

SCA Service Component Architecture

SCA服务构件架构

WS-BPEL客户端及实现模型

SCA Service Component Architecture

Client and Implementation Model Specification for WS-BPEL

SCA Version 1.00, March 21 2007

Technical Contacts:

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

Copyright Notice

© Copyright BEA Systems, Inc., Cape Clear Software, International Business Machines Corp, Interface21, IONA Technologies, Oracle, Primeton Technologies, Progress Software, Red Hat, Rogue Wave Software, SAP AG., Siemens AG., Software AG., SunMicrosystems, Inc., Sybase Inc., TIBCO Software Inc., 2005, 2007. All rights reserved.

License

The Service Component Architecture Specification is being provided by the copyright holders under the following license. By using and/or copying this work, you agree that you have read, understood and will comply with the following terms and conditions:

Permission to copy and display the Service Component Architecture Specification and/or portions thereof, without modification, in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the Service Component Architecture Specification, or portions thereof, that you make:

1. A link or URL to the Service Component Architecture Specification at this location:
 - <http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>
2. The full text of this copyright notice as shown in the Service Component Architecture Specification.

BEA, Cape Clear, IBM, Interface21, IONA, Oracle, Primeton, Progress Software, Red Hat, Rogue Wave, SAP, Siemens AG., Software AG., Sun, Sybase, TIBCO (collectively, the "Authors") agree to grant you a royalty-free license, under reasonable, non-discriminatory terms and conditions to patents that they deem necessary to implement the Service Component Architecture Specification.

THE Service Component Architecture SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, REGARDING THIS SPECIFICATION AND THE IMPLEMENTATION OF ITS CONTENTS, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OR TITLE.

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE SERVICE COMPONENT ARCHITECTURE SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Service Component Architecture Specification or its contents without specific, written prior permission. Title to copyright in the Service Component Architecture

Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Status of this Document

This specification may change before final release and you are cautioned against relying on the content of this specification. The authors are currently soliciting your contributions and suggestions. Licenses are available for the purposes of feedback and (optionally) for implementation.

IBM is a registered trademark of International Business Machines Corporation in the United States, other countries, or both.

BEA is a registered trademark of BEA Systems, Inc.

Cape Clear is a registered trademark of Cape Clear Software

IONA and IONA Technologies are registered trademarks of IONA Technologies plc.

Oracle is a registered trademark of Oracle USA, Inc.

Progress is a registered trademark of Progress Software Corporation

Primeton is a registered trademark of Primeton Technologies, Ltd.

Red Hat is a registered trademark of Red Hat Inc.

Rogue Wave is a registered trademark of Quovadx, Inc

SAP is a registered trademark of SAP AG.

SIEMENS is a registered trademark of SIEMENS AG

Software AG is a registered trademark of Software AG

Sun and Sun Microsystems are registered trademarks of Sun Microsystems, Inc.

Sybase is a registered trademark of Sybase, Inc.

TIBCO is a registered trademark of TIBCO Software Inc.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

目录

<i>SCA SERVICE COMPONENT ARCHITECTURE</i>	I
COPYRIGHT NOTICE	III
LICENSE	III
STATUS OF THIS DOCUMENT	IV
目录	V
1. WS-BPEL 客户端及实现模型	6
1.1 目标	6
1.2 WS-BPEL 流程作为构件实现	6
1.3 WS-BPEL 流程定义的构件类型	7
1.3.1 服务与引用	7
1.3.2 伙伴连接类型和 SCA 接口	8
1.3.3 指定具有伙伴连接类型的 SCA 接口	9
1.3.4 本地伙伴连接的处理	10
1.3.5 对会话接口的支持	10
1.4 WS-BPEL 的 SCA 扩展	11
1.4.1 属性	11
1.4.2 多值引用	12
1.5 SCA 中使用 BPEL4WS 1.1	14
2. 附录	14
2.1 XML SCHEMA	14
2.2 参考文献	16
3. 后记: SCA 中文规范项目	17
3.1 声明	17
3.2 致谢	17
3.3 参与人员	17
3.4 讨论、建议和勘误	19
3.5 技术圈子	19
3.6 满江红	19
3.7 招募	19

