? />
```

The default value of multiplicity is `"1..n"`.

The `sca-bpel:serviceReferenceList` element declaration is the following:

```
<xsd:element name="serviceReferenceList">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="sref:service-ref"
            minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

A typical use of a variable that holds a multi-valued reference would be to have a `<forEach>` activity with an iteration for each element in the list. The body of the `<forEach>` activity would declare a local partner link and assign one of the list elements to the local partner link. Such a local partner link is typically categorized as the "References" case 1 listed in section 2.1.

To assist a more effective SCA modeling, another SCA extension is introduced to associate a multi-valued reference, manifested as a `"sca-bpel:serviceReferenceList"` variable with a partner link. This extension is in an attribute form attached to the partner link declaration. Syntax of this extension is:

```
<partnerLink ... sca-bpel:multiRefFrom="bpel:BPELVariableName" />
```

[SBPEL3010] The value of the `sca-bpel:multiRefFrom` attribute MUST refer to the name of a variable manifesting an SCA multi-valued reference. [SBPEL3011] The `partnerLinkType` and `partnerRole` attributes of the partner link and multi-valued reference variable MUST be matched. [SBPEL3012] There MUST be at least one code-path where the values from the multi-valued reference variable are copied to the `partnerRole` of the partner link.

If any above constraints are violated, it will be considered an error during static analysis.

When this `sca-bpel:multiRefFrom` extension is applied to pair up a multi-valued reference variable and a partner link which is categorized as the "References" case 1 (as described in section 2.1), the partner link and variable are manifested as a single multi-valued reference entity in SCA assembly model using the name of the variable. If the interface involved is bi-directional, this implies the wiring of the bi-directional interface as a single reference in SCA.

For example:

```
<process>
   ...
   <variable name="vendors" element="sca-bpel:serviceReferenceList">
      <sca-bpel:multiReference partnerLinkType="pos:vendorPT"
         partnerRole="vendor" />
   </variable>
   ...
   <forEach counterName="idx" ...>
      <startCounterValue>1</startCounterValue>
      <finalCounterValue>
         count($vendors/sref:service-ref)
      </finalCounterValue>
      ...
      <scope>
         ...
         <partnerLink name="vendorLink"
            partnerLinkType="pos:vendorPT"
            partnerRole="vendor"
            myRole="quoteRequester"
            sca-bpel:multiRefFrom="vendors" />
         ...
         <assign>
            <copy>
               <from>$vendors/sref:service-ref[$idx]</from>
               <to partnerLink="vendorLink" />
            </copy>
         </assign>
         ...
      </scope>
   </forEach>
   ...
</process>
```

A multi-valued reference named "vendors" is declared in the example above. The partner link named "vendorLink", which is categorized as the "References" case 1, is not manifested directly into the SCA Assembly Model. The extra `sca-bpel:multiRefFrom="vendors"` extension associates the "vendorLink" partner link with multi-valued reference variable "vendors". Consequently, the partner link and variable are manifested as a single multi-valued reference named "vendors" in SCA. This makes the SCA Assembly modeling easier to follow.

3.3 Partner Link Mapping to Services and References

[SBPEL3013] A WS-BPEL process definition MAY override the default mapping of partner links to services or references as described in section 2.1 by explicitly marking the partner link with an SCA attribute that describes the service or reference.

[SBPEL3014] To explicitly map a partner link to a service, the `sca-bpel:service` attribute MAY be specified for the partner link. Example:

```
<partnerLink ... sca-bpel:service="xsd:NCName" />
```

[SBPEL3015] The name of the service specified in the `sca-bpel:service` attribute MUST NOT conflict with any other service name generated in the component type for this process.
[SBPEL3016] The `sca-bpel:service` attribute MUST NOT be specified for a partner link with a `partnerRole` attribute referencing a role which is the only role of a partner link type.

[SBPEL3017] To explicitly map a partner link to a reference, the `sca-bpel:reference` attribute MAY be specified for the partner link. Example:

```
<partnerLink ... sca-bpel:reference="xsd:NCName" />
```

[SBPEL3018] The name of the reference specified in the `sca-bpel:service` attribute MUST NOT conflict with any other reference name generated in the component type for this process.
[SBPEL3019] The `sca-bpel:reference` attribute MUST NOT be specified for a partner link with a `myRole` attribute referencing a role which is the only role of a partner link type.

When either of these attributes is used, the componentType will include a service or reference with the given name and no other service or reference will be generated for the partner link. The type of that service or reference is unaffected (it will be as specified in section 2.2).

[SBPEL3020] A process MUST NOT include both `sca-bpel:service` and `sca-bpel:reference` attributes on a single partner link.

3.4 Required Intents for Partner Links

[SBPEL3021] An SCA extension attribute `sca-bpel:requires` MAY be used to declare required policy intents on a partner link. This can be used by WS-BPEL process designers to require specific abstract policies to be associated with the partner link, without limiting the bindings that can be used for the partner link. The form of the attribute is the following:

```
<partnerLink ... sca-bpel:requires="sca:listOfQNames" />
```

[SBPEL3022] The contents of the `sca-bpel:requires` attribute MUST be a space separated list of SCA intent QNames, exactly as specified in the SCA Policy Framework Specification for the contents of the `@sca:requires` attribute.

[SBPEL3023] If the `sca-bpel:requires` attribute is specified, the corresponding service or reference in the introspected component type MUST include an `@sca:requires` attribute with the same contents.

4 Using BPEL4WS 1.1 with SCA (Non-Normative)

A BPEL4WS 1.1 process definition can be used as the implementation of an SCA component. The syntax introduced in section Introduction is used to define a component having a BPEL4WS 1.1 process as the implementation. In this case, the process attribute specifies the target QName of a BPEL4WS 1.1 executable process.

A BPEL4WS 1.1 process definition can be used to generate an SCA Component Type.

5 Conformance

There are two categories of artifacts that this specification defines conformance for: SCA Documents and SCA Runtimes.

5.1 SCA WS-BPEL Document

A SCA WS-BPEL Document is a document that complies with the requirements defined by WS-BPEL 2.0 **[WS-BPEL]** and MAY include the SCA WS-BPEL extensions defined in Section 3. Any document using these extensions must comply with the sca-bpel schema and any other constraints defined by this specification.

5.2 SCA Runtimes

There are two conformance options defined by this specification:

1. Implementations of an SCA WS-BPEL Runtime
2. Implementations of an SCA Extended WS-BPEL Runtime.

5.2.1 SCA WS-BPEL Runtime

An implementation that claims to conform to an SCA WS-BPEL Runtime MUST meet the following conditions:

1. The implementation MUST meet all the conformance requirements defined by the SCA Assembly Model Specification **[SCA-Assembly]** i.e. it MUST be a conforming SCA Runtime.
2. The implementation MUST be a compliant WS-BPEL Processor as defined in WS-BPEL 2.0. It must accept and process WS-BPEL 2.0 process descriptions in a manner defined by WS-BPEL 2.0.
3. The SCA BPEL extensions defined in this specification MUST be treated as WS-BPEL 2.0 extensions. WS-BPEL process descriptions containing the SCA BPEL extensions MAY be rejected.
4. With the exception of the SCA BPEL extensions, the implementation MUST comply with all the normative statements in this specification (Appendix B), notably all the MUST statements have to be implemented.

5.2.2 SCA Extended WS-BPEL Runtime

An implementation that claims to conform to an SCA Extended WS-BPEL Runtime MUST meet the following conditions:

1. The implementation MUST meet the conditions for an SCA WS-BPEL Runtime above with the exception that SCA BPEL extensions defined in this specification MUST be supported. WS-BPEL process descriptions containing the SCA BPEL extensions MUST NOT be rejected
2. The implementation MUST support the SCA BPEL extensions defined in Section 3, and MUST implement them as defined.