1. WS-BPEL 客户端及实现模型

1.1 目标

SCA WS-BPEL客户端以及实现模型说明了WS-BPEL 2.0如何与SCA一起使用。本规范的目标是描述如下的应用场景。

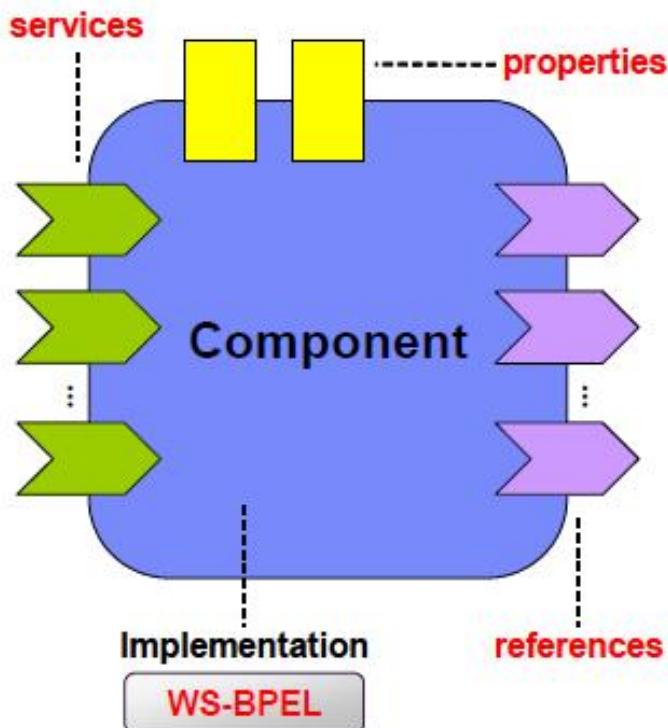
从WS-BPEL流程开始。在SCA中，可以使用任何有效的WS-BPEL流程定义作为一个构件的实现。它还可以从任何WS-BPEL流程定义产生一个SCA 构件类型，并且在一个SCA装配中使用。大多数BPEL4WS 1.1 流程定义可以通过使用在1.5章中描述的向后兼容方法来被SCA所使用。

从SCA 构件类型开始。使用WS-BPEL来实现仅使用WSDL接口来定义服务和引用的SCA构件类型是可行的，也可能需要在流程定义中使用一些SCA相关的扩展。

从带有SCA扩展的WS-BPEL开始。可以创建一个使用SCA扩展的WS-BPEL流程定义并且产生一个SCA 构件类型，然后在一个SCA 装配中使用这个类型。一些SCA特性（如properties和multi-party 引用）只能被使用SCA 扩展的WS-BPEL流程定义所使用。

1.2 WS-BPEL 流程作为构件实现

WS-BPEL 流程定义能够被当作 SCA 构件的实现来使用。



这样的构件定义具有如下形式：

```
<component name="xs:NCName">*
  <implementation.bpel process="xs:QName" />
```

```

<property name="xs:NCName" source="xs:string"? file="xs:anyURI"?>*
  property-value?
</property>
<reference name="xs:NCName" />*
  wire-target-URI
</reference>
</component>
  
```

唯一和 WS-BPEL 有关的是<implementation.bpel> 元素。它的 process 属性指定了一些可执行的 WS-BPEL 流程的目标 QName。

1.3 WS-BPEL 流程定义的构件类型

WS-BPEL 流程定义可以用于构件的实现，它同时也确定了任何使用这个实现的 SCA 构件的 ComponentType。这个构件类型描述 SCA 需要知道的实现方面以支持装配和使用该实现的构件的装配和部署。SCA 装配规范[1]中定义的构件类型的一般形式如下：

```

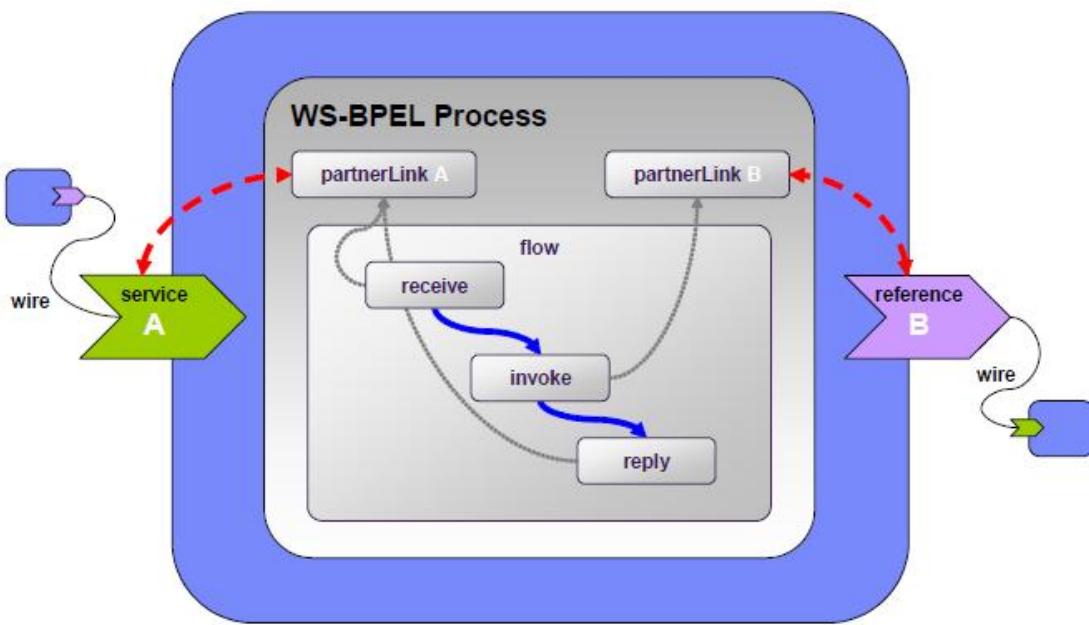
<componentType xmlns="http://www.osoa.org/xmlns/sca/1.0" >
  <service name="xs:NCName">*
    <interface/>
  </service>
  <reference name="xs:NCName" override="sca:OverrideOptions"?*
    multiplicity="0..1 or 1..1 or 0..n or 1..n"?>*
    <interface/>
  </reference>
  <property name="xs:NCName" (type="xs:QName" | element="xs:QName")*
    many="xs:boolean"? required="xs:boolean"?>* default-property-value?
  </property>
</componentType>
  
```

构件类型可以通过内省从 WS-BPEL 流程定义生成。

1.3.1 服务与引用

在 SCA 中，服务和引用都符合 WS-BPEL 伙伴连接的概念，服务和引用的区别由在一个会话中哪一方发送第一条消息决定。不管这个双向会话有多少条消息以及它持续了多久，但总会存在首条消息。发送首条消息的一方就被认为是客户端，而接收方则为服务提供者。从服务提供者到客户端的消息称为回调消息。

从谁发送首条消息这个方面来说，WS-BPEL 的伙伴连接毫无区别。因此为了将 WS-BPEL 流程映射到 SCA 构件类型，有必要知道是哪一方发送了首条消息。对控制流进行的简单静态分析（不涉及确定任何表达式的值）可以确定哪一方可以发送首条消息。



Services:如果流程静态分析判定到达伙伴连接的首条消息可能在一个<receive>活动、<pick>活动的<onMessage>元素或事件处理器的<onEvent>元素中被接收，那么伙伴连接在此构件类型中存在对应的 SCA 服务。如果在伙伴连接的声明中设置了 initializePartnerRole="yes"，那么这个服务必须要使用一个绑定，这个绑定能在伙伴连接激活的同时获知其对应的伙伴的标识（例如，绑定不能依靠使用“reply-to”字段作为初始化伙伴角色的机制）。