A. XML Schemas

XML Schema for SCA-BPEL Extensions of SCA Elements

The definitions contributed by the SCA-BPEL specifications to the common SCA namespace are also provided in a separate XML Schema artifact.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
   Copyright (c) OASIS Open 2008. All Rights Reserved.
-->
<schema
   targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200903"
   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200903"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <!-- SCA-Assembly XML Schema -->
   <include
      schemaLocation="sca-core-1.1-cd03.xsd" />

   <!-- SCA-BPEL Component Implementation Type -->
   <element name="implementation.bpel"
      type="sca:BPELImplementation" substitutionGroup="sca:implementation" />

   <complexType name="BPELImplementation">
      <complexContent>
         <extension base="sca:Implementation">
            <sequence>
               <any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded" />
            </sequence>
            <attribute name="process" type="QName" use="required" />
            <anyAttribute namespace="##any" processContents="lax" />
         </extension>
      </complexContent>
   </complexType>

</schema>
```

XML Schema for SCA-BPEL Extensions of WS-BPEL 2.0

The definitions of SCA-BPEL extensions to WS-BPEL 2.0 are also provided in a separate XML Schema artifact.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
   Copyright (c) OASIS Open 2008. All Rights Reserved.
-->
<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/executable"
   xmlns:sref="http://docs.oasis-open.org/wsbpel/2.0/serviceref"
```

```
   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="sca-core-1.1-cd03.xsd" />

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

   <!-- WS-BPEL 2.0 XML Schema for Service References -->
   <import
      namespace="http://docs.oasis-open.org/wsbpel/2.0/serviceref"
      schemaLocation="http://docs.oasis-open.org/wsbpel/2.0/OS/serviceref/ws-
bpel_serviceref.xsd" />

   <!--
      WS-BPEL extension attribute for a bpel:variable associated with
      an SCA property
   -->
   <attribute name="property" type="bpel:tBoolean" />

   <!--
      WS-BPEL extension attribute for a bpel:partnerLink associated with
      an SCA multi-valued reference
   -->
   <attribute name="multiRefFrom" type="bpel:BPELVariableName" />

   <!--
      WS-BPEL extension element for a bpel:variable holding
      an SCA multi-valued reference
   -->
   <element name="multiReference">
      <complexType>
         <simpleContent>
            <extension base="xsd:string">
               <attribute name="partnerLinkType" type="QName" />
               <attribute name="partnerRole" type="NCName" />
               <attribute name="multiplicity"
                  type="sca-bpel:Multiplicity"
                  use="optional" default="1..n" />
            </extension>
         </simpleContent>
      </complexType>
   </element>

   <simpleType name="Multiplicity">
      <restriction base="string">
         <enumeration value="0..n" />
         <enumeration value="1..n" />
      </restriction>
   </simpleType>

```

```
   <!--
      SCA-BPEL element representing a list of WS-BPEL service references
   -->
   <element name="serviceReferenceList">
      <complexType>
         <sequence>
            <element ref="sref:service-ref"
               minOccurs="0" maxOccurs="unbounded" />
         </sequence>
      </complexType>
   </element>

   <!--
      WS-BPEL extension attribute for a bpel:partnerLink explicitly naming
      the service that should be generated for this partnerLink in the
      component type.
   -->
   <attribute name="service" type="xsd:NCName" />

   <!--
      WS-BPEL extension attribute for a bpel:partnerLink explicitly naming
      the reference that should be generated for this partnerLink in the
      component type.
   -->
   <attribute name="reference" type="xsd:NCName" />

   <!--
      WS-BPEL extension attribute for a bpel:partnerLink specifying required
      intents for the service or reference that is generated for
      this partner link.
   -->
   <attribute name="requires" type="sca:listOfQNames" />

</schema>
```

B. Conformance Items

This section contains a list of conformance items for the SCA-BPEL specification.