References:如果流程的静态分析无法判定该伙伴连接应该映射到一个 SCA 服务，那么此伙伴连接被映射为该构件类型中的一个 SCA 引用。

引用的重数由如下算法决定：

1. **多重引用.** 如果伙伴连接使用 `sca:multiRefFrom="aVariableName"` 扩展进行声明，那么 SCA 引用的重数应该由相应变量所使用的 `sca:multiReference` 扩展中的 `multiplicity` 属性来决定。变量的重数声明要么是 `0..n` 要么是 `1..n`。扩展的具体细节在 1.4.2 章节介绍。
2. **必需的引用.** 如果不为 (1) 并且伙伴连接设置了 `initializePartnerRole="yes"`，那么重数为 `1..1` (即是必需的引用)。
3. **Stub 引用.** 如果不为 (1) 或 (2) 的情况并且流程分析判定首次在活动中使用的伙伴连接是在设置了伙伴角色的赋值活动中，那么 `multiplicity` 的值为 “`0..1`” 并且 `wiredByImpl` 属性设置为 `true`。带有 `wiredByImpl="true"` 的引用为 Stub 引用。尽管无法为该引用设置目标，SCA 仍可为其应用绑定和策略，如果该接口为双向接口，也许还需要为回调设置端点地址。
4. **可选引用.** 如果不为 (1)、(2) 或 (3) 的情况，那么 `multiplicity="0..1"`.

对于服务和引用两者来说，如果名字唯一，那么服务名或引用名就是伙伴连接名（如何处理有歧义的状况请参考后面的处理本地伙伴连接章节）。

1.3.2 伙伴连接类型和 SCA 接口

当一个伙伴连接确定对应于一个 SCA 服务时，该服务的类型由此伙伴连接的类型所决定。伙伴连接作为 `myRole` 指定的角色提供了该服务的 WSDL 端口类型。如果此伙伴连接类型有两个角色，那么

`partnerRole` 提供此回调接口 WSDL 端口类型。

考虑这样一个例子，它用到了作为 WS-BPEL 规范中的例子而被使用的一种伙伴连接类型。此伙伴连接类型的定义如下：

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

提供发票服务的“`invoiceProcess`”将定义一个使用如下类似的类型声明的伙伴连接

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

在流程的某处，一个开始活动会使用这个伙伴连接，类似如下：

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

因为该伙伴连接被用在一个开始活动中，SCA 将该伙伴连接映射到一个服务上。这种情况下构件类型的服务元素如下：

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

反过来，当一个伙伴连接确定对应到一个 SCA 引用时，伙伴连接作为 `partnerRole` 指定的角色提供了该引用的 WSDL 端口类型。如果这个伙伴连接类型拥有两个角色，那么 `myRole` 提供的就是回调接口的 WSDL 端口类型。

1.3.3 指定具有伙伴连接类型的 SCA 接口

在上面提到的方法中，服务和引用的 SCA 定义使用 `<interface.wsdl>` 来重申接口和已在 WS-BPEL `partnerLinkType` 中出现的回调接口间的关联。`partnerLinkType` 通过指定服务在会话中的角色定义了两

个服务间的关联。一个 partnerLinkType 至少指定一个角色。

对于更喜欢 WS-BPEL 元素的用户来说，为接口类型定义不同形式的 partnerLinkType 是可能的。这种形式提供的信息并不会比<interface.wsdl>元素能提供的信息更多。上文的例子看起来类似这样：

```
<interface.partnerLinkType type="lns:invoicingLT"
    serviceRole="invoiceService"/>
```

接口类型定义的一般形式如下：

```
<interface.partnerLinkType type="xs:QName"
    serviceRole="xs:NCName"?/ >
```

Type 属性是必需的并引用了一个伙伴连接类型。如果该伙伴连接类型有两个角色，那么必须使用可选属性 serviceRole 来指定哪一个角色作为接口使用。另一个角色则用来作为回调接口。如果 partnerLinkType 只有一个角色，那它不能是一个回调，而且 serviceRole 属性也可被省略。

相对 interface.wsdl 形式，这种形式拥有很多优点。它更简洁，无需重申接口和回调接口间的联系。因此该形式的使用能使伙伴连接类型改变用来定义某种角色的 portType，而且不需要为那些构件类型改变接口的定义，从而使得用到该伙伴连接类型的所有的 SCA 构件类型能保持正确。这种形式可能对某些用户更为熟悉。

1.3.4 本地伙伴连接的处理

WS-BPEL 中除了在<process>级别声明伙伴连接之外，声明本地于<scope>的伙伴连接也是可能的。不同的<scope>中定义的伙伴连接名可能隐式共享相同的名字。

如果遇到这种名字相同的情况，下列方案用于区分伙伴连接声明的不同场合：

- 在多个伙伴连接声明中假定"originalName"是最初使用的 NCName
- 这些伙伴连接暴露给 SCA 装配时，不管它们以什么形式（即服务与引用）加入 SCA 装配，它们都要给出从"_orginalName_1"到"_orginalName_N"的别名。
- 别名定义过程中如果任何一个"_orginalName_i"（其中 $1 \leq i \leq N$ ）别名已经被存在的伙伴连接声明使用，附加的下划线可以加在别名前面以避免冲突。

1.3.5 对会话接口的支持

WS-BPEL 能用来实现提供会话服务的 SCA 构件。参考 SCA 装配规范[1]对会话接口的描述。当一个被标记为会话式的接口用作伙伴连接的一个角色时，并不需要用其它机制（如 WS-BPEL 关联机制）来关联伙伴连接上的消息，当然关联也是支持的。这即是说，SCA 会话接口是被当作一个隐形的关联机制来关联在单一会话中的伙伴连接上的双向消息交换。当伙伴角色的 EPR 初始化时，一个新会话必须被用来执行会话服务的一条操作。

任何流程在经过静态分析后必须能确保在 endsConversation 操作完成之后使用会话接口的操作都会被拒绝。

如果静态分析无法确认在 `endsConversation` 操作完成之后使用会话接口的操作会被拒绝，那么在会话结束后再使用会话伙伴连接将会产生一个 `sca:ConversationViolation` 错误。参考 SCA 装配规范[1]，第 1.5.3 节描述了这个错误。

要重点指出的是，WS-BPEL 关联机制并不局限于单一的伙伴连接。它可以用来协助不同的伙伴连接上的信息交换到一个特别的 WS-BPEL 流程实例。

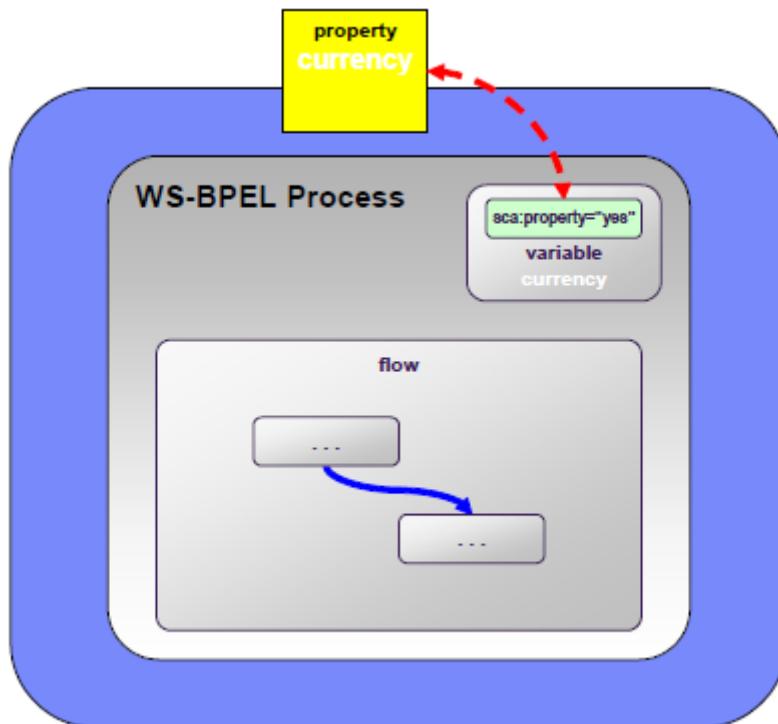
1.4 WS-BPEL 的 SCA 扩展

在 SCA 中使用 WS-BPEL 流程，而流程可能对 SCA 一无所知。一些 SCA 的概念只对一些支持 SCA 特定扩展的 WS-BPEL 处理器有用。WS-BPEL 识别 SCA 的能力由 SCA 扩展提供，流程定义中需要增加如下声明：

```
<process ...>
  <extensions>
    <extension namespace="http://www.osoa.org/xmlns/sca/bpel/1.0"
      mustUnderstand="yes" />
  </extensions>
```