Conformance ID	Description
[SBPEL1001]	The process attribute of the `<implementation.bpel>` element MUST be the QName of an executable WS-BPEL process.
[SBPEL2001]	If a partner link specifies a `sca-bpel:service` attribute, then a service MUST be generated for the introspected component type.
[SBPEL2002]	The name of the service MUST be the value of the `sca-bpel:service` attribute
[SBPEL2003]	If a partner link specifies a `sca-bpel:reference` attribute, then a reference MUST be generated for the introspected component type.
[SBPEL2004]	The name of the reference MUST be the value of the `sca-bpel:reference` attribute.
[SBPEL2005]	If neither `sca-bpel:service` nor `sca-bpel:reference` is present on the partner link, then if a static analysis of the process determines that it is possible that the first message for a partner link will be received in a `<receive>` activity, the `<onMessage>` element of a `<pick>` activity or the `<onEvent>` element of an event handler then the introspected component type MUST include an SCA service that corresponds to the partner link in the component type.
[SBPEL2006]	If the name of the partner link is unique within the process, then it MUST be used as the name of the service.
[SBPEL2007]	If the rules [SBPEL2001]-[SBPEL2006] do not determine that the partner link should map to an SCA service, then the introspected component type MUST include an SCA reference that corresponds to the partner link in the component type.
[SBPEL2008]	If the name of the partner link is unique within the process, then it MUST be used as the name of the reference.
[SBPEL2009]	The multiplicity of the reference MUST be determined according to the algorithm defined by rules [SBPEL2010]-[SBPEL2013].
[SBPEL2010]	If the partner link is declared with an `sca-bpel:multiRefFrom="aVariableName"` extension, the multiplicity of the SCA reference MUST be determined by the `multiplicity` attribute of `sca-bpel:multiReference` extension used in the corresponding variable.
[SBPEL2011]	If [SBPEL2010] does not apply and the partner link has `initializePartnerRole="yes"`, then the multiplicity MUST be `"1..1"`.

[SBPEL2012]	If neither [SBPEL2010] nor [SBPEL2011] apply and the analysis of the process determines that the first use of the partner link by any activity is in an `<assign>` activity that sets the partner role, then the multiplicity MUST be `"0..1"` and the attribute `wiredByImpl` MUST be set to `"true"`.
[SBPEL2013]	If neither [SBPEL2010] nor [SBPEL2011] nor [SBPEL2012] apply, then the multiplicity MUST be `"0..1"`.
[SBPEL2014]	If `initializePartnerRole="yes"` is specified for a partner link and the partner link maps to a service in the component type, then any component that uses this business process as an implementation MUST configure the corresponding service to use a binding that knows the identity of the partner as soon as the partner link becomes active (e.g. the binding cannot depend on using a “reply-to” field as the mechanism to initialize the partner role).
[SBPEL2015]	The WSDL port type in the `<interface.wsdl>` declaration for the service in the introspected component type MUST be the same as the port type of the `myRole` of the partner link.
[SBPEL2016]	If the partner link type has two roles, then the `<interface.wsdl>` declaration MUST also have a `@callbackInterface` attribute whose value points to the same WSDL port type as the `partnerRole` of the partner link.
[SBPEL2017]	The WSDL port type in the `<interface.wsdl>` declaration for the reference MUST be the same as the port type of the `partnerRole` of the partner link.
[SBPEL2018]	If the partner link type has two roles, then the `<interface.wsdl>` declaration MUST also have a `@callbackInterface` attribute whose value points to the same WSDL port type as the `myRole` of the partner link.
[SBPEL2019]	When multiple partner links share the same name, the scheme defined by [SBPEL2020]-[SBPEL2022] MUST be used to disambiguate different occurrences of partner link declaration.
[SBPEL2020]	The introspected component type MUST include services or references corresponding to these partner links with names: `"_orginalName_1"` to `"_orginalName_N"`.
[SBPEL2021]	The number suffixes for the partner links MUST be based on the lexical order of the corresponding partner link occurrences in the process definition.
[SBPEL2022]	If any `"_orginalName_i"` (where 1 <= i <= N) is already the name of a partner link declaration in the process definition, additional underscore characters MAY be added at the beginning of all aliases consistently to avoid collision.
[SBPEL3001]	The name of a variable used as a property of the component MUST be unique within the process.
[SBPEL3002]	If a value is provided for a property, any initialization from-spec MUST still be evaluated, but the value of the variable will be changed to the provided property value immediately after the