1.4.1 属性

WS-BPEL 变量声明可能包含一个 SCA 扩展，表明这个变量作为一个构件的 SCA 属性，而这个构件使用 WS-BPEL 流程来表示。



声明类似如下：

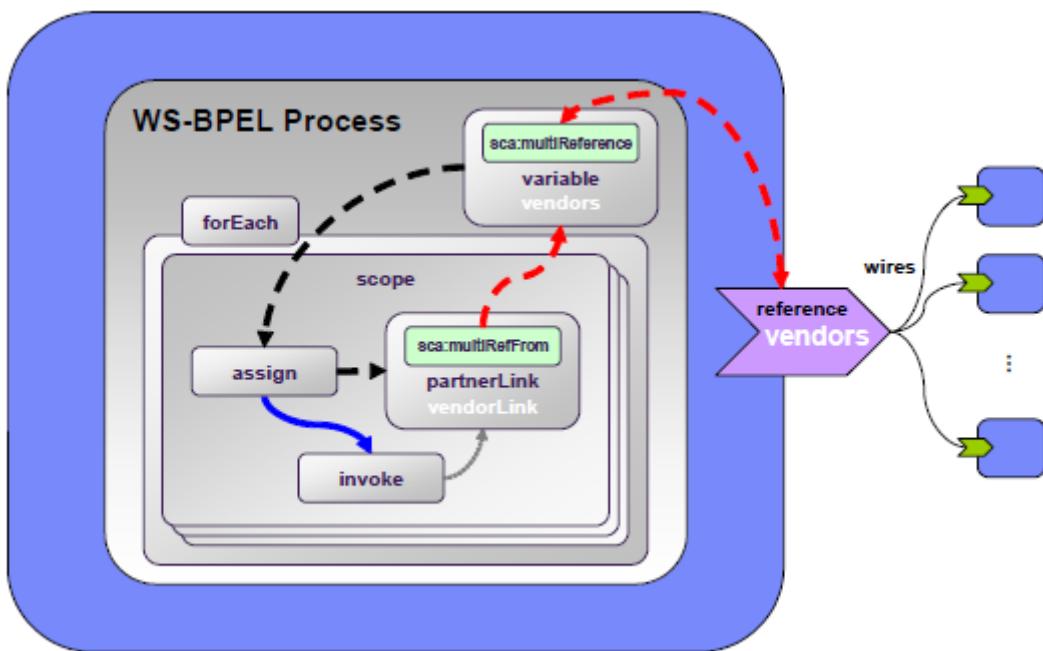
```
<variable name="currency" type="xsd:string" sca:property="yes" />
```

当在一个变量声明中设置 `sca:property="yes"` 的时候，变量名作为由 WS-BPEL 流程描述的构件类型的属性名，在此处理流程中必须唯一。

运行时变量将会按照构件定义提供的值进行初始化，该构件定义使用了 WS-BPEL 流程。

1.4.2 多值引用

构件类型可以声明多重引用，即一个引用被连线到多个目标上。该能力的一个示范用例是购买构件连接到一组可接受的供应商。SCA 假定每种编程语言的绑定都会在编程语言内部为保证目标列表可用提供自己的方法。



WS-BPEL 中一个变量可能包含一个 `sca:multiReference` 扩展元素，它声明的变量表示一个多值引用。变量类型必须是 `sca:serviceRefList` 的元素之一。然而，既然该类型仅仅指定该变量包含一个端点引用的列表，`sca:multiReference` 元素也有属性来指定伙伴连接类型和目标引用的伙伴角色。一个用来表示引用列表的变量的例子看起来将会是如下这样：

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

该扩展的语法是：

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

`Multiplicity` 的默认值为 `1..n`。

`sca:serviceReferenceList` 元素声明如下：

```
<xsd:element name="serviceReferenceList">
```

```

<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="bpel:service-ref" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
</xsd:element>

```

典型的一个包含多值引用变量的应用，将是包含一个具有`<forEach>`活动，其包含列表中的每个元素的迭代。`<forEach>`活动的主体将声明一个本地伙伴连接和分派该列表中的一个元素给这个伙伴连接。一个这样的本地伙伴连接，一般会被列为“References”，就像在第 1.3.1 章节的案例 1 提到的那样。

为使 SCA 模型更有效，另一个 SCA 扩展被用来去关联一个多值引用，显然的，其被用来作为具有一个伙伴连接的“`sca:serviceReferenceList`”变量。该扩展在一个绑定到伙伴连接定义的属性 `form` 中。

该扩展的语法是：

```
sca:multiRefFrom="xs:NCName"
```

其属性值必须对应到一个变量名，表示一个 SCA 多值引用。该伙伴连接的伙伴连接类型和伙伴角色属性以及多值引用变量也必须匹配。并且至少应该有一个 `code-path`，其来自于该多值引用变量的数据被复制到该伙伴连接的伙伴角色。

如果违反了任何一个上述的约束，静态分析时都会被认为是错误。

当这个 `sca:multiRefFrom` 扩展被应用来给一个多值引用变量和一个伙伴连接配对时，伙伴连接和变量作为使用该变量名的 SCA 装配模型中的单一多值引用实体，其中的伙伴连接按“References”案例 1 进行分类（如第 1.3.1 节所述）。如果包含的接口是双向的，此即意味该双向接口的连线在 SCA 中是作为单一的引用出现。

例如：

```

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

```

```

  ...
<assign>
  <copy>
    <from>$vendors/bpel:service-ref[$idx]</from>
    <to partnerLink="vendorLink" />
  </copy>
</assign>
  ...
</scope>
</forEach>

```

一个名为 `vendors` 的多值引用在上述示例中被声明。命名为 `vendorLink` 的伙伴连接，其被作为“References”案例 1 进行分类，不表明直接进入 SCA 装配模型。另外的 `sca:multiRefFrom="vendors"` 扩展关联了具有多值引用变量“`vendors`”的“`vendorLink`”伙伴连接。因此，该伙伴连接和变量显然作为一个叫做“`vendors`”的单一的多值引用。这使得 SCA 装配模型易于参照。

1.5 SCA 中使用 BPEL4WS 1.1

BPEL4WS 1.1 流程定义可以用作 SCA 构件的实现。在 1.2 节中介绍的语法用来定义一个把 BPEL4WS 1.1 流程作为实现的构件。此例中 `process` 属性指定了一个 BPEL4WS 1.1 可执行流程的目标 QName。

BPEL4WS 1.1 流程定义可用来生成 SCA 构件类型。

2. 附录

2.1 XML Schema

```

XML Schema sca-implementation-bpel.xsd
<xsd:schema xmlns="http://www.osoa.org/xmlns/sca/1.0"
  targetNamespace="http://www.osoa.org/xmlns/sca/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <xsd:include schemaLocation="sca-core.xsd"/>
  <xsd:element name="implementation.bpel"
    type="BpelImplementation"
    substitutionGroup="implementation"/>
  <xsd:complexType name="BpelImplementation">
    <xsd:complexContent>
      <xsd:extension base="Implementation">

```

```

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

<xsd:element name="interface.partnerLinkType"
  type="BpelPartnerLinkType"
  substitutionGroup="interface" />
<xsd:complexType name="BpelPartnerLinkType">
  <xsd:complexContent>
    <xsd:extension base="Interface">
      <xsd:sequence>
        <xsd:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:QName"
        use="required"/>
      <xsd:attribute name="serviceRole" type="xsd:NCName"
        use="optional"/>
      <xsd:anyAttribute namespace="##any" processContents="lax" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:schema>

```

XML Schema sca-bpel.xsd

```

<xsd:schema xmlns="http://www.osoa.org/xmlns/sca/bpel/1.0"
  targetNamespace="http://www.osoa.org/xmlns/sca/bpel/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sref="http://docs.oasis-open.org/wsbpel/2.0/serviceref"
  elementFormDefault="qualified">

  <xsd:import namespace="http://docs.oasis-open.org/wsbpel/2.0/serviceref"
    schemaLocation="ws-bpel_serviceref.xsd" />

  <xsd:simpleType name="tMultiplicity">
    <xsd:restriction base="xsd:string">