	initialization is evaluated, and specifically, before any following variable initialization from-spec is evaluated.
[SBPEL3003]	If a BPEL variable that is used as a property has an initialization from-spec then `mustSupply="false"` MUST be specified on the component type property declaration, even if the default value is not literal and therefore not represented in the component type.
[SBPEL3004]	In a WS-BPEL process definition, a variable MAY include an `sca-bpel:multiReference` extension element that declares that the variable represents a multi-valued reference.
[SBPEL3005]	When a variable declaration contains the `sca-bpel:multiReference` extension, the type of the variable MUST be an element of sca-bpel:serviceReferenceList.
[SBPEL3006]	The introspected component type MUST include a reference with a multiplicity of either `"0..n"` or `"1..n"` that corresponds to a variable with the `sca-bpel:multiReference` element.
[SBPEL3007]	The type of the reference MUST be determined by the partner link type and the partner role attributes of the `sca-bpel:multiReference` extension element.
[SBPEL3008]	The `sca-bpel:multiRefFrom` attribute MUST NOT be specified for a partner link with a `myRole` attribute referencing a role which is the only role of a partner link type.
[SBPEL3009]	The `sca-bpel:multiRefFrom` attribute MUST NOT be specified for a partner link that has the `sca-bpel:service` attribute.
[SBPEL3010]	The value of the `sca-bpel:multiRefFrom` attribute MUST refer to the name of a variable manifesting an SCA multi-valued reference.
[SBPEL3011]	The `partnerLinkType` and `partnerRole` attributes of the partner link and multi-valued reference variable MUST be matched.
[SBPEL3012]	There MUST be at least one code-path where the values from the multi-valued reference variable are copied to the `partnerRole` of the partner link.
[SBPEL3013]	A WS-BPEL process definition MAY override the default mapping of partner links to services or references as described in section 2.1 by explicitly marking the partner link with an SCA attribute that describes the service or reference.
[SBPEL3014]	To explicitly map a partner link to a service, the `sca-bpel:service` attribute MAY be specified for the partner link.
[SBPEL3015]	The name of the service specified in the `sca-bpel:service` attribute MUST NOT conflict with any other service name generated in the component type for this process.
[SBPEL3016]	The `sca-bpel:service` attribute MUST NOT be specified for a partner link with a `partnerRole` attribute referencing a role which is the only role of a partner link type.
[SBPEL3017]	To explicitly map a partner link to a reference, the `sca-`

	`bpel:reference` attribute MAY be specified for the partner link.
[SBPEL3018]	The name of the reference specified in the `sca-bpel:service` attribute MUST NOT conflict with any other reference name generated in the component type for this process.
[SBPEL3019]	The `sca-bpel:reference` attribute MUST NOT be specified for a partner link with a `myRole` attribute referencing a role which is the only role of a partner link type.
[SBPEL3020]	A process MUST NOT include both `sca-bpel:service` and `sca-bpel:reference` attributes on a single partner link.
[SBPEL3021]	An SCA extension attribute `sca-bpel:requires` MAY be used to declare required policy intents on a partner link.
[SBPEL3022]	The contents of the `sca-bpel:requires` attribute MUST be a space separated list of SCA intent QNames, exactly as specified in the SCA Policy Framework Specification for the contents of the `@sca:requires` attribute.
[SBPEL3023]	If the `sca-bpel:requires` attribute is specified, the corresponding service or reference in the introspected component type MUST include an `@sca:requires` attribute with the same contents.

C. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Members of the SCA-BPEL Technical Committee:

Najeeb Andrabi, TIBCO Software Inc.
Graham Barber, IBM
William Barnhill, Booz Allen Hamilton
Charlton Barreto, Adobe Systems
Hanane Becha, Nortel
Michael Beisiegel, IBM
Jeffrey Bik, Active Endpoints, Inc.
David Booz, IBM
David Burke, TIBCO Software Inc.
Fred Carter, AmberPoint
Martin Chapman, Oracle Corporation
Eric Clairambault, IBM
James Bryce Clark, OASIS
Mark Combellack, Avaya, Inc.
Kevin Conner, Red Hat
Jean-Sebastien Delfino, IBM
Jacques Durand, Fujitsu Limited
Mike Edwards, IBM
Raymond Feng, IBM
Mark Ford, Active Endpoints, Inc.
Genadi Genov, SAP AG
Alejandro Guizar, Red Hat
Uday Joshi, Oracle Corporation
Khanderao Kand, Oracle Corporation
Anish Karmarkar, Oracle Corporation
Jason Kinner, Oracle Corporation
Dieter Koenig, IBM
Rich Levinson, Oracle Corporation
Mark Little, Red Hat
Ole Madsen, OIOXML eBusiness Standardization Group
Ashok Malhotra, Oracle Corporation
Keith McFarlane, Avaya, Inc.
Jeff Mischkinsky, Oracle Corporation
Simon Moser, IBM
Sanjay Patil, SAP AG
Michael Pellegrini, Active Endpoints, Inc.