```

```
<xsd:enumeration value="0..n"/>
<xsd:enumeration value="1..n"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="tBoolean">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="yes"/>
<xsd:enumeration value="no"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:attribute name="property" type="tBoolean"/>

<xsd:attribute name="multiRefFrom" type="xsd:NCName"/>

<xsd:element name="multiReference" type="tMultiReference"/>
<xsd:complexType name="tMultiReference">
<xsd:attribute name="partnerLinkType" type="xsd:QName"
use="required"/>
<xsd:attribute name="partnerRole" type="xsd:NCName"
use="required"/>
<xsd:attribute name="multiplicity" type="tMultiplicity"
use="optional" default="1..n"/>
</xsd:complexType>

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

2.2 参考文献

- [1] SCA Assembly Model Specification, Version 1.0, February 2007
http://www.osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf
- [2] Web Services Business Process Execution Language Version 2.0, November 2006
<http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>
- [3] Business Process Execution Language for Web Services Version 1.1, May 2003

<http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp>
<http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>
<http://msdn2.microsoft.com/en-us/library/aa479359.aspx>
<https://www.sdn.sap.com/irj/sdn/developerareas/esa/standards>
<http://www.siebel.com>

3. 后记：SCA 中文规范项目

3.1 声明

SCA 相关规范的中文文档得到 OSOA Chinese Community (OSOA 中文社区) 的直接授权和大力的支持，其目的是在中文世界推广优秀的 SOA 相关技术标准。本次翻译活动由满江红开放技术研究组织 (<http://www.redsaga.com>) 和 OSOA 中文社区 (<http://www.osoa.org/pages/viewpage.action?pageId=416>) 共同发起、组织，将翻译 SCA 的绝大多数规范，并得到 goCom 社区 (<http://www.gocom.cc/>) 的赞助。我们在此郑重宣布，本次翻译遵循原 Service Component Architecture Specification 的授权协议。在完整保留全部文本包括本版权页，并不违反 Service Component Architecture Specification 协议的前提下，允许和鼓励任何人进行全文转载及推广。我们在此宣布所有参与人员放弃除署名权外的一切权利。

3.2 致谢

SCA 规范的翻译对很多人来说都是一个挑战，再次感谢所有参与翻译、评审工作的同学们，没有你们的辛勤劳动，也就不会有中文规范的面试。

3.3 参与人员

规范分配情况，最后的校对和统稿由管理员统一完成。

Specification	翻译	一审	二审
SCA Assembly Model V1.00	ligang1111	wangfeng	wangfeng
SCA Policy Framework V1.00	liang_ma,max	pesome	needle
SCA Transaction Policy V1.00	max	jiangxd	guangh
SCA Java Common Annotations and APIs V1.00	ligang1111	HiugongGwok	dc
SCA Java Component Implementation V1.00	ligang1111	pinelygao	nekesai
SCA Spring Component Implementation V1.00	ligang1111	pinelygao	Caozw
SCA BPEL Client and Implementation V1.00	jiangxd	hongsoft	hongsoft

SCA C++ Client and Implementation V1.00	SpringWater	SpringWater s	SpringWater s
SCA Web Services Binding V1.00	xichengmylove	hongsoft	hongsoft
SCA JMS Binding V1.00	YuLimin/Echo	hongsoft	pegasus
SCA EJB Session Bean Binding V1.00	max	hongsoft	yanfei
SCA JCA Binding V1.00 (暂不发布)	bsspirit	jiangxd	benja

参与人员列表:

网名	姓名	网名	姓名
billytree	冯博	guangh	光华
ligang1111	李刚	Caozw	小曹
xichengmylove	严永华	yanfei	晏斐
pesome	张俊	benja	老贾
max	孙浩	HiugongGwok	郭晓刚
needle	needle	bsspirit	张丹
YuLimin	俞丽敏	Echo	杨春花
wangfeng	王锋	pinelygao	高松
hongsoft	洪波	liang_ma	马亮
jiangxd	姜晓东	Dc	王葱权
SpringWater	张世富	pegasus	马捷
chriscchengzh	chris	nekesai	蔡永保
jiaoly	老焦	keqiang	克强

下列人员报名，但由于种种原因没有参与，在此感谢:

网名	姓名	网名	姓名
jetyang_1	杨文明	cenwenchu	岑文初
lafay	叶江	Eric	孟庆余
laodingshan	丁跃斌		