Luciano Resende, IBM
Michael Rowley, Active Endpoints, Inc.
Paul Tazbaz, Wells Fargo
Clifford Thompson, Individual
Ivana Trickovic, SAP AG
Danny van der Rijn, TIBCO Software Inc.
Mark Walker, Avaya, Inc.
Prasad Yendluri, Software AG, Inc.
Alex Yiu, Oracle Corporation

OSOA Contributors:

Martin Chapman, Oracle
Sabin Ielceanu, TIBCO Software Inc.
Dieter Koenig, IBM
Michael Rowley, BEA Systems, Inc.
Ivana Trickovic, SAP AG
Alex Yiu, Oracle

D. Revision History

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
2	2007-10-10	Dieter König	Issue resolutions BPEL-4, BPEL-7 New section “5. Conformance” List of XML namespaces Table of Contents formatting References formatting Syntax and Examples formatting
3	2007-10-10	Dieter König	Reduced component/composite syntax in sections 1 and 2
4	2007-12-05	Dieter König	Issue resolutions BPEL-5, BPEL-6, BPEL-9, BPEL-13 Document title according to OASIS rules
5	2008-01-11	Michael Rowley	Issue resolution for BPEL-11
6	2008-01-17	Dieter König	Approved Committee Draft
7	2008-03-17	Dieter König	Revised Approved Committee Draft Applied resolution to BPEL-19: Added XML Schema definitions as Appendix A
8	2008-03-27	Michael Rowley	Applied resolution to BPEL-14
9	2008-04-10	Michael Rowley	Added @sca-bpel:requires attribute, also as part of resolving BPEL-14.
CD01-rev5	2008-06-19	Michael Rowley	Reworked 2.1 to use 2119 language. Removed Alex Yiu from editor list.
CD01-rev7	2008-07-07	Najeeb Andrabi	Reverted 2.1 to CD01-rev2 Issue resolutions BPEL-3
CD01-rev8	2008-07-10	Dieter König	Namespace prefix "xsd" used consistently in all XML Schema snippets Fixed definition of `sca-bpel:requires` attribute in section 3.4 and XML Schema Added import for `sca-core.xsd` to the XML schema defining WS-BPEL extension attributes
CD01-rev9	2008-07-10	Michael Rowley	Marked Chapter 4 Non-Normative (issue 1) Reapplied changes from rev5. Added section 2.6 (issue 17) and added sca-bpel:implementationRef to XML Schema

CD01-rev10	2008-09-5	Martin Chapman	Backed out changes for section 2.1, which was a partial proposal for Issue 18. Added new section 2.7 (issue 2). Change Section 3 Intro (issue 20).
CD01-rev11	2008-09-5	Michael Rowley	Rewrite of section 2.1 for clearer 2119 requirements (accepted by the TC 16-oct-08)
CD01-rev12	2008-10-23	Michael Rowley	2119 language for the rest of the document.
CD01-rev13	2008-10-30	Michael Rowley	Updates 2119 language through sections 2.2 based on TC call of Oct-30.
CD01-rev14	2008-10-30	Michael Rowley	Removed interface.partnerlink as per Issue 22. 2119 updates starting at section 2.3.
CD01-rev15	2009-02-07	Dieter König	Issue resolutions BPEL-23, BPEL-25. All conformance statements labeled. New Appendix section added containing a table of all conformance statements. Minor formatting improvements.
CD01-rev16	2009-02-26	Michael Rowley	Issue resolutions for BPEL-12, BPEL-24, BPEL-28, BPEL-29 and BPEL-27
CD02	2009-03-05	Dieter König	Committee Draft 02 and Public Review Draft 01

